

# **Development Activities of the Frame Processor for the Athena WFI Instrument**

## **Dissertation**

der Mathematisch-Naturwissenschaftlichen Fakultät  
der Eberhard Karls Universität Tübingen  
zur Erlangung des Grades eines  
Doktors der Naturwissenschaften  
(Dr. rer. nat.)

vorgelegt von  
Samuel Pliego Caballero  
aus Mexiko Stadt/Mexiko

Tübingen  
2026

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der  
Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:

22.07.2026

Dekan:

Prof. Dr. Thilo Stehle

1. Berichterstatter:

Prof. Dott. Andrea Santangelo

2. Berichterstatter:

Prof. Dr. Josef Jochum



Athena is the next generation X-ray observatory to be launched by the European Space Agency (ESA), bringing several innovations and higher performance than any previous X-ray mission. Two main instruments will be flying on board Athena: the X-ray Integral Field Unit (X-IFU) and the Wide Field Imager (WFI). The first consists of a microcalorimeter instrument based on transition edge sensor technology, and the second one consists of state-of-the-art DEPFET sensors providing unprecedented energy resolution and high-speed readout capabilities. The focus of this work lies on the WFI instrument, specifically on the design and development of the frame pre-processing pipeline, which is a fundamental element of the WFI's detector electronics subsystem.

The unprecedented high-count rate capabilities of the WFI instrument imply that an enormous amount of data is produced during WFI observations. This precious data far surpasses the current bandwidth capabilities of modern spacecraft systems, creating the necessity of on-board data screening and preprocessing, downloading then only the scientifically relevant X-ray data to the ground.

A demonstrator model of the WFI frame processor has been designed, developed, and manufactured, demonstrating successfully the technological capability of processing on-board in real-time the WFI data utilizing a pipeline concept in a single space qualified FPGA device. This thesis focuses on the description of such pipelines, outlining the design challenges, and concluding by providing the verification and validation results obtained during the frame processor demonstrator model development.



## II ZUSAMMENFASSUNG

Athena ist das Röntgenobservatorium der nächsten Generation, das von der Europäischen Weltraumorganisation (ESA) ins All gebracht wird. Es bietet zahlreiche Innovationen und eine höhere Leistungsfähigkeit als jede bisherige Röntgenmission. An Bord von Athena kommen zwei Hauptinstrumente zum Einsatz: die X-ray Integral Field Unit (X-IFU) und der Wide Field Imager (WFI). Während das X-IFU aus einem Mikrokalorimeter auf Basis der Transition-Edge-Sensor-Technologie besteht, verfügt das WFI über hochmoderne DEPFET-Sensoren, die eine beispiellose Energieauflösung und schnelle Auslesefähigkeiten bieten. Der Schwerpunkt dieser Arbeit liegt auf dem WFI-Instrument, insbesondere auf dem Entwurf und der Entwicklung der Vorverarbeitungs-Pipeline für Bilddaten. Diese stellt ein grundlegendes Element des Detektorelektronik-Subsystems des WFI dar.

Die beispiellosen Fähigkeiten des WFI-Instruments hinsichtlich hoher Zählraten bedeuten, dass während der WFI-Beobachtungen eine enorme Datenmenge erzeugt wird. Diese wertvollen Daten übersteigen bei Weitem die aktuellen Bandbreitenkapazitäten moderner Raumfahrzeugsysteme. Daher ist eine Datenfilterung und -vorverarbeitung an Bord erforderlich, um anschließend nur die wissenschaftlich relevanten Röntgendaten zur Erde herunterzuladen.

Zu diesem Zweck wurde ein Demonstrationsmodell des WFI-Frame-Prozessors entworfen, entwickelt und hergestellt. Es demonstriert erfolgreich die technologische Fähigkeit, die WFI-Daten unter Verwendung eines Pipeline-Konzepts in einem einzigen, für den Weltraum geeigneten FPGA-Baustein in Echtzeit an Bord zu verarbeiten. In dieser Arbeit wird der Fokus auf die Beschreibung solcher Pipelines gelegt. Darüber hinaus werden die Herausforderungen beim Entwurf skizziert und die Verifizierungs- und Validierungsergebnisse präsentiert, die während der Entwicklung des Demonstrationsmodells des Frame-Prozessors gewonnen wurden.

### III TABLE OF CONTENTS

|            |                                                                    |           |
|------------|--------------------------------------------------------------------|-----------|
| <b>I</b>   | <b>Abstract .....</b>                                              | <b>4</b>  |
| <b>II</b>  | <b>Zusammenfassung .....</b>                                       | <b>5</b>  |
| <b>III</b> | <b>Table of Contents .....</b>                                     | <b>6</b>  |
| <b>IV</b>  | <b>List of Figures .....</b>                                       | <b>8</b>  |
| <b>V</b>   | <b>List of Tables .....</b>                                        | <b>12</b> |
| <b>VI</b>  | <b>List of Acronyms .....</b>                                      | <b>13</b> |
| <b>1</b>   | <b>Introduction .....</b>                                          | <b>17</b> |
| 1.1        | <i>X-Ray Astronomy .....</i>                                       | <i>17</i> |
| 1.2        | <i>Brief History of X-ray Astronomy .....</i>                      | <i>18</i> |
| 1.2.1      | The Beginning of X-ray Astronomy: Sounding Rockets.....            | 18        |
| 1.2.2      | Early X-ray Satellites (Past Missions).....                        | 18        |
| 1.2.3      | X-ray Satellites (Active Missions) .....                           | 20        |
| <b>2</b>   | <b>The (New)Athena Mission .....</b>                               | <b>22</b> |
| 2.1        | <i>The Hot and Energetic Universe.....</i>                         | <i>22</i> |
| 2.2        | <i>The Athena Satellite.....</i>                                   | <i>24</i> |
| <b>3</b>   | <b>The WFI Instrument.....</b>                                     | <b>28</b> |
| 3.1        | <i>WFI Instrument Architecture .....</i>                           | <i>28</i> |
| 3.2        | <i>The Primary Structure Subsystem.....</i>                        | <i>29</i> |
| 3.3        | <i>The Camera Head Subsystem .....</i>                             | <i>30</i> |
| 3.3.1      | The DEPFET Sensors .....                                           | 30        |
| 3.3.2      | The Large Detector Array.....                                      | 32        |
| 3.3.3      | The Fast Detector .....                                            | 33        |
| 3.4        | <i>The Detector Electronics Subsystem .....</i>                    | <i>34</i> |
| 3.4.1      | The Galvanic Isolation Module.....                                 | 35        |
| 3.4.2      | The Power Conditioning Module .....                                | 35        |
| 3.4.3      | The Frame Processor Module .....                                   | 37        |
| 3.5        | <i>The Instrument Control Subsystem.....</i>                       | <i>42</i> |
| <b>4</b>   | <b>The Fast Detector Frame Processor .....</b>                     | <b>44</b> |
| 4.1        | <i>Design Philosophy.....</i>                                      | <i>44</i> |
| 4.2        | <i>The Readout Requirements of the FD Compared to the LDA.....</i> | <i>44</i> |
| 4.3        | <i>The FD Frame Processor Architecture.....</i>                    | <i>46</i> |
| 4.3.1      | Architecture Overview .....                                        | 47        |
| 4.3.2      | Timing Control and Clock Distribution .....                        | 47        |
| 4.3.3      | Event Preprocessing Pipeline .....                                 | 49        |
| 4.3.4      | Event List Generator .....                                         | 70        |
| 4.3.5      | Sequencer Unit.....                                                | 72        |
| 4.3.6      | Look-Up Table Generator .....                                      | 78        |
| 4.3.7      | Diagnostic Unit.....                                               | 82        |

|          |                                                                  |            |
|----------|------------------------------------------------------------------|------------|
| 4.3.8    | Housekeeping Unit .....                                          | 84         |
| 4.3.9    | Interface Controller .....                                       | 85         |
| 4.3.10   | High Speed Serial Interface .....                                | 85         |
| <b>5</b> | <b>The Large Detector Array Frame Processor .....</b>            | <b>89</b>  |
| 5.1      | <i>The Large Detector Array Redout Requirements .....</i>        | <i>89</i>  |
| 5.2      | <i>The LD Frame Processor Architecture.....</i>                  | <i>91</i>  |
| 5.3      | <i>Event Preprocessing Pipeline .....</i>                        | <i>92</i>  |
| 5.3.1    | Pattern Analyzer Unit .....                                      | 94         |
| 5.4      | <i>Event List Generator .....</i>                                | <i>97</i>  |
| 5.5      | <i>Memory Controller.....</i>                                    | <i>99</i>  |
| 5.6      | <i>Look-Up Table Generator and Look-Up Table Programmer.....</i> | <i>101</i> |
| 5.6.1    | OLT Calculation.....                                             | 102        |
| 5.6.2    | NLT Calculation.....                                             | 103        |
| <b>6</b> | <b>Performance and Tests .....</b>                               | <b>105</b> |
| 6.1      | <i>Implementation.....</i>                                       | <i>105</i> |
| 6.2      | <i>Testbench Simulation and Verification.....</i>                | <i>108</i> |
| 6.3      | <i>Hardware Bread-Boarding.....</i>                              | <i>110</i> |
| 6.3.1    | SmartFusion 2 Tests .....                                        | 110        |
| 6.3.2    | RTG4 Development Kit Tests .....                                 | 114        |
| 6.3.3    | Frame Processor Mother-Board Tests.....                          | 115        |
| 6.4      | <i>PRE-EM Hardware Integration Tests.....</i>                    | <i>120</i> |
| 6.4.1    | PRE-EM Firmware Integration Test.....                            | 122        |
| 6.4.2    | Camera Head PRE-EM Integration Test .....                        | 123        |
| <b>7</b> | <b>Outlook and Conclusion.....</b>                               | <b>125</b> |
| 7.1      | <i>PRE-EM Validation Tests .....</i>                             | <i>125</i> |
| 7.2      | <i>New Technologies .....</i>                                    | <i>126</i> |
| 7.3      | <i>Conclusion .....</i>                                          | <i>127</i> |
|          | <b>Appendix A: FD Interface Controller Operation .....</b>       | <b>129</b> |
|          | <i>Housekeeping Interface.....</i>                               | <i>130</i> |
|          | <i>Command Interface .....</i>                                   | <i>130</i> |
|          | <b>Appendix B: LD Interface Controller Operation .....</b>       | <b>142</b> |
|          | <i>Housekeeping Interface.....</i>                               | <i>143</i> |
|          | <i>Command Interface .....</i>                                   | <i>143</i> |
|          | <b>Bibliography.....</b>                                         | <b>157</b> |

## IV LIST OF FIGURES

|                                                                                                                                                                                                                                                                                                                     |    |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| FIGURE 1: (NEW)ATHENA SATELLITE FLYING (RENDERED IMAGE) [18].                                                                                                                                                                                                                                                       | 24 |
| FIGURE 2: CONCEPTUAL DESIGN OF THE MIRROR STRUCTURE (LEFT) SHOWING AN SPO MODULE (RIGHT) [20].                                                                                                                                                                                                                      | 25 |
| FIGURE 3: SCHEMATIC DIAGRAM OF A TES (LEFT), CHANGE IN RESISTANCE AS A FUNCTION OF TEMPERATURE IN THE TRANSITION REGION (RIGHT) [22].                                                                                                                                                                               | 26 |
| FIGURE 4: SCHEMATIC VIEW OF THE COMPLEX X-IFU CRYOGENIC ARCHITECTURE [23].                                                                                                                                                                                                                                          | 26 |
| FIGURE 5: CAD DESIGN OF THE WFI INSTRUMENT, THE TWO ACTIVE PIXEL SENSORS ARE LOCATED AT THE CENTER (BLUE), FIVE DETECTOR ELECTRONICS BOXES SURROUND THE INSTRUMENT (YELLOW) AND A FILTER WHEEL IS PLACED AT THE X-RAY WINDOW (BEIGE) [25].                                                                          | 27 |
| FIGURE 6: BLOCK DIAGRAM WITH THE GENERAL ARCHITECTURE OF THE WFI INSTRUMENT [27].                                                                                                                                                                                                                                   | 29 |
| FIGURE 7: CAD MODEL OF THE PRIMARY STRUCTURE SUBSYSTEM [29].                                                                                                                                                                                                                                                        | 30 |
| FIGURE 8: CAD MODEL OF THE WFI CAMERA HEAD SUBSYSTEM [31].                                                                                                                                                                                                                                                          | 30 |
| FIGURE 9: SCHEMATIC VIEW DEPICTING THE STRUCTURE OF A ROUND DEPFET SENSOR PIXEL [36].                                                                                                                                                                                                                               | 31 |
| FIGURE 10: SCHEMATIC SHOWING THE READOUT PROCEDURE OF THE DEPFET SENSORS, THE BASELINE (OBTAINED AFTER THE GATE CLEAR) IS SUBTRACTED FROM THE ORIGINAL MEASUREMENT (SIGNAL + BASELINE) TO OBTAIN THE PHOTON SIGNAL [35].                                                                                            | 32 |
| FIGURE 11: IMAGE OF A LARGE DETECTOR DEPFET OF 512 x 512 PIXELS, EIGHT VERITAS ASICs (BOTTOM LEFT) READOUT THE SIGNALS WHILE 8 SWITCHER ASICs (TOP) STEER THE SENSOR READOUT [34].                                                                                                                                  | 33 |
| FIGURE 12: ATHENA WFI FAST DETECTOR PROTOTYPE. TWO VERITAS FRONT-END ASICs CAN BE APPRECIATED ON OPPOSITE SIDES OF THE DETECTOR (BOTTOM LEFT AND TOP RIGHT SIDE OF THE DETECTOR), WHILE TWO SWITCHER CONTROL ASICs STEER THE READOUT (TOP LEFT DETECTOR SIDE) [37].                                                 | 34 |
| FIGURE 13: BREADBOARD MODEL OF THE GALVANIC ISOLATION MODULE [27].                                                                                                                                                                                                                                                  | 35 |
| FIGURE 14: ENGINEERING MODEL OF THE POWER CONDITIONING MODULE. TO REDUCE DEVELOPMENT COSTS, A MIX OF SPACE QUALIFIED AND COMMERCIAL EQUIVALENT COMPONENTS ARE USED.                                                                                                                                                 | 36 |
| FIGURE 15: PRE-ENGINEERING MODEL OF THE FPM. A SPACE QUALIFIED RTG4 FPGA IS IN THE TOP-CENTER. EIGHT ADCs ARE USED TO PERFORM DATA DIGITIZATION, TO SAVE RESOURCES, ONLY ONE SPACE QUALIFIED COMPONENT WAS USED, AND THE REST USE COMMERCIAL COMPONENTS. THE SAME APPLIES FOR THE LVDS BUFFER (BOTTOM CORNER) [40]. | 38 |
| FIGURE 16: SIMPLIFIED SCHEMATIC SHOWING THE INTERNAL STRUCTURE OF FPGA'S LOGIC ELEMENTS. A LOOK-UP TABLE OUTPUT IS CONNECTED TO A CLOCKING DEVICE TO KEEP SYNCHRONIZATION. THE CONTENT OF THE LUT CAN BE CONFIGURED AS THE USER REQUIRES, AN EXAMPLE IS SHOWN (LEFT).                                               | 39 |
| FIGURE 17: A SIMPLIFIED SCHEMATIC ILLUSTRATING THE STRUCTURE OF AN FPGA. LEs COMMUNICATE WITH ONE ANOTHER THROUGH A NETWORK OF CONNECTORS AND SWITCHES, AND TO THE OUTSIDE THROUGH SPECIALIZED IOS.                                                                                                                 | 40 |
| FIGURE 18: SIMPLIFIED FUNCTIONAL BLOCK DIAGRAM OF AN RTG4 LE [41].                                                                                                                                                                                                                                                  | 41 |
| FIGURE 19: RTG4 INTERNAL LOGIC RESOURCES LAYOUT OVERVIEW WITH LABELS POINTING TO THE ON-CHIP LOCATION OF SEVERAL SPECIALIZED RESOURCES [41].                                                                                                                                                                        | 41 |
| FIGURE 20: THE GR740 PROCESSOR, A HIGH-PERFORMANCE PROCESSOR THAT ENABLES DATA COMPRESSION FOR THE WFI INSTRUMENT [44].                                                                                                                                                                                             | 43 |
| FIGURE 21: SCHEMATIC SHOWING THE LINE OF PIXELS READOUT DURING A PIXEL-ROW TIME BY 2 ASICs ON THE FD (LEFT) AND 8 ASICs ON THE LDA (RIGHT).                                                                                                                                                                         | 45 |
| FIGURE 22: GENERAL ARCHITECTURE BLOCK DIAGRAM FOR THE FD-FP FIRMWARE.                                                                                                                                                                                                                                               | 47 |
| FIGURE 23: TIME DIAGRAM ILLUSTRATING THE PIXEL CLOCK, PIXEL-TIME AND PIXEL-ROW TIME.                                                                                                                                                                                                                                | 48 |
| FIGURE 24: TIME DIAGRAM ILLUSTRATING THE PIXEL-TIME WITH RESPECT TO THE SYSTEM CLOCK CYCLES. DATA IS ALIGNED FOR 3 SYSTEM CLOCKS PER PIXEL-CLOCK (ABOVE) AND 6 SYSTEM CLOCKS PER PIXEL CLOCK (BELOW).                                                                                                               | 49 |
| FIGURE 25: TIME DIAGRAM ILLUSTRATING A FRAME CLOCK AND ITS ALIGNMENT WITH RESPECT TO THE FRAME DATA. IT SHOULD BE NOTED THAT A FRAME IS COMPOSED OF 2048 PIXELS.                                                                                                                                                    | 49 |
| FIGURE 26: STRUCTURE OF THE FP PIPELINES, IT SHOULD BE NOTED THAT APART FROM THE MBU, THE NUMBER OF UNITS IS THE SAME AS FOR THE LDA-FP.                                                                                                                                                                            | 50 |

|                                                                                                                                                                                                                                                 |    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| FIGURE 27: PIXEL ENERGY FRACTION SPREADING TO SUBSEQUENT PIXELS AS A FUNCTION OF THE MULTIPLEXING CLOCK (PIXEL-CLOCK) LENGTH. DATA IS ALIGNED TO THE FALLING EDGE FOR PROPER COMPARISON [45].                                                   | 51 |
| FIGURE 28: PIXEL ENERGY TAIL SPREADING TO SUBSEQUENT PIXELS AS A FUNCTION OF THE CABLE LENGTH [45].                                                                                                                                             | 52 |
| FIGURE 29: SIMPLIFIED BLOCK DIAGRAM OF THE CTU.                                                                                                                                                                                                 | 53 |
| FIGURE 30: INTERNAL STRUCTURE OF THE CTU INPUT AND OUTPUT BITS.                                                                                                                                                                                 | 54 |
| FIGURE 31: SCHEMATIC SHOWING THE MEMORY SIZE AND WORD DISTRIBUTION OF THE MBU MEMORIES.                                                                                                                                                         | 55 |
| FIGURE 32: BLOCK DIAGRAM DEPICTING THE MULTIPLEXING STRUCTURE CONNECTING THE OUTPUT OF THE SENSORS TO THE THREE BUFFERING MEMORIES.                                                                                                             | 56 |
| FIGURE 33: TIMELINE SCHEMATIC SHOWING HOW CONSECUTIVE FRAMES ARE READOUT IN THE TWO PIPELINES.                                                                                                                                                  | 57 |
| FIGURE 34: SIMPLIFIED BLOCK DIAGRAM OF THE PCU.                                                                                                                                                                                                 | 58 |
| FIGURE 35: INTERNAL STRUCTURE OF THE PCU INPUT AND OUTPUT BITS.                                                                                                                                                                                 | 59 |
| FIGURE 36: SIMPLIFIED BLOCK DIAGRAM OF THE CMU.                                                                                                                                                                                                 | 60 |
| FIGURE 37: SCHEMATIC DEPICTING THE SORTING MECHANISM OF THE CMU. IT SHOULD BE NOTED THAN ON EVEN CLOCK NUMBERS THE COMPARISON IN OFFSET, SKIPPING ADDRESS 0 AND THUS ALLOWING PIXELS TO KEEP DRIFTING ON EACH COMPARISON UNTIL THEY ARE SORTED. | 61 |
| FIGURE 38: INTERNAL STRUCTURE OF THE CMU INPUT AND OUTPUT BITS.                                                                                                                                                                                 | 62 |
| FIGURE 39: SIMPLIFIED BLOCK DIAGRAM OF THE EFU.                                                                                                                                                                                                 | 63 |
| FIGURE 40: INTERNAL STRUCTURE OF THE EFU INPUT AND OUTPUT BITS.                                                                                                                                                                                 | 63 |
| FIGURE 41: SIMPLIFIED BLOCK DIAGRAM OF THE PAU.                                                                                                                                                                                                 | 64 |
| FIGURE 42: PIXEL PATTERNS CORRESPONDING TO VALID X-RAY EVENTS ALLOWED BY THE PAU.                                                                                                                                                               | 64 |
| FIGURE 43: SCHEMATIC DEPICTING THE CLOCKING AND MOVING OF PIXELS INSIDE THE PAU ARRAYS. DATA ENTERS ON EACH PIXEL CLOCK, ONCE THE FIRST ARRAY IS FILLED, THE FULL ARRAYS GET COPIED TO THE FOLLOWING ARRAY SYNCHRONIZED WITH THE PAU CLK.       | 65 |
| FIGURE 44: COMPARISON MASK USED BY THE PATTERN ANALYSIS UNIT TO FLAG BAD PATTERNS, THE MASKS TRANSLATES VERTICALLY OVER FA4 PERFORMING THE COMPARISONS ON EACH PIXEL.                                                                           | 66 |
| FIGURE 45: SCHEMATIC SHOWING THE PIXEL PROMOTION MASK RADIUS.                                                                                                                                                                                   | 67 |
| FIGURE 46: SCHEMATIC SHOWING THE MIP NEIGHBOR MASK RADIUS.                                                                                                                                                                                      | 68 |
| FIGURE 47: INTERNAL STRUCTURE OF THE EFU INPUT AND OUTPUT BITS.                                                                                                                                                                                 | 68 |
| FIGURE 48: INTERNAL STRUCTURE OF THE PFU MASK AND PFU BYPASS REGISTERS BITS.                                                                                                                                                                    | 69 |
| FIGURE 49: SIMPLIFIED BLOCK DIAGRAM OF THE PFU.                                                                                                                                                                                                 | 70 |
| FIGURE 50: INTERNAL STRUCTURE OF THE PFU INPUT AND OUTPUT BITS.                                                                                                                                                                                 | 70 |
| FIGURE 51: SIMPLIFIED BLOCK DIAGRAM OF THE ELG.                                                                                                                                                                                                 | 71 |
| FIGURE 52: TIMELINE DEPICTING THE ORDER IN WHICH DATA IS FETCHED BY THE READOUT SCHEDULER FSM.                                                                                                                                                  | 72 |
| FIGURE 53: DATA FORMAT OF THE EVENT LISTS AND THE INTERNAL BIT STRUCTURE OF EACH WORD.                                                                                                                                                          | 72 |
| FIGURE 54: SIMPLIFIED BLOCK DIAGRAM OF THE SEQUENCER UNIT.                                                                                                                                                                                      | 73 |
| FIGURE 55: SIMPLIFIED SCHEMATIC OF THE MEMORY BASE SEQUENCER, GENERATING THE FIRST 27 CLOCK CYCLES OF A 4-BIT WIDTH BUS.                                                                                                                        | 74 |
| FIGURE 56: SIMPLIFIED BLOCK DIAGRAMS WITH THE SEQUENCER MEMORIES INTERNAL DIVISIONS.                                                                                                                                                            | 75 |
| FIGURE 57: SIMPLIFIED BLOCK DIAGRAM OF THE VERITAS CONTROL MODULE.                                                                                                                                                                              | 77 |
| FIGURE 58: WORD ADDRESSES ORGANIZATION INSIDE THE VERITAS MEMORY. THE FAST DETECTOR IMPLEMENTATION ONLY USES THE A AND B BLOCKS OF THE CYCLIC CHECK MEMORY ALLOCATION.                                                                          | 78 |
| FIGURE 59: SIMPLIFIED BLOCK DIAGRAM OF THE LTG ARCHITECTURE.                                                                                                                                                                                    | 80 |
| FIGURE 60: GOOD FRAME (LEFT) AND BAD FRAME (RIGHT) DEFAULT TEST EXAMPLES.                                                                                                                                                                       | 83 |
| FIGURE 61: SIMPLIFIED BLOCK DIAGRAM OF THE DIAGNOSTIC UNIT.                                                                                                                                                                                     | 83 |
| FIGURE 62: SIMPLIFIED BLOCK DIAGRAM OF THE CURRENT "DUMMY" HK INTERFACE.                                                                                                                                                                        | 84 |
| FIGURE 63: SIMPLIFIED BLOCK DIAGRAM OF THE INTERFACE CONTROLLER.                                                                                                                                                                                | 85 |
| FIGURE 64: SIMPLIFIED BLOCK DIAGRAM OF THE HSSI.                                                                                                                                                                                                | 86 |
| FIGURE 65: STRUCTURE OF THE HSSI PACKETS.                                                                                                                                                                                                       | 88 |
| FIGURE 66: SIZE COMPARISON BETWEEN THE FD SENSOR AND A QUADRANT OF THE LDA. THE LDA QUADRANT IS 64 TIMES LARGER THAN THE FD.                                                                                                                    | 90 |

|                                                                                                                                                                                                                                                                                                                             |     |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| FIGURE 67: BLOCK DIAGRAM SHOWING THE DATA PROCESSING FLOW IN THE LD-FP. DATA IS READ OUT AND CONVERTED IN PARALLEL COMING FROM THE EIGHT VERITAS ASICs. IN A MIRRORED MANNER, DATA IS PREPROCESSED IN EIGHT PARALLEL PIPELINES. AFTERWARDS, DATA IS MERGED INTO AN EVENT LIST AND SENT TO THE ICS VIA A SpW INTERFACE. .... | 91  |
| FIGURE 68: GENERAL ARCHITECTURE BLOCK DIAGRAM OF THE LD-FP FIRMWARE. ....                                                                                                                                                                                                                                                   | 92  |
| FIGURE 69: SIMPLIFIED BLOCK DIAGRAM SHOWING THE EPPs ARCHITECTURE. ....                                                                                                                                                                                                                                                     | 93  |
| FIGURE 70: EXAMPLES OF THE PATTERNS GENERATED BY X-RAY EVENTS. THE PAU ALLOWS TO IDENTIFY NEIGHBOR PIXELS AND PROMOTES THEM TO EVENT PIXELS TO PREVENT DATA LOST. ....                                                                                                                                                      | 95  |
| FIGURE 71: SCHEMATIC SHOWING THE DISTRIBUTION OF THE SENSOR SECTIONS (LEFT) AND THEIR CORRESPONDING PROCESSING UNITS INSIDE THE PAU (RIGHT). ....                                                                                                                                                                           | 96  |
| FIGURE 72: SIMPLIFIED BLOCK DIAGRAM SHOWING THE INTERNAL STRUCTURE OF THE PAU, TO AID READABILITY ONLY ONE SPA, ONE PDB AND ONE BPA MODULE ARE SHOWN. ....                                                                                                                                                                  | 97  |
| FIGURE 73: SIMPLIFIED BLOCK DIAGRAM OF THE ELG. ....                                                                                                                                                                                                                                                                        | 98  |
| FIGURE 74: SIMPLIFIED BLOCK DIAGRAM OF THE MEMORY CONTROLLER INTERFACE WITH THE EPP. ....                                                                                                                                                                                                                                   | 100 |
| FIGURE 75: SIMPLIFIED BLOCK DIAGRAM SHOWING THE STRUCTURES OF THE EPP, LTG, LTP AND MEMORY CONTROLLER. ALL INTERFACING TOGETHER TO FACILITATE THE OLG AND NLG TASKS INSIDE THE FPGA. ....                                                                                                                                   | 102 |
| FIGURE 76: SCREENSHOT SHOWING A TYPICAL VHDL SIMULATION EXAMPLE. IN THE IMAGE, THE SPA UNIT SIGNALS (ONE SIGNAL PER ROW) ARE SHOWN EVOLVING OVER TIME, THE SIMULATION TOOL ALLOWS TO OBSERVE SEVERAL SIGNALS IN PARALLEL TO VERIFY THE CONCURRENT OPERATION OF THE DIFFERENT LOGIC UNITS. ....                              | 108 |
| FIGURE 77: TESTBENCH CANVAS FOR THE LDA-FP. THE FULL LDA-FP DESIGN (D3C_IO_FRAME) IS CONNECTED TO A STIMULUS UNIT WHICH GENERATES SpW COMMANDS, DATA PATTERNS AND THE PIXEL CLOCKS. TO EMULATE THE EXTERNAL SSRAM MEMORY THE SSRAM_CONTROLLER_TB WAS ADDED CREATING A COMPLETE VIRTUAL ENVIRONMENT. ....                    | 109 |
| FIGURE 78: PICTURES OF THE TWO INITIALLY DEVELOPED FMC BOARDS: ON THE LEFT THE MEMORY TESTING BOARD, ON THE RIGHT THE GPIO BOARD. ....                                                                                                                                                                                      | 111 |
| FIGURE 79: PICTURE OF THE SMARTFUSION 2 DEVELOPMENT KIT WITH THE SpW-MEMORY BOARD CONNECTED ON ONE FMC PORT (RIGHT) AND THE GPIO BOARD CONNECTED ON THE OTHER PORT (TOP). ....                                                                                                                                              | 112 |
| FIGURE 80: TESTING SETUP WITH TWO SF2 DEVELOPMENT KITS, ONE SF2 CONTAINS A SIMPLE DIAGNOSTIC SYSTEM (LEFT) AND SENDS DATA TO THE FP BOARD TO BE PROCESSED AND OUTPUT THE PROCESSED DATA VIA SpW (RIGHT). ....                                                                                                               | 113 |
| FIGURE 81: THE SMARTFUSION 2 DEVELOPMENT KIT WITH THE SpW-MEMORY BOARD (RIGHT) AND THE GIGABIT-ETHERNET FMC BOARD USED FOR THE HSSI (TOP). ....                                                                                                                                                                             | 113 |
| FIGURE 82: EARLY VERSION OF THE HADES COMMAND LINE INTERFACE (LEFT), THE QUICK FRAME VISUALIZER SHOWING DIAGNOSTIC DATA WITH THE ATHENA LOGO. ....                                                                                                                                                                          | 115 |
| FIGURE 83: THE RTG4 DEVELOPMENT KIT BOARD WITH THE GIGABIT-ETHERNET FMC (TOP) AND SpW-MEMORY FMC (RIGHT) CONNECTED. THE SpW BRICK (TOP LEFT) RECEIVES THE PREPROCESSED DATA. ....                                                                                                                                           | 115 |
| FIGURE 84: REAL-TIME PERFORMANCE TESTING SETUP OPERATING AT IAAT. THE FP-MB RECEIVES DATA FROM THE VIRTEX 7 DEVELOPMENT BOARD (TOP) THROUGH A FLEX-RIGID PCB CONNECTOR. OUTPUT DATA IS RECEIVED THROUGH THE SpW INTERFACE FOR ITS ANALYSIS. ....                                                                            | 116 |
| FIGURE 85: BLOCK DIAGRAM SHOWING THE TEST SETUP USED FOR THE REAL-TIME PERFORMANCE VERIFICATION TESTS OF THE FP (LEFT). SAMPLE 3D VISUALIZATION OF THE FRAME DATA GENERATED BY THE MATLAB SOFTWARE (RIGHT). ....                                                                                                            | 117 |
| FIGURE 86: COMPARISON BETWEEN THE FP OUTPUT AND EXPECTED OUTPUT DATA, SHOWING COMPLETE AGREEMENT [39]. ....                                                                                                                                                                                                                 | 118 |
| FIGURE 87: TESTING PROCEDURE FOLLOWED FOR THE PERFORMANCE VERIFICATION TEST (LEFT). SCHEMATIC DEPICTING THE TESTING SETUP AND DATA FLOW (RIGHT) [23]. ....                                                                                                                                                                  | 119 |
| FIGURE 88: THE FP-MB WITH THE ASICs FMC (TOP LEFT). TEST SETUP FOR THE ASICs INTERFACE; A LOGIC ANALYZER ALLOWS THE VISUALIZATION OF THE PROGRAMMING AND SEQUENCING SIGNALS. A WAVEFORM GENERATOR (BLACK) EMULATES THE ANALOG SIGNALS COMING FROM THE CH. ....                                                              | 120 |
| FIGURE 89: STEP FROM THE FP-MB (LEFT) TO THE PRE-EM (RIGHT) [40]. ....                                                                                                                                                                                                                                                      | 121 |
| FIGURE 90: SCHEMATIC BLOCK DIAGRAM SHOWING THE ARCHITECTURE OF THE PRE-EM WITH THE RTG4 AS A CENTRAL ELEMENT (LEFT). PICTURE OF THE FPM PRE-EM SHOWING ITS INTERFACES (RIGHT) [15]. ....                                                                                                                                    | 121 |

|                                                                                                                                                                                                                                                                                                                                                                                                                         |     |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| FIGURE 91: TEST SETUP FOR THE FIRST FIRMWARE INTEGRATION OF THE PRE-EM CONDUCTED AT IAAT. THE PRE-EM HARDWARE (CENTER) IS CONNECTED TO THE LABORATORY POWER SUPPLY AND A WAVEFORM GENERATOR GENERATING THE CLOCKING SIGNALS DRIVING THE HARDWARE. A LOGIC ANALYZER WAS USED TO CHECK THE CHS SIGNALS WHILE THE TESTING COMPUTER ENABLED DATA CAPTURE THROUGH THE SPW BRICK AND GIGABIT-ETHERNET PORT FOR THE HSSI. .... | 122 |
| FIGURE 92: LDA TESTING SETUP (LEFT) AND FD TESTING SETUP (RIGHT) INSIDE THE LAMINAR FLOW CHAMBER. THE LDA AND FD CAMERA HEAD PROTOTYPES CAN BE APPRECIATED ON THE TOP. ....                                                                                                                                                                                                                                             | 123 |
| FIGURE 93: WINDOW SHOWING AN EXAMPLE OF A PATCH BEING MODIFIED. THE ORIGINAL LINE IS SHOWN ON THE LEFT, AND THE MODIFIED ONE IS ON THE RIGHT. ON THE BOTTOM, THE PATCH VISUALIZER SHOWS THE MODIFIED PATCH SIX. .                                                                                                                                                                                                       | 124 |
| FIGURE 94: TESTING SETUP WITH THE LABORATORY LOGIC ANALYZER RECORDING THE CAMERA HEAD SIGNALS (LEFT). AN OSCILLOSCOPE RECORDED THE SUCCESSFUL ACQUISITION OF ANALOG SIGNALS PULSES ON THE SPECIFIED CHANNELS COMING FROM THE CH (RIGHT).....                                                                                                                                                                            | 124 |
| FIGURE 95: MOUNTING OF THE CAMERA HEAD AND DETECTOR ELECTRONICS SUBSYSTEMS INSIDE THE PUMA FACILITY VACUUM CHAMBER [26]. ....                                                                                                                                                                                                                                                                                           | 125 |
| FIGURE 96: RELATION OF THE CORRECTION FACTOR COEFFICIENTS AND CORRECTION PERCENTAGES .....                                                                                                                                                                                                                                                                                                                              | 133 |

## V LIST OF TABLES

|                                                                                                  |     |
|--------------------------------------------------------------------------------------------------|-----|
| TABLE 1: SUMMARY ON THE READOUT SPECIFICATIONS OF THE LDA AND FD. ....                           | 46  |
| TABLE 2: EXAMPLES OF CORRECTION FACTORS IN PERCENTAGE DEPENDENT ON THE R1 AND R2 REGISTERS. .... | 53  |
| TABLE 3: LINE MEMORY INTERNAL ADDRESS MAPPING. ....                                              | 76  |
| TABLE 4: LDA-FP RESOURCE UTILIZATION REPORT SUMMARY. ....                                        | 107 |
| TABLE 5: FD-FP RESOURCE UTILIZATION REPORT SUMMARY. ....                                         | 107 |
| TABLE 6: 7-BYTE COMMAND PACKET STRUCTURE CONTENTS. ....                                          | 129 |
| TABLE 7: INTERFACE ADDRESS MAP. ....                                                             | 129 |
| TABLE 8: SEND/RECEIVE POSSIBLE VALUES. ....                                                      | 130 |
| TABLE 9: REGISTER LIST OF THE HOUSEKEEPING INTERFACE. ....                                       | 130 |
| TABLE 10: REGISTERS LIST OF THE COMMAND INTERFACE FD EPP REGISTERS. ....                         | 131 |
| TABLE 11: MEMORY SELECTION ADDRESS MAP. ....                                                     | 132 |
| TABLE 12: ADDRESS RANGE AVAILABLE TO EACH MEMORY. ....                                           | 132 |
| TABLE 13: EPP COMMAND LIST. ....                                                                 | 134 |
| TABLE 14: REGISTER LIST OF THE COMMAND INTERFACE COMMUNICATION AND DEBUG REGISTERS. ....         | 135 |
| TABLE 15: REGISTER LIST OF THE COMMAND INTERFACE SEQUENCER REGISTERS. ....                       | 137 |
| TABLE 16: INTERNAL STRUCTURE OF THE SPI INTERFACE MEMORY. ....                                   | 138 |
| TABLE 17: VERITAS ASIC CHECK SIZE REGISTER ADDRESSES. ....                                       | 139 |
| TABLE 18: LIST OF SEQUENCER COMMANDS. ....                                                       | 140 |
| TABLE 19: 7-BYTE COMMAND PACKET STRUCTURE CONTENTS. ....                                         | 142 |
| TABLE 20: INTERFACE ADDRESS MAP. ....                                                            | 142 |
| TABLE 21: SEND/RECEIVE POSSIBLE VALUES. ....                                                     | 143 |
| TABLE 22: REGISTER LIST OF THE HOUSEKEEPING INTERFACE. ....                                      | 143 |
| TABLE 23: REGISTERS LIST OF THE COMMAND INTERFACE LD EPP REGISTERS. ....                         | 144 |
| TABLE 24: MEMORY SELECTION ADDRESS MAP. ....                                                     | 145 |
| TABLE 25: ADDRESS RANGE AVAILABLE TO EACH MEMORY. ....                                           | 145 |
| TABLE 26: SRAM FIFO 3 REGISTERS CORRESPONDENCE TO SSRAM INTERFACE. ....                          | 146 |
| TABLE 27: SLICE SELECT REGISTER TRUTH TABLE. ....                                                | 147 |
| TABLE 28: EPP COMMAND LIST. ....                                                                 | 148 |
| TABLE 29: FP DEBUG REGISTERS. ....                                                               | 149 |
| TABLE 30: SLICE SELECT REGISTER VALUES DECODING. ....                                            | 151 |
| TABLE 31: REGISTER LIST OF THE COMMAND INTERFACE SEQUENCER REGISTERS. ....                       | 152 |
| TABLE 32: MAPPING OF COMMAND ADDRESSES TO THE VERITAS MEMORY OF THE FP. ....                     | 152 |
| TABLE 33: VERITAS ASIC CHECK SIZE REGISTER ADDRESSES. ....                                       | 154 |
| TABLE 34: LIST OF SEQUENCER COMMANDS. ....                                                       | 155 |



## VI LIST OF ACRONYMS

|        |                                                  |
|--------|--------------------------------------------------|
| ADR    | Adiabatic Demagnetization Refrigerator           |
| ADU    | Analog Digital Units                             |
| AIU    | Analog Interface Unit                            |
| AN     | Auto-Negotiation                                 |
| AS&E   | American Science and Engineering company         |
| ASCA   | Advance Satellite for Cosmology and Astrophysics |
| ASIC   | Application Specific Integrated Circuit          |
| Athena | Advanced Telescope for High-ENergy Astrophysics  |
| BPA    | Border Pattern Analyzer                          |
| BSM    | Bi-level Switch Monitor                          |
| CAD    | Computer-Aided Design                            |
| CCD    | Charge Coupled Device                            |
| CH     | Camera Head                                      |
| CHS    | Camera Head Subsystem                            |
| CMD    | Command (Interface)                              |
| CMU    | Common Mode Unit                                 |
| CTU    | Crosstalk Unit                                   |
| DAC    | Digital-to-Analog Converters                     |
| DC/DC  | Direct Current / Direct Current                  |
| DDR    | Double Data Rate                                 |
| DEPFET | Depleted P-Channel Field Effect Transistor       |
| DES    | Detector Electronics Subsystem                   |
| DF     | Data FIFO                                        |
| DIAG   | Diagnostic Unit                                  |
| DNEL   | Disconnection of Non-Essential Loads             |
| DP     | Dead Pixel                                       |
| DPU    | Digital Processing Unit                          |
| EDA    | Electronic Design Automation                     |
| EDAC   | Error Detection And Correction                   |
| EFU    | Event Filtering Unit                             |
| EGSE   | Electronic Ground Support Equipment              |
| ELG    | Event List Generator                             |
| EPIC   | European Photon Imaging Camera                   |
| EPP    | Event Preprocessing Pipeline                     |
| ESA    | European Space Agency                            |
| EUV    | Extreme Ultraviolet                              |
| FA     | Flag Array                                       |
| FD     | Fast Detector                                    |
| FD-FP  | Fast Detector Frame Processor                    |
| FDIR   | Fault Detection, Isolation, and Recovery         |
| FEE    | Front-End-Electronics                            |

|        |                                                       |
|--------|-------------------------------------------------------|
| FF     | Flip-Flop                                             |
| FIFO   | First-In First-Out                                    |
| FMC    | FPGA Mezzanine Connector                              |
| FoV    | Field of View                                         |
| FP     | Frame Processor                                       |
| FP-MB  | Frame Processor Mother Board                          |
| FPGA   | Field Programmable Gate Array                         |
| FPM    | Frame Processor Module                                |
| FSM    | Finite State Machine                                  |
| FW     | Filter Wheel                                          |
| GIM    | Galvanic Isolation Module                             |
| GMII   | Gigabit Media Independent Interface                   |
| GPIO   | General Purpose Input output                          |
| GPU    | Graphics Processing Unit                              |
| HETG   | High-Energy Transmission Grating                      |
| HK     | Housekeeping                                          |
| HKU    | Housekeeping Unit                                     |
| HP     | Hot Pixel                                             |
| HRC    | High-Resolution Camera                                |
| HRI    | High-Resolution Imager                                |
| HSSI   | High-Speed Serial Interface                           |
| HT     | High Threshold                                        |
| I/O    | Input / Output                                        |
| IAAT   | Institute for Astronomy and Astrophysics Tübingen     |
| IC     | Integrated Circuit                                    |
| ICS    | Instrument Control Subsystem                          |
| IEEE   | Institute of Electrical and Electronics Engineers     |
| IFC    | Interface Controller                                  |
| INAF   | Istituto Nazionale di Astrofisica                     |
| INT    | Interface Module                                      |
| IOB    | Input-Output Bank                                     |
| IP-UDP | Internet Protocol + User Datagram Protocol            |
| IRAP   | Insitut de Recherche en Astrophysique et Planetologie |
| JTAG   | Join Test Action Group                                |
| LCL    | Latch-Current-Limiter                                 |
| LD     | Large Detector                                        |
| LD-FP  | Large Detector Frame Processor                        |
| LDA    | Large Detector Array                                  |
| LDA-FP | Large Detector Array Frame Processor                  |
| LE     | Logic Element                                         |
| LED    | Light Emitting Diode                                  |
| LETG   | Low-Energy Transmission Grating                       |
| LPC    | Low-Power-Command                                     |

|         |                                                  |
|---------|--------------------------------------------------|
| LSRAM   | Large Static Random-Access Memory                |
| LT      | Low Threshold                                    |
| LTG     | Look-Up Table Generation (Unit)                  |
| LTP     | Look-Up Table Programmer                         |
| LUT     | Look-Up Table                                    |
| LVDS    | Low-Voltage Differential Signaling               |
| MAC     | Media Access Control and Reconciliation          |
| MACC    | Multiply Accumulate                              |
| MBS     | Memory-Based Sequencer                           |
| MBU     | Memory Buffering Unit                            |
| MIP     | Minimum Ionizing Particle                        |
| MN      | MIP Neighbor                                     |
| MOS     | Metal Oxide Semiconductor                        |
| MOSFET  | MOS Field Effect Transistor                      |
| MPE     | Max Plank Institut für Extraterrestrische Physik |
| NASA    | National Aeronautics Space Administration        |
| NLT     | Noise Look-up Table                              |
| OLT     | Offset Look-up Table                             |
| OSI     | Open Systems Interconnection                     |
| PAU     | Pattern Analysis Unit                            |
| PCB     | Printed Circuit Board                            |
| PCM     | Power Conditioning Module                        |
| PCS     | Physical Coding Sublayer                         |
| PCU     | Pixel Correction Unit                            |
| PDB     | Pattern Data Buffer                              |
| PDB     | Pixel Data Buffer                                |
| PDB     | Pattern Data Bypass                              |
| PFU     | Pixel Filtering Unit                             |
| PHY     | Physical Layer                                   |
| PMA     | Physical Medium Attachment                       |
| PP      | Promotion Pixel                                  |
| PPS     | Pulse Per Second                                 |
| PRE-EM  | Preliminary Engineering Model                    |
| PSPC    | Position Sensitive Proportional Counter          |
| PSS     | Primary Structure Subsystem                      |
| PSU     | Power Supply Unit                                |
| PT      | Bad Pattern (Flag)                               |
| PUMA    | Pixel Detector Under MPE Analysis (facility)     |
| QML     | Qualified Manufacturers List (space components)  |
| RadHard | Radiation Hardened                               |
| RAM     | Random-Access Memory                             |
| RGS     | Reflection Grating Spectrometer                  |
| ROSAT   | Röntgen Satellite                                |

|         |                                                                  |
|---------|------------------------------------------------------------------|
| RS      | Reconciliation Sublayer                                          |
| RTG4    | Radiation Tolerant Generation 4 (Microchip FPGA)                 |
| S/C     | Spacecraft                                                       |
| SEL     | Single Event Latch-Up                                            |
| SEQ     | Sequencer (unit)                                                 |
| SerDes  | Serializer-Deserializer                                          |
| SEU     | Single Event Upset                                               |
| SGMII   | Serial Gigabit Media independent Interface                       |
| SIM     | Science Instrument Module                                        |
| SNR     | Supernova Remnant                                                |
| SPA     | Slice Pattern Analyzer                                           |
| SPI     | Serial Peripheral Interface                                      |
| SPO     | Silicon Pore Optics                                              |
| SpW     | SpaceWire                                                        |
| SRAM    | Static Random-Access Memory                                      |
| SRON    | Netherlands Institute for Space Research                         |
| SSRAM   | Synchronous Static Random-Access Memory                          |
| TBI     | Ten Bit Interface                                                |
| TES     | Transition-Edge Sensor                                           |
| TID     | Total Ionizing Dose                                              |
| TMR     | Triple Module Redundancy                                         |
| UART    | Universal Asynchronous Receive Transmit                          |
| UV      | Ultraviolet                                                      |
| VD      | Valid pixel (flag)                                               |
| Veritas | Versatile Readout based on Integrated Trapezoidal Analog Shapers |
| VHDL    | VHSIC Hardware Description Language                              |
| WFC     | Wide Field Camera                                                |
| WFI     | Wide Field Imager                                                |
| X-IFU   | X-ray Integral Field Unit                                        |
| XMM     | X-ray Multi-Mirror Mission                                       |
| XRT     | X-ray Telescope                                                  |

## 1 INTRODUCTION

This thesis details the work undertaken in developing instrumentation for the Athena X-ray space observatory, which is designed to facilitate advanced research in X-ray astronomy. Given the scale of this endeavor, it is essential to justify the significant investment of resources and effort by emphasizing the scientific objectives that drive research in this field.

Therefore, this introductory chapter opens by presenting the scientific foundation of X-ray astronomy, emphasizing how observations in this energy range unveil the physics of the universe's most intense and extreme phenomena. These are produced by objects such as black holes, neutron stars, supernova remnants, and the hot plasma in galaxy clusters. The chapter then briefly outlines the early milestones in cosmic X-ray detection, beginning with sounding rocket and balloon experiments, progressing to the advent of non-imaging detectors.

A concise overview of pioneering satellite missions such as Uhuru, the Einstein Observatory and ROSAT follows, each of which contributed key advancements in survey capabilities and imaging techniques. Lastly, attention turns to currently operating space-based observatories, Chandra and XMM Newton, highlighting their technological innovations in imaging resolution, spectral analysis, and hard X-ray focusing.

### 1.1 X-RAY ASTRONOMY

High energy astrophysics is a branch of study which mainly through the observation of X-rays, gamma rays and cosmic rays, focuses on studying the most energetic processes and regions the universe, where temperatures soar several million kelvin and matter is subject to extreme gravity, pressures and magnetic fields. The understanding of this physics is fundamental to the overall advancements on astrophysics and fundamentally our understanding of physics.

Objects studied in high-energy astrophysics encompass astrophysical sources such as accreting compact objects (black holes and neutron stars), supernova remnants, relativistic particle accelerators and explosive events (supernovae or gamma-ray bursts). These highly energetic processes are particularly bright in the X-ray waveband. X-ray astronomy focuses on exploring and studying the universe at these wavelengths, revealing physical processes which are, many times, invisible in optical or radio bands.

Modern X-ray instrumentation is capable of precisely acquiring energy, arrival time, direction and even polarization of X-ray photons on a photon-by-photon basis. Therefore, X-ray data can provide direct measurements of temperature, density, and chemical composition of hot plasmas. They also capture rapid variability, from

milliseconds in neutron star bursts to hours in active galactic nuclei or even days and months in Seyfert 1 galaxies [1], which is essential to understand accretion and jet physics. Among the key celestial objects studies in X-ray are included:

- Accreting black holes and neutron stars,
- X-ray binaries,
- Supernova remnants, and
- Hot gas in galaxy clusters [2].

## 1.2 BRIEF HISTORY OF X-RAY ASTRONOMY

In contrast to visible light, X-ray emissions from celestial bodies do not penetrate the Earth's atmosphere; even at the lowest X-ray energies (0.5–5 keV), minimal atmospheric presence is sufficient to prevent their arrival at the planet's surface [3]. While this atmospheric shielding is vital for sustaining life on Earth, it presents significant challenges for observational studies in the X-ray spectrum, unlike optical astronomy. As a result, X-ray astronomy emerged as a distinct field only in the latter half of the twentieth century.

### 1.2.1 THE BEGINNING OF X-RAY ASTRONOMY: SOUNDING ROCKETS

After World War II, V2 military rockets were repurposed by the U.S. Naval Research Laboratory to study X-ray emissions from the solar corona. Although it was determined that objects as bright as the Sun would not be observable with their instrumentation, researchers proceeded to investigate the existence of other non-solar celestial sources of X-rays. In 1962, a team from the American Science and Engineering company (AS&E), led by Riccardo Giacconi, launched an instrument equipped with Geiger counters and collimators to restrict the field of view (FoV). The instrument's FoV happened to pass over Sco-X1, the brightest known non-solar X-ray source, thereby confirming the existence of non-solar X-ray sources. This discovery increased interest among various research teams and led to the identification of additional X-ray sources [4].

### 1.2.2 EARLY X-RAY SATELLITES (PAST MISSIONS)

On December 12, 1970, NASA launched from San Marco, Kenya, the first satellite dedicated to X-ray astronomy: Uhuru [4]. The instrument, developed by Riccardo Giacconi's team, consisted of two sets of proportional counters, each with an effective area of 840 cm<sup>2</sup>. These counters were arranged back-to-back and equipped with

collimators measuring  $0.52^\circ \times 0.52^\circ$  and  $5.2^\circ \times 5.2^\circ$  to restrict the FoV. The proportional counters had a sensitivity of more than 10 percent in the energy range of 1.7–18 keV, allowing the satellite to detect sources with a flux as low as  $5 \times 10^{-4}$  that of the Crab Nebula. Uhuru's primary objectives were to conduct a sky survey for sources with this sensitivity, to determine the locations of such sources, and to study the structure of extended sources by analyzing their gross spectral features and variability. Additionally, the mission aimed to coordinate simultaneous observations with other researchers whenever possible [4].

Uhuru was a highly successful mission that conducted a comprehensive survey of the X-ray sky, identifying 339 X-ray sources and detecting diffuse X-ray emission from clusters of galaxies [5].

Following the success of Uhuru in 1978, NASA launched the second of its three High Energy Astrophysical Observatories, HEAO-2, known as the Einstein Observatory. Renamed to celebrate Einstein's centenary, it provided for the first time true X-ray imaging using Wolter type 1 focusing optics sensitive in the 0.1 to 4 keV band [5].

Einstein carried four primary instruments in its focal plane: a High-Resolution Imager, an Imaging Proportional Counter, a Solid-State Spectrometer, and a Focal Plane Crystal Spectrometer. Additionally, it included a Monitor Proportional Counter and an Objective Grating Spectrometer that operated in conjunction with the High-Resolution Imager. By combining focusing optics with proportional counters, Einstein achieved the sensitivity to discover and resolve many new sources, perform the first morphological studies of supernova remnants, and detect X-ray jets from Centaurus A aligned with its radio jets. Operated as a multi-institution collaboration under the NASA umbrella, Einstein was the first X-ray mission to host a guest observer program, laying the groundwork for today's collaborative research model [4].

The Röntgen Satellite (ROSAT) followed the Einstein Observatory as the next major X-ray observatory. It was a collaborative mission between Germany, the United States, and the United Kingdom, equipped with both X-ray and EUV instruments. ROSAT was launched on June 1, 1990, and operated for nearly nine years before being deactivated on February 12, 1999 [5].

ROSAT was equipped with two main instruments: the X-Ray Telescope (XRT), featuring four nested Wolter-1 mirrors and either two Position Sensitive Proportional Counters (PSPCs) or a High-Resolution Imager (HRI) placed at the focal plane; and the Wide Field Camera (WFC), a co-aligned device using Wolter-Schwarzschild mirrors, microchannel plates at the focal plane, and an eight-filter wheel.

During its initial six months of operation, ROSAT conducted a comprehensive full-sky survey, which was the first to utilize X-ray imaging instruments, supplemented by EUV data from the WFC. With sensitivity nearly 1,000 times greater than Uhuru, ROSAT

catalogued over 150,000 objects. Following the completion of the six-month survey, it continued operations for an additional eight years, supporting a guest observer program [4].

---

### 1.2.3 X-RAY SATELLITES (ACTIVE MISSIONS)

Although several X-ray missions have been launched over the years, with some still operational today, two missions, NASA's Chandra and ESA's XMM Newton, stand out as the key observatories of the current generation.

Chandra was launched aboard the Space Shuttle Columbia on July 23, 1999, into a highly elliptical orbit, enabling long, uninterrupted observations of celestial objects. The telescope combines sub-arcsecond angular resolution with significant collecting area and sensitivity up to 10 keV. These capabilities facilitate the study of extremely faint or heavily absorbed sources in dense regions, allow for the resolution of fine structure in supernova remnants and galactic nuclei, and enable the detection of weak point sources that were previously unobservable. With a spatial resolution better than 1 arcsecond on-axis and a field of view of approximately 30 arcminutes [5], Chandra still holds the record for the finest imaging ever achieved in X-ray astronomy.

Four focal-plane instruments can be positioned behind the mirror: the Advanced CCD Imaging Spectrometer (ACIS) with ten CCDs (eight front-illuminated, two back-illuminated) covering 0.2–10 keV, the High-Resolution Camera (HRC) with two microchannel plate detectors for imaging in the same range, and two transmission grating spectrometers, HETG and LETG, that provide high energy resolution data when placed in front of ACIS or HRC. ACIS CCD technology was first utilized on the ASCA X-ray astronomy mission in 1993.

Chandra's legacy includes imaging shock fronts in supernova remnants (SNRs), mapping hot gas in galaxy clusters to trace dark matter, discovering X-ray jets from active galaxies, and revealing a compact object in the Cas A SNR. Its ongoing observations continue to advance our understanding of black holes, neutron stars, galaxy evolution, and cosmic structure.

XMM Newton was launched on December 10, 1999, on board an Ariane 504 rocket. It remains the largest scientific satellite ever placed in orbit by the European Space Agency. The mission was designed to study the most energetic processes in the universe through simultaneous observations in X-rays and optical light. To achieve uninterrupted exposures, the satellite was inserted into a highly elliptical orbit with a 48-hour period [6].



The observatory carries three identical X-ray telescopes based on the Wolter type I design. Each telescope uses 58 nested mirror shells, is 7.5 meters long and has an outer diameter of 70 centimeters. The mirror modules were tuned to maximize effective collecting area in the 0.1 to 10 keV range, with special emphasis on the 7 keV band. This yields an effective area of 1500 cm<sup>2</sup> at 2 keV and 900 cm<sup>2</sup> at 7 keV [6]. The central field of view delivers an angular resolution that allows to study complex crowded regions down to 5 arcseconds [7].

Behind two of the X-ray telescopes, Reflection Grating Spectrometers (RGS) are mounted. These devices intercept roughly half of the incoming X-rays and disperse them across nine CCD detectors in spectrometer strips. The gratings and detector combination achieve an energy resolution ( $E/\Delta E$ ) of 150 to 800 over the 0.35 to 2.5 keV band [8].

At the focus of all three telescopes sits the European Photon Imaging Camera (EPIC). EPIC consists of three CCD-based detectors. Two of these are Metal Oxide Semiconductor arrays placed behind the grating modules, known as MOS. The MOS detectors are front-illuminated in a seven-chip array that can operate with readout times down to 2.6 seconds [9]. The third unit employs a monolithic array of twelve back-illuminated pn-CCDs and is unobstructed by the gratings with a time resolution down to 73.3 ms [10]. The three CCD cameras cover a 30-arcminute diameter FoV in the 0.15 to 15 keV energy band with a spectral resolution ( $E/\Delta E$ ) of 20 to 50 [10].

XMM Newton also carries an Optical/UV Monitor, consisting of a 30-centimeter telescope with ultraviolet and optical filters that operates simultaneously with the X-ray instruments [11]. This capability provides vital multiwavelength context for targets ranging from active galaxies to stellar coronae.

Over more than 25 years of operation, XMM Newton has transformed our view of the high-energy universe. It has revealed the processes driving Jupiter's spectacular auroras [12], measured the surface temperatures of three exceptionally young and unusually cold neutron stars [13], and detected the faint glow of hot gas threading through the Perseus galaxy cluster [14]. Its deep surveys continue to enrich our understanding of galaxy evolution and the growth of supermassive black holes.

## 2 THE (NEW)ATHENA MISSION

After briefly reviewing key past and current X-ray astronomy missions, this chapter focuses on the future of X-ray astronomy introducing the Athena observatory, summarizing its advanced instruments, scientific goals, and expected impact on high-energy astrophysics in the coming decades.

The Athena mission was selected in 2014 as the second large-class mission within the European Space Agency’s Cosmic Vision program. Following a reassessment of development costs for such a sophisticated instrument, compounded by economic challenges including those arising from the COVID-19 pandemic, the mission was re-scoped and officially renamed “NewAthena” in November 2023, aiming to be operational in the late 2030’s. For clarity and continuity, this document will continue to refer to the mission as Athena. Given the mission’s background and its re-scoping timeline, this chapter presents an overview situated within the historical evolution of the Athena mission. It begins by examining the scientific questions that motivated the proposal and subsequent approval of Athena in 2014. This is followed by a detailed description of the Athena spacecraft, highlighting its optical systems and scientific instruments. Given the mission’s rescope is important to remark that the spacecraft architecture is on-going a redesign with the final design proposal still on-going. The chapter concludes by discussing the mission’s science objectives, outlining potential applications and advancements made possible by the X-ray satellite as the project has evolved.

### 2.1 THE HOT AND ENERGETIC UNIVERSE

Athena, the Advanced Telescope for High-ENergy Astrophysics, was proposed in 2014 as a cornerstone mission within the framework of the European Space Agency’s Cosmic Vision 2015–2025 program. Its scientific objectives are rooted in the scientific theme of the hot and energetic universe, aiming to explore the physical processes that govern the most extreme environments in the cosmos. At the heart of Athena’s mission lie two fundamental astrophysical questions. The first asks how ordinary matter assembles into the large-scale structures that we observe in the universe today. The second seeks to understand how black holes grow and influence the formation and evolution of galaxies and cosmic structures.

To answer the first of these questions, Athena will be equipped with a unique combination of advanced features. As the largest X-ray observatory ever built, it will have a very large effective area, outstanding spectral resolution, and high-angular-resolution imaging capabilities. These qualities will enable Athena to detect and analyze the diffuse gas found in clusters, groups, and the intergalactic medium, which together

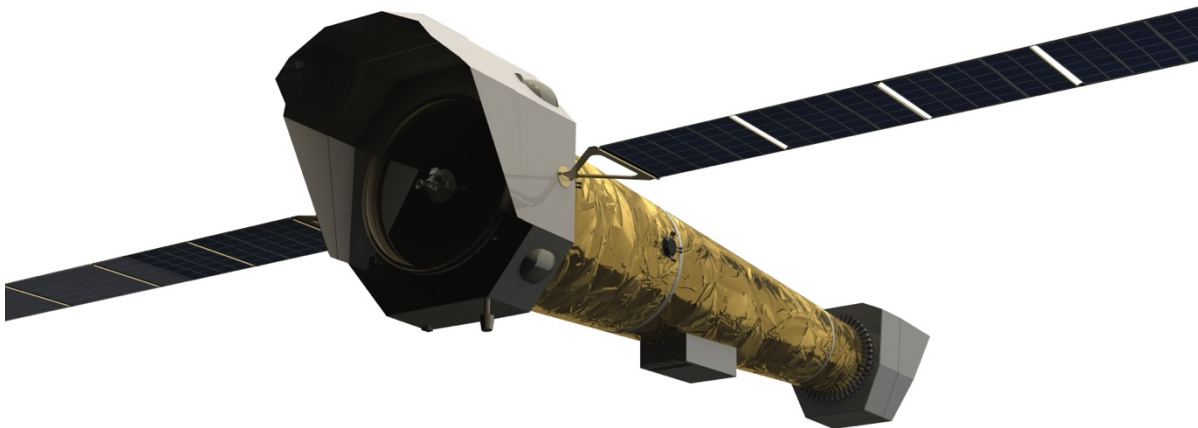
account for most of the baryonic matter in the local universe. Gaining insight into this hot gas is vital for understanding how clusters and groups evolve over cosmic time by accretion of this hot gas [15].

While the main component driving the growth of large structures is given by the distribution of dark matter on a large scale, processes of astrophysical origin play a major role as well. And in order to understand these processes, it is necessary to understand several physical properties of the hot gas, such as its velocities, thermodynamics, and chemical composition. The temperature of the hot gas makes it emit mainly in X-rays; however, current observatories do not have enough collecting area and spectral resolution to help answer this question. This is where the extraordinary capabilities of the Athena instruments, such as a large effective area with high spectral resolution, come into consideration [15]. The key element that Athena will enable is that it will be capable of seeing beyond the local universe ( $z > 1$ ), and with its spatially resolved spectral capabilities, it will allow for mapping and observing the physical properties of bound baryonic structures. By mapping the distribution and physical state of this diffuse gas, Athena will provide critical insights into the processes that drive the formation of galaxy clusters and the evolution of large-scale cosmic structures. Its performance will exceed that of any previous or existing X-ray mission, enabling transformative progress in our understanding of the universe's baryonic content and its dynamic behavior across cosmic time.

Addressing the second question (how do black holes grow and shape the Universe?) requires us to understand one of the critical processes shaping hot baryon evolution known as feedback from supermassive black holes. This process governs the energy input of hot baryons, and it bears high importance as the process happening at the scale of the event horizon seems to have widespread effects on structures 10 orders of magnitude larger, making this feedback mechanism of great importance to understand galaxy evolution. X-ray astronomy provides important information about these processes, revealing how winds close to black holes are generated and determining the coupling of the energy and matter flows. These important processes, however, are not completely understood yet. To provide key insight, Athena will help push our observation of black holes and study their evolution to high redshifts, the moment where the first galaxies were formed, at  $z = 6-10$ . With its extraordinary survey capabilities, it will be able to reach these early epochs and perform a census of the black hole population. This capitalizes on the high sensitivity of Athena, together with its high angular and energy resolutions, coupled with its wide field of view, making Athena a key instrument in aiding our understanding of these processes [16].

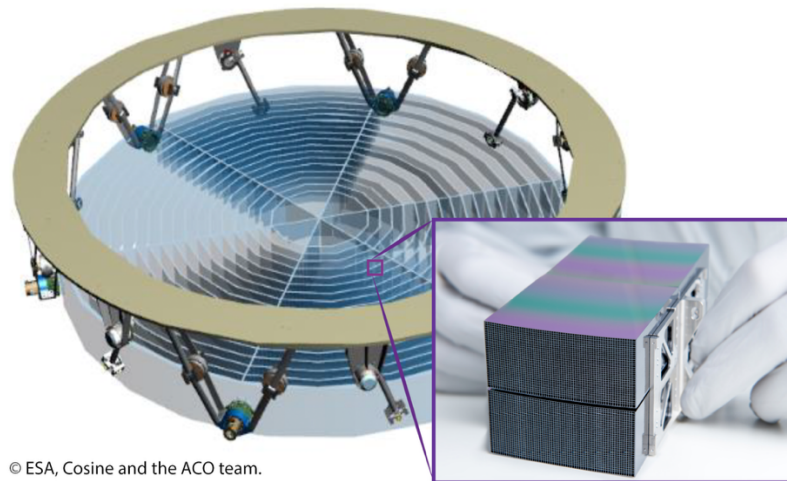
## 2.2 THE ATHENA SATELLITE

To reach its ambitious goals, Athena will be the largest X-ray observatory ever built, bringing “an unprecedented combination of collecting area, survey capabilities and energy resolution” [17]. Athena consists of a single 12-meter focal length telescope with a nested mirrors array in a Wolter I configuration utilizing Silicon Pore Optics (SPO) technology, which, thanks to its relatively light weight, is a key enabler for Athena’s outstanding collecting area of  $1 \text{ m}^2$  at 1 keV. The telescope shifts focus between two instruments housed in its payload module: the X-ray Integral Field Unit (X-IFU), an imaging microcalorimeter with exceptional energy resolution of 4 eV at 7 keV, and the Wide Field Imager, an active pixel detector providing a large field of  $40 \times 40 \text{ arcmin}^2$ , a better than 2 ms time resolution, and a quasi-Fano-limited energy resolution of 160 eV at 7 keV. A render of the Athena satellite flying can be found in Figure 1.



**Figure 1: (New)Athena satellite flying (rendered image) [18].**

To reach the surface quality required to achieve Athena’s outstanding resolution of 9 arcsec on-axis, a material with exceptional surface roughness qualities is necessary. This requirement is achieved by utilizing silicon wafers, which are readily produced in today’s semiconductor industry with the required surface figure and roughness for X-ray optics. Silicon wafers are diced into 60 mm wide rectangles, wedged, and grooves of 0.15 mm are cut. The wafers are stack pressed together, bonding them and creating parallel ribs in-between them, which act as the entry windows for the X-ray photons (the pores). To aid the X-ray reflection capabilities, the silicon pores are coated with a high-Z material; they are molded into a curved shape using precision mandrels which follow the paraboloid and hyperboloid surfaces needed for the first and second reflections for the Wolter type I configuration, an SPO module is thus composed by two vertically aligned stacks, the SPO modules are then placed in a nested concentric configuration to create the mirror array, see Figure 2 [19].



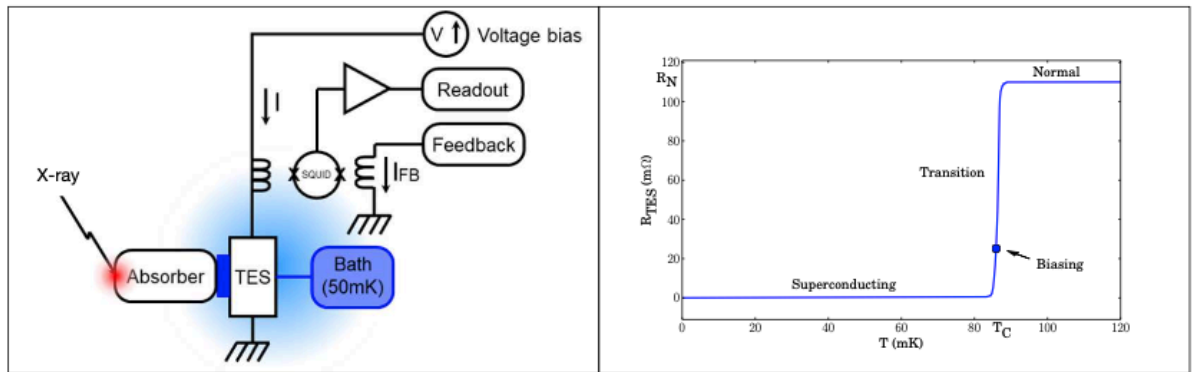
© ESA, Cosine and the ACO team.

**Figure 2: Conceptual design of the mirror structure (left) showing an SPO module (right) [20].**

SPOs are crucial for the effective area of Athena and thus the science goals of Athena, allowing, with their light weight, to incorporate enough reflective surfaces in a large diameter configuration. This technology provides a breakthrough in the weight-to-effective-area ratio, a feature that would not be possible with the traditional optics used in Chandra or XMM Newton, while still preserving state of the art angular resolution.

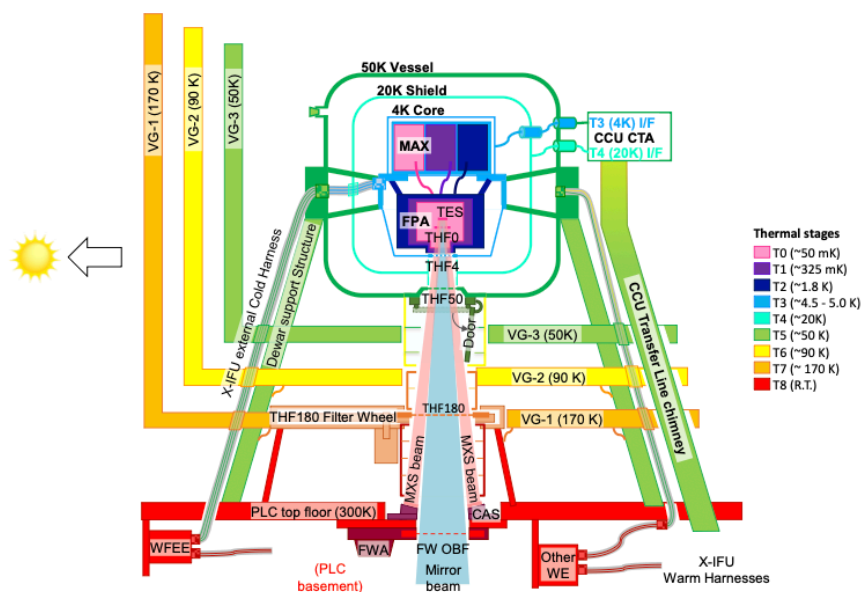
The first of Athena's instruments, the X-IFU, consists of an image capable state of the art X-Ray microcalorimeter utilizing transition edge sensors and operating in the 0.2 to 12 keV range. Its development is led by IRAP (France) with contributions from other institutions such as SRON (Netherlands), INAF (Italy) and other smaller contributions from several ESA member states.

To achieve the X-IFU's outstanding energy resolution of 4 eV at 7 keV, the instrument uses a Transition-Edge Sensor (TES) array located at its core. Calorimeters operate by detecting an increase in temperature in an absorbing material, which is proportional to the absorbed energy. X-IFU advances this approach by being able to detect temperature increases caused by a single X-ray photon, which is on the order of a millikelvin for a 7 keV photon. This means X-IFU must be capable of measuring temperature changes of a fraction of a millikelvin [21]. To achieve this precision, the X-IFU sensors utilize the conductive properties of metals when cooled near absolute zero, specifically when placed in the temperature range between superconductivity and normal conductivity. In this range, the smallest change in temperature has a significant impact on electrical conductivity (resistance), making it possible to measure minuscule temperature changes with great precision. In other words, X-IFU uses thermistors kept in the transition superconducting state to measure temperature changes with unprecedented precision, hence the name transition-edge sensors. The TESs are biased, and their resistance is continuously monitored, with changes in resistance indicating a change in temperature as seen in Figure 3 below.



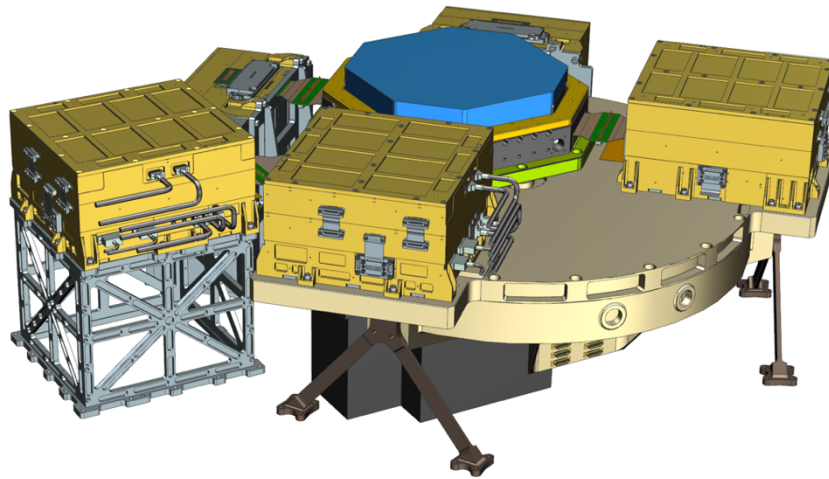
A TES consists of a  $50\text{ }\mu\text{m}^2$  bilayer film of gold and molybdenum, which is coupled to an Au/Bi/Au absorber. This absorber provides the X-ray stopping power and is optimized to maximize both energy resolution and dynamic range. The absorbers, or pixels, are arranged in a hexagonal pattern with a  $317\text{ }\mu\text{m}$  pitch, totaling 2,376 pixels divided into six groups of 396 pixels each, and covering a total FoV of  $4\text{ arcmin}^2$  [22].

Maintaining the TES at its operating temperature of 50 mK requires a highly complex thermal management chain, see Figure 4. This includes a passive radiator and shielding system located in the payload compartment, a 50 K dewar that provides an outer envelope hosting a 20 K shield and a 4 K core stage. Inside the 4 K stage, a multi-stage Adiabatic Demagnetization Refrigerator (ADR) cools the TES, which is located in the Focal Plane Array, down to 50 mK. Keeping the detectors at the optimal temperature also necessitates the development of sophisticated multiplexed readout electronics, which enable a 10  $\mu$ s readout resolution while ensuring that no heat is transferred from the electronics to the TES [23].



**Figure 4: Schematic view of the complex X-IFU cryogenic architecture [23].**

The second instrument onboard Athena is the Wide Field Imager (WFI), which provides, as its name suggests, a wide FoV combined with good energy resolution and high-count rate capabilities over the 0.1 to 12 keV range. As can be seen in Figure 5, the WFI's core is two active pixel sensors (DEPFETs). With the large sensor with a  $40 \times 40$  arcmin<sup>2</sup> FoV located on Athena's focal point and a second smaller sensor located purposely off focus to enable the observation of bright sources up to the Crab regime [24]. The WFI development is led by the Max Planck Institute for Extraterrestrial Physics in Germany, and it involves several German and European institutions such as the Institute for Astronomy and Astrophysics of the University of Tuebingen. A detailed description of the WFI instrument is given in the following chapter.



**Figure 5: CAD design of the WFI instrument, the two active pixel sensors are located at the center (blue), five detector electronics boxes surround the instrument (yellow) and a filter wheel is placed at the X-Ray window (beige) [25].**



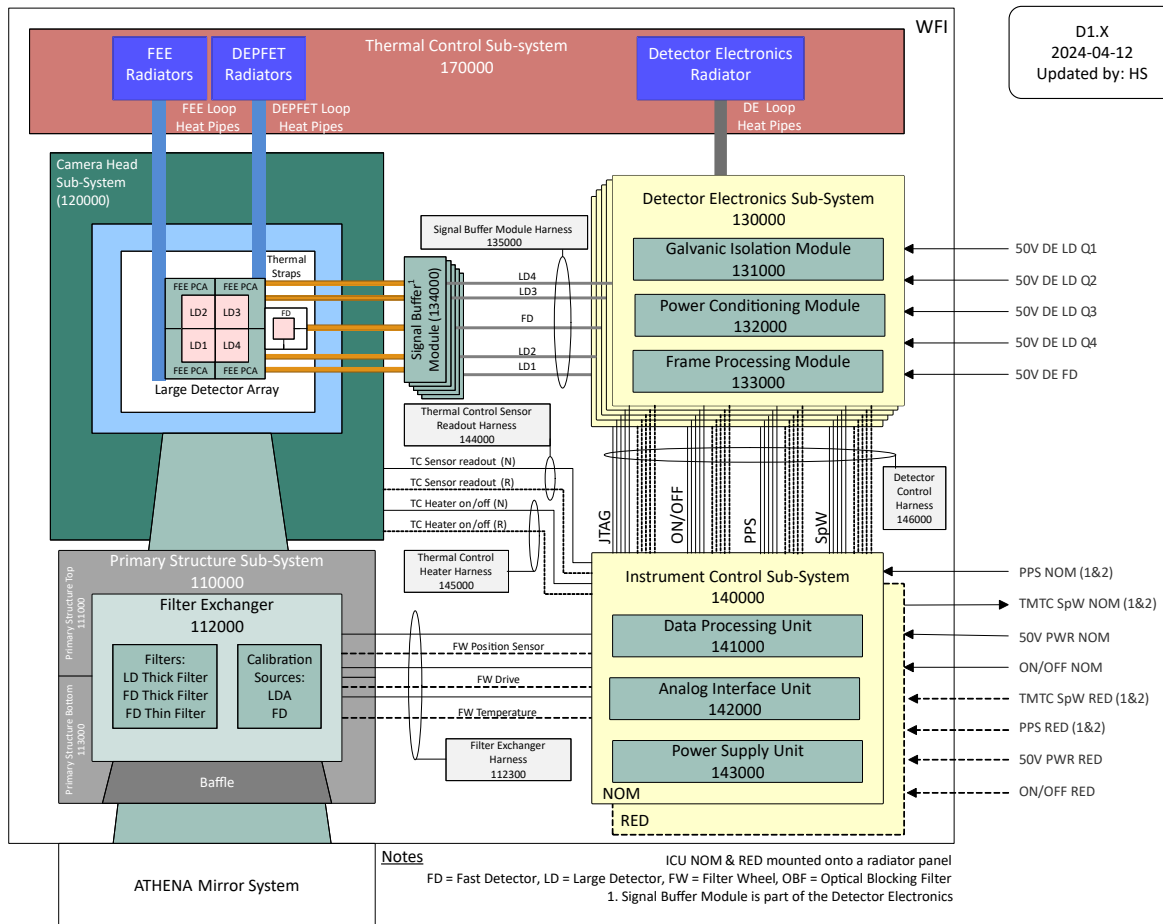
### 3 THE WFI INSTRUMENT

In this chapter, the technical details of the WFI instrument are described, covering its general architecture, data flow, and design tradeoffs. Focus is then given on the Detector Electronics Subsystem (DES) and, in particular, in the Frame Processor Module (FPM), where the main subject of this thesis, the frame processor, resides.

#### 3.1 WFI INSTRUMENT ARCHITECTURE

As described in the previous section, the WFI instrument is one of the two instruments in the focal plane of ESA's new-generation X-ray observatory, Athena. It provides two detectors, one with a large field of view capability of  $40 \times 40$  arcmin<sup>2</sup>, the large detector array (LDA), and a detector with very high-count rate capability for the observation of very bright point sources, the fast detector (FD) [26]. The instrument can be subdivided into four main subsystems: the primary structure subsystem (PSS), the camera head subsystem (CHS), the detector electronics subsystem (DES), and the instrument control subsystem (ICS). The PSS provides the main mechanical structure providing the mounting point of the WFI instrument to the science instrument module (SIM), contains the filter wheel (FW) and a shielding baffle. The CHS, mounted behind the PSS, is the front-end of the instrument, where the LDA and FD DEPFET sensors are located. The analog signals coming from the sensors, containing the sensors' acquisition information, are sent from the CHS to the DES, where they will be digitized and preprocessed in the Frame Processor Module (FPM). The FPM is one of three modules in the DES, together with the power conditioning module (PCM) in charge of providing the different bias voltages to the CHS and the galvanic isolation module (GIM), which receives the main power lines coming from the S/C and generates the intermediate voltages used by the PCM [27]. The FPM communicates with the ICS via SpaceWire protocol (SpW) and transmits all the scientific preprocessed data. The ICS compresses and packs the data and is the interface to the spacecraft (S/C). The ICS also controls the CHS filter wheel and the DES; Figure 6 provides a simplified block diagram view of the WFI instrument.

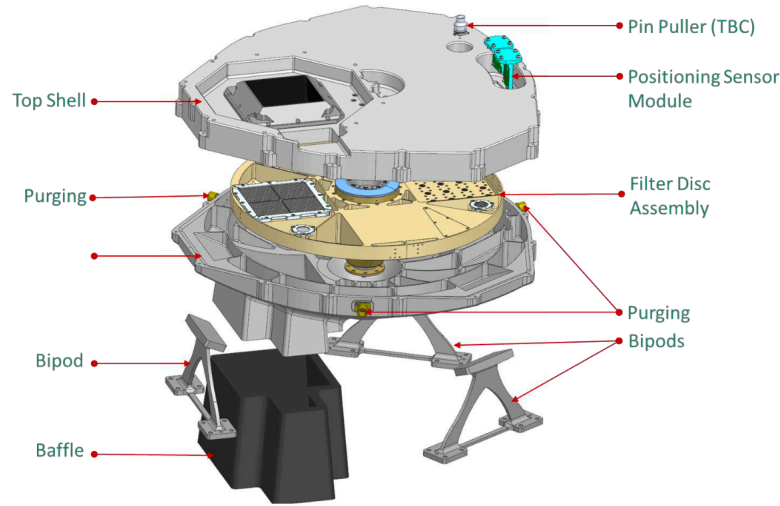




**Figure 6: Block diagram with the general architecture of the WFI instrument [27].**

### 3.2 THE PRIMARY STRUCTURE SUBSYSTEM

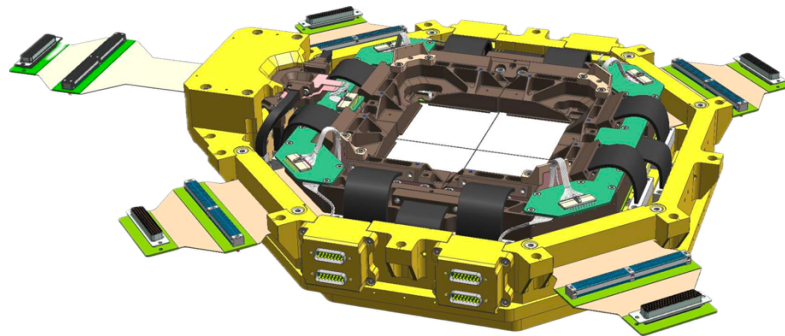
The PSS, as its name suggests, provides the mounting point for the WFI instrument to the SIM via three mounting bipods, shown in Figure 7. It houses the FW containing optical blocking filters that enable bright sources observation and calibration sources which enable in-flight energy calibration. In between, the FW entry window and the SIM structure, a shielding baffle is located, serving as an X-rays aperture and protecting the DEPFET sensors from radiation, a graded-Z shield which prevents the baffle from generating additional fluorescence lines background is foreseen. Three filters are located in the FW, a thin and thick filter for the FD, and a thin filter for the LDA. Two calibration sources are also part of the FW, one for the LDA, and one for the FD. A graded-Z plate provides shielding for the sensors to reduce instrumental background [28].



**Figure 7: CAD model of the primary structure subsystem [29].**

### 3.3 THE CAMERA HEAD SUBSYSTEM

The CHS, shown in Figure 8, is composed of the two sensors LDA and FD and their respective thermal, mechanical, and electrical interfaces, a proton shield to protect the sensors from radiation damage, a light-trap to prevent photons from reaching the front of the sensors, and a graded-Z shield to reduce background and increase the energy resolution of the detectors [30]. The LDA is mounted directly in the focal plane of the Athena optics, while the FD is slightly off-focus. In front of the CHS, in between the optics and the sensors, the PSS is located, where the CHS is mounted, behind the filter wheel structure [28].

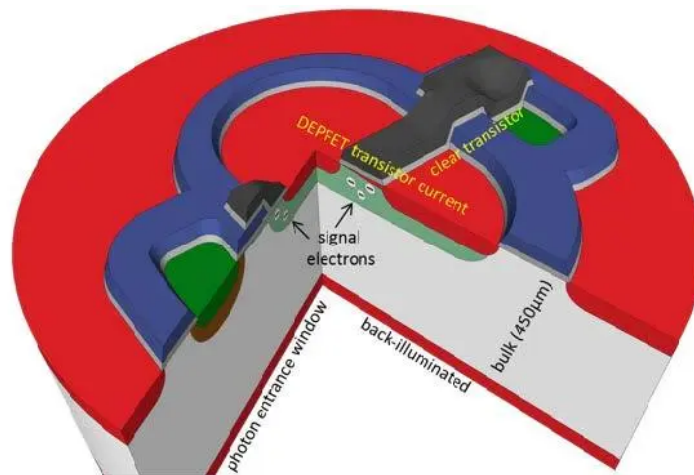


**Figure 8: CAD Model of the WFI Camera Head Subsystem [31].**

#### 3.3.1 THE DEPFET SENSORS

The WFI back-illuminated DEPFET sensors (shown in Figure 9) consist of arrays of  $130 \times 130 \mu\text{m}^2$  active pixels, each combining a detector-amplifier structure, where a P-Channel MOSFET is integrated into a n-type fully depleted silicon bulk [32]. High energy

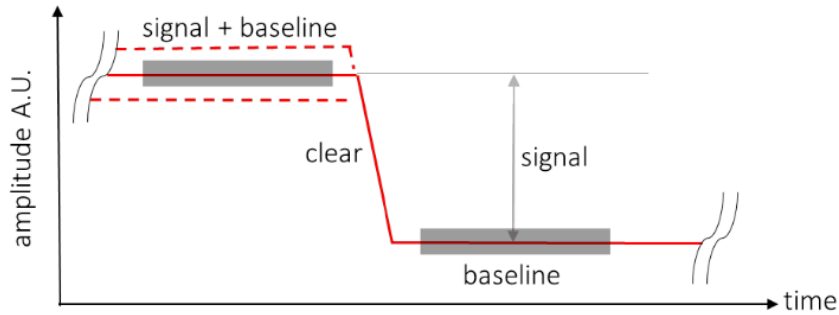
resolution readout is achieved through channel conductivity modulation, when an incident photon hits the silicon bulk the generated electrons are collected into the internal gate, the conductivity of the gate is proportional to the amount of charge deposited in the internal gate and therefore proportional to the energy of the incident photon [33]. To maximize quantum efficiency, in the energy range from 0.2 keV to 15 keV, the silicon bulk is 450  $\mu\text{m}$  thick and is fully depleted [34]. An optical blocking filter consisting of a layer of 90 nm of Aluminum is built in directly into the pixels' photon entrance window. The internal gate readout is performed using drain current readout, which enables higher readout speeds compared to the traditional source follower mode and enhances spectral performance by enabling longer integration times [30]. The readout process involves a double-sampling technique. A row of pixels is activated, and the currents in each channel are measured. Subsequently, the internal gates are cleared, and the channel currents are measured again. If a pixel was hit, the difference in currents between the two measurements is directly proportional to the energy of the incident photons [35].



**Figure 9: Schematic view depicting the structure of a round DEPFET sensor pixel [36].**

Such readout scheme (Figure 10) is enabled thanks to the usage of the Versatile Readout based on Integrated Trapezoidal Analog Shapers (Veritas 2.2) Application Specific Integrated Circuits (ASICs), which are capable of reading out 64 pixel channels in parallel, both in source follower and drain current readout mode, and which were optimized for fast-low noise readout required by the DEPFET sensors, enabling a sensor pixel-row readout speed of 2.5  $\mu\text{s}$  [28]. The activation and deactivation of the pixels' internal gates is managed by the Switcher ASICs, which are capable of activating a full pixel row and controlling up to 64 of such pixel-rows. Together with its readout electronics the Athena DEPFET sensors achieve Quasi Fano-limited spectral performance, an outstanding energy resolution of 61.1 eV at 1 keV and 138.4 eV have

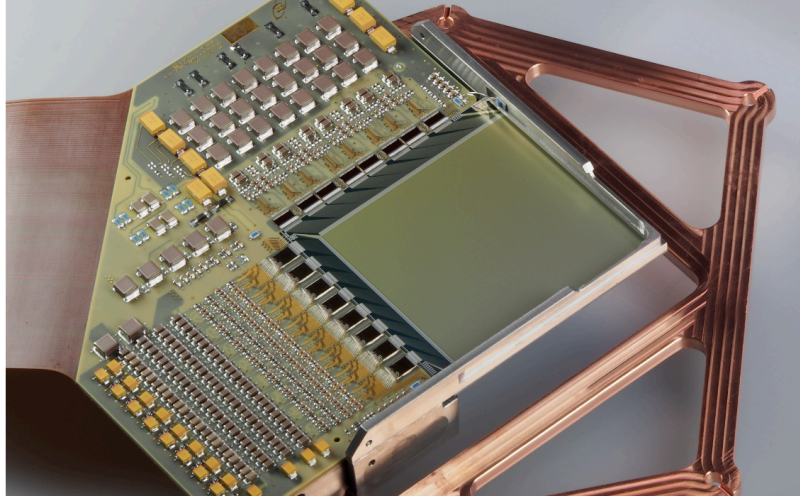
been recorded fitting well within the Athena WFI spectral requirements of 80 eV at 1 keV and 170 eV at 7 keV [34].



**Figure 10: Schematic showing the readout procedure of the DEPFET sensors, the baseline (obtained after the gate clear) is subtracted from the original measurement (signal + baseline) to obtain the photon signal [35].**

### 3.3.2 THE LARGE DETECTOR ARRAY

The LDA is the central element of the WFI instrument and is composed of an array of four Large Detectors (LD), each consisting of a 512 x 512 pixel DEPFET sensor, see Figure 11. Each LDA quadrant is mounted surrounded by a rigid-flex PCB containing the control and analog front-end ASICs (Switcher & Veritas ASICs, respectively). Eight Veritas ASICs mounted on one side of the sensor read a LD, with each Veritas having the capability of reading 64 channels (pixels) in parallel; the eight Veritas ASICs working in parallel, therefore, cover a 512 pixels row of the LD. Perpendicular to the Veritas ASICs, the Switcher control ASICs are mounted; these ASICs control the readout of the sensor by switching on and off detector rows in rolling shutter mode. Each Switcher can steer up to 64 pixel-rows; by combining eight Switcher ASICs, the readout of the 512 LD rows is achieved [30].



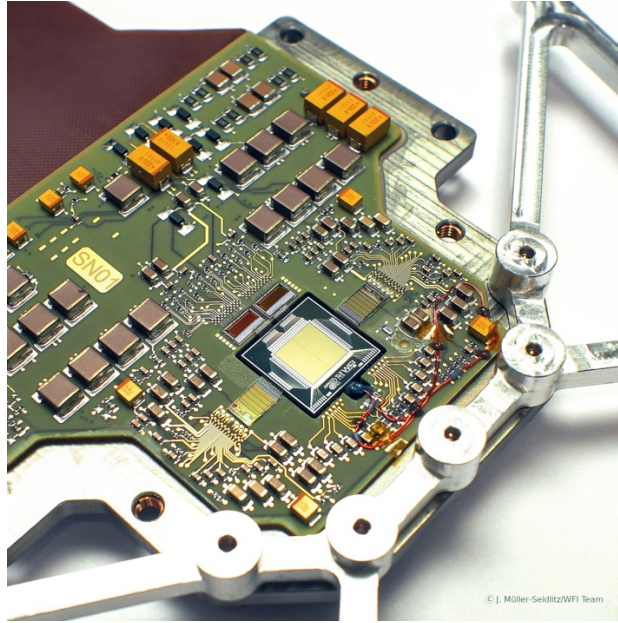
**Figure 11: Image of a large detector DEPFET of 512 x 512 pixels, eight Veritas ASICs (bottom left) readout the signals while 8 Switcher ASICs (top) steer the sensor readout [34].**

The four LDA quadrants, each mounted on its own PCB with front-end ASICs, form an array of 1024 x 1024 pixels. This parallelization enables a full LDA frame readout in less than 5 ms (with a target of 1.28 ms), significantly surpassing the readout speed of any other solid-state detectors in X-ray astronomy. The pixel size of  $130 \times 130 \mu\text{m}^2$ , combined with the Athena optics, allows each pixel to cover an area of the sky of  $4.8 \text{ arcsec}^2$  [30]. Consequently, the complete LDA offers an exceptional total field of view of  $40 \times 40 \text{ arcmin}^2$ . This combination of a wide field of view, spectral and timing performance makes the LDA instrument truly unique.

---

### 3.3.3 THE FAST DETECTOR

The fast detector (Figure 12), the second sensor in the WFI instrument, serves a crucial purpose: enabling the observation of extremely bright sources with exceptional spectral and temporal resolution [30]. To achieve this, it is positioned slightly off-focus from the Athena optics. This design strategy spreads the photons originating from point-like sources across the sensor's area, effectively helping in preventing pileup [26].



**Figure 12: Athena WFI fast detector prototype. Two Veritas front-end ASICs can be appreciated on opposite sides of the detector (bottom left and top right side of the detector), while two Switcher control ASICs steer the readout (top left detector side) [37].**

The FD is a DEPFET sensor, like the LD sensors, but with a smaller size of only 64 x 64 pixels. Its primary distinction from the LD lies in its exceptionally fast readout speed of just 80  $\mu$ s per frame. This remarkable speed is achieved by seamlessly combining the exceptional speed performance capabilities of the DEPFET active sensors with the rapid readout capabilities of the Switcher and Veritas ASICs. Unlike the LD, the readout of the FD involves reading out two detector rows simultaneously with two Veritas ASICs. Each Veritas ASIC reads out a row of 64 pixels, located on opposite sides of the sensor [34]. Furthermore, the two Switcher ASICs execute a rolling shooter mode, simultaneously reading out a sensor half, comprising 32 pixel-rows, from the sensor's center towards the sides, where the Veritas ASICs are located.

### 3.4 THE DETECTOR ELECTRONICS SUBSYSTEM

The detector electronics subsystem (DES) is responsible for driving the camera head sensors' readout, receiving analog signals from the Veritas ASICs, and providing power to the camera head. The DES comprises five detector electronics boxes, one for each LDA quadrant and one for the FD. Each box contains three modules: the Galvanic Isolation Module, the Power Conditioning Module, and the Frame Processor Module [28]. In this section, the focus will be on the functions of each module in the detector electronics, particularly the FPM.



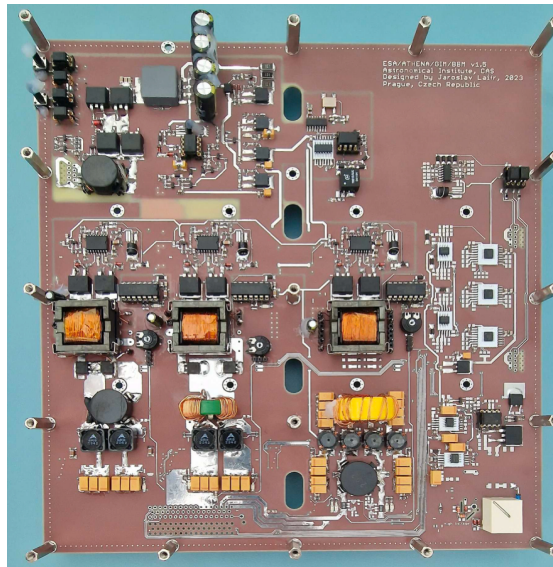
---

### 3.4.1 THE GALVANIC ISOLATION MODULE

The Galvanic Isolation Module (GIM), shown in Figure 13, primarily functions as a power converter, receiving the main +50 V power supply voltage and converting it into intermediate voltages while ensuring galvanic isolation. This galvanic isolation is crucial for safeguarding sensitive electronics from electrical damage and preventing the propagation of failures between satellite subsystems. The GIM achieves this through the use of transformers, which physically separate electrical circuits, thereby preventing unwanted current flow between them.

The GIM offers three sets of intermediate voltages, each generated using push-pull converters. Each intermediate voltage is further filtered by output filters. Additionally, it provides a high voltage of 0 to -200 V, which can be controlled by the power conditioning module using a diode multiplier.

The GIM's protection circuitry includes overvoltage and undervoltage protections, and the turn-on/off process is initiated by a latch switch via LPC discrete command from the instrument control unit. Furthermore, a discrete line Bi-level Switch Monitor (BSM) provides power good status readout [27].



**Figure 13: Breadboard model of the Galvanic Isolation Module [27].**

---

### 3.4.2 THE POWER CONDITIONING MODULE

The power condition module (PCM) is the second component in the DES. Its primary functions can be summarized as follows:

- It provides the supply voltages to the CHS.
- It provides Latch-Current-Limiter (LCL) protection circuitry.

- It monitors the voltages and currents status.
- It controls the high voltage output of the GIM.
- It serves as a passthrough for the FPM supply.

The PCM, shown in Figure 14, is crucial for the WFI instrument's exceptional energy resolution performance. It provides the 25 different supply voltages required for CHS operation, including fixed and variable outputs. To achieve low noise, the PCM employs voltage regulators instead of DC/DC converters and incorporates filtering circuitry at its outputs. The Vorago VA41630 microcontroller controls the variable voltages. This process involves controlling the PCM's Digital-to-Analog Converters (DACs) via an SPI interface. The DACs set the reference voltage for the variable regulators, proportionally modifying the output. Additionally, the microcontroller controls a DAC for the HV supply in the GIM. Latch circuitry ensures safe supply control, with latching switches on all control ports of the different supplies. A selected set of Latch-Current-Limiters (LCLs) are placed on selected power lines, providing protection in the case of radiation-induced Single Event Upset (SEU) events. In case of a microcontroller failure, all supplies remain in the same state until the microcontroller is safely rebooted. This is vital for CHS functional safety, as there is a fixed turn-on and off sequence for its supplies, and the microcontroller supervises this sequence. The GIM provides a +5V power line to the PCM. This supply also powers the Frame Processor Module (FPM). As mentioned earlier, the PCM generates housekeeping (HK) data on supply currents and variable voltages. It achieves this by reading its internal ADCs and some monitor ADCs in the GIM. The HK information is collected by the microcontroller and transmitted to the FPM via a serial Universal Asynchronous Receive Transmit (UART) interface. This interface is also used by the FPM to command the PCM [27].



**Figure 14: Engineering Model of the Power Conditioning Module. To reduce development costs, a mix of space qualified and commercial equivalent components are used.**



---

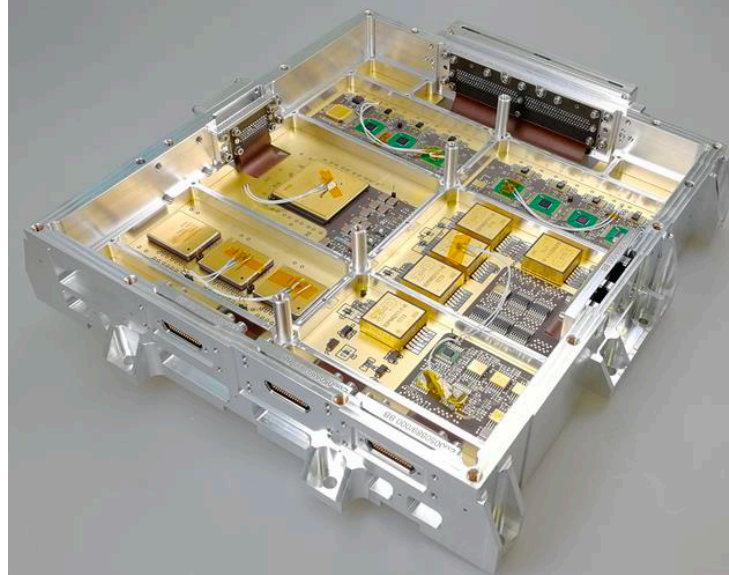
### 3.4.3 THE FRAME PROCESSOR MODULE

The Frame Processor Module (FPM) is the core of the detector electronics and shown in Figure 15. It receives analog signals from the CHS, digitizes them using 14-bit ADCs (AD9246S), acquires the data, performs real-time preprocessing and filtering, and organizes the relevant information into frame-wise event lists. These lists are then transmitted to the instrument control subsystem (ICS) via a SpaceWire (SpW) communication interface. Additionally, the FPM is responsible for driving and synchronizing the detector electronics readout by generating sequencing signals that control the Veritas and Switcher ASICs in the CHS. Furthermore, the FPM monitors and controls the PCM module via a serial UART interface between the two modules [38].

In this section, the focus will be on the description of the FPM hardware. Given that the Frame Processor (FP) is the main subject of this thesis, the details of the functionalities and internal design of the FPM will be discussed in great detail in the following chapters.

The CHS state-of-the-art sensors generate an enormous amount of data. For instance, the LDA, with its 1024 x 1024 pixels, is read out 200 times per second at a resolution of 14 bits per pixel. This alone generates approximately 2.9 Gbit/s, which increases to 3.65 Gbit/s when considering the FD. Considering the limited WFI data budget of 144 Gbit/day, on-board data pre-processing becomes crucial. To achieve this, a Field Programmable Gate Array (FPGA) is integrated into the FPM. This enables high-level parallelization in data pre-processing [39]. Each FPM receives data from a single DEPFET sensor. In the case of the LDs, data comes from eight Veritas ASICs, necessitating eight 14-bit ADCs in the LD-FPMs. Conversely, the FD only requires two Veritas ASICs for readout, so only two ADCs are used for digitization. Before digitization, the analog signals are buffered and preconditioned to match the voltage levels of the FPM. The FPM draws power from the +5 V power supply generated by the GIM and generates its own internal supply voltages for the FPGA and ADCs. Dedicated ADCs are used to monitor internal voltage levels and generate HK information. As mentioned earlier, the FPM also drives the readout of the front-end ASICs. This is accomplished through a train of parallel signals that switch between high and low states at a predefined timing. These signals are then transmitted to the CHS, using Low-Voltage Differential Signaling buffers (LVDS) which are also mounted on the FPM board. A Joint Test Action Group (JTAG) interface allows for configuration of the FPGA device and connects it to the ICS, providing the FPM with in-orbit reconfiguration capabilities. Additional LVDS buffers are used to drive the SpW communication signals to the ICS. The FPM also contains high-speed Synchronous Static Random Access Memory (SSRAM) memories that are utilized by the FPGA for data preprocessing in the case of the LDA. The FD, however, does not require such memories and relies solely on the internal resources of the FPGA. Finally, a High-Speed-Serial-Interface, exclusively

designed for development purposes, enables the FPM to transmit unprocessed raw data over gigabit Ethernet. This interface facilitates the comparison of raw data and preprocessed data to validate the FPM's functionalities by comparing its processing with external offline tools [40].



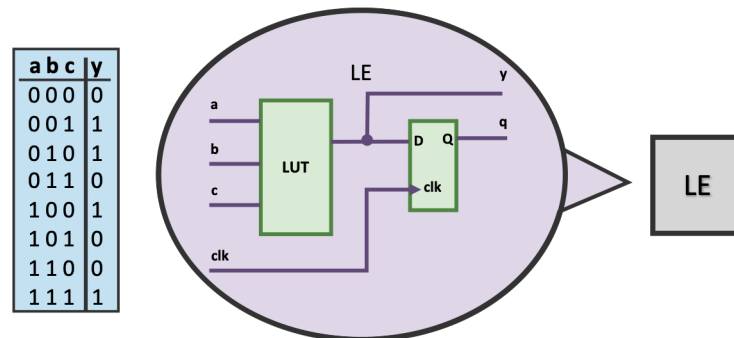
**Figure 15: Pre-Engineering Model of the FPM. A space qualified RTG4 FPGA is in the top-center. Eight ADCs are used to perform data digitization, to save resources, only one space qualified component was used, and the rest use commercial components. The same applies for the LVDS buffer (bottom corner) [40].**

#### 3.4.3.1 THE HEART OF THE FPM: FPGAS

Given their significance in the architecture design of the FPM and in general to the WFI instrument, this subsection delves deeper into the capabilities and features of Field Programmable Gate Arrays (FPGAs). Specifically, it focuses on the RTG4-FPGA, highlighting its unique characteristics that make it the ideal choice for carrying out the preprocessing tasks of the FPM. This enables the exceptional on-board processing performance required to read out the WFI sensors.

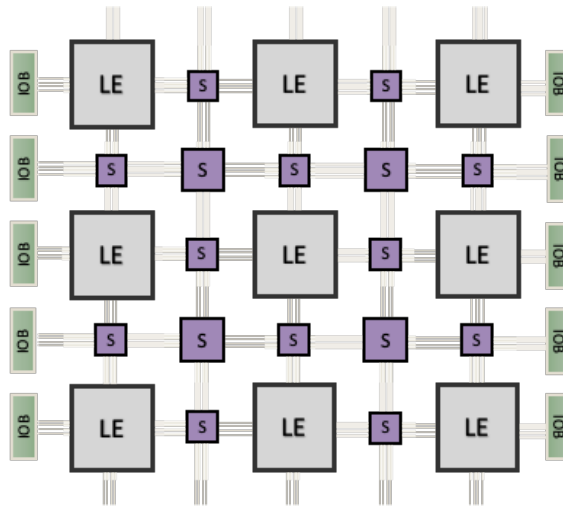
An FPGA is a type of integrated circuit (IC) which, unlike most ICs, is not designed to perform any defined function but instead is designed to be flexible and therefore programmed by the user to accommodate their logic needs. In essence, an FPGA is usually made up of small structures called Logic Elements (LEs), an example is shown in Figure 16, which vary slightly depending on the FPGA manufacturer. These LEs are fundamental logic units, each containing two basic elements: a small Look-Up Table (LUT) with three or four inputs and a programmable output that depends on the inputs. These logic elements are known as combinatorial logic because the combination of their inputs results in a specific output. Additionally, there is a clocking element, a Flip-Flop (FF). Each LUT's output is typically connected to the input of a clocking element,

which facilitates synchronization and collaboration among the outputs of multiple LUTs.



**Figure 16: Simplified schematic showing the internal structure of FPGA's Logic Elements. A Look-Up Table output is connected to a clocking device to keep synchronization. The content of the LUT can be configured as the user requires, an example is shown (left).**

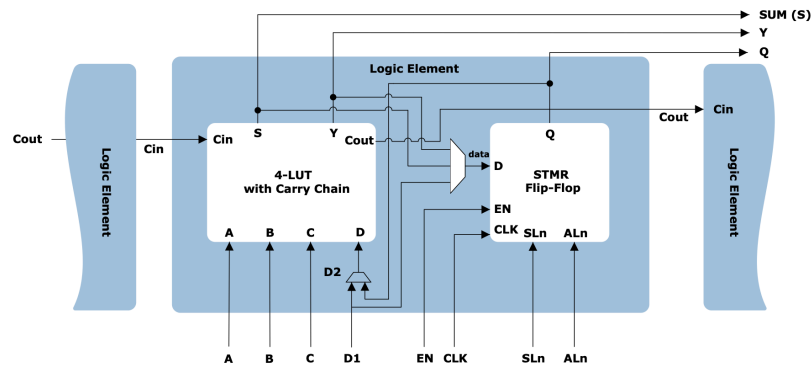
While an LE is not a very complex structure, the usage of many LEs together, synchronized by a unified clocking signal synchronizing the LEs FFs, enables the realization of any digital processing task, only limited by the amount of LEs at the user's disposal. An FPGA usually contains thousands to hundreds of thousands of these LEs. To connect the inputs and outputs of these elements, a routing and switching network interconnecting the LUTs is in place. LEs are connected or disconnected to each other accordingly thanks to small switching elements between the LEs and the routing network. Specialized programmable Input-Output Banks (IOBs) allow the user to connect the FPGA logic to external devices, making them extremely versatile. For a schematic illustration of the structure of FPGAs, see Figure 17. The FPGA user determines the way the elements are interconnected and the content of each LUT on the LEs using specialized Hardware Description Languages, commonly VHDL or Verilog. Manufacturer-specific tools are used to synthesize and program these designs into target FPGAs. An important feature of most FPGA devices, excluding technologies like anti-fuse FPGAs, is that they can be programmed and reprogrammed several times, making the development cost lower compared to the development of an ASIC.



**Figure 17: A simplified schematic illustrating the structure of an FPGA. LEs communicate with one another through a network of connectors and switches, and to the outside through specialized IOs.**

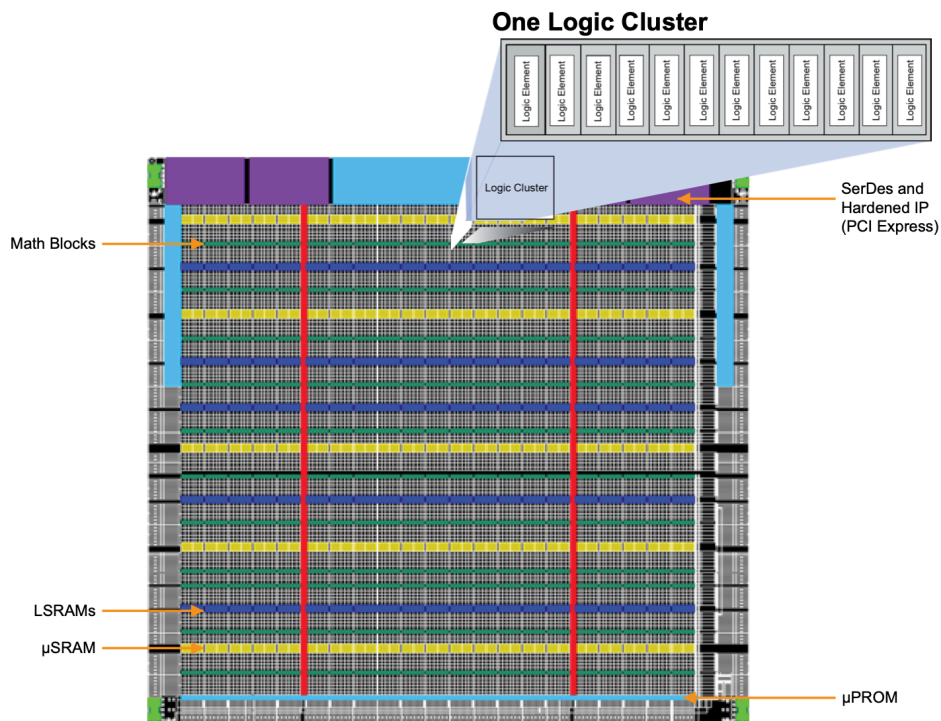
An important distinction between an FPGA and a processor is the fact that in the FPGA the logic is directly programmed in hardware according to the processing needs, while a processor uses hardware predefined structures to run software commands. This means that while processors tend to perform faster in many ways, they are restricted by the original hardware architecture and are generally optimized to perform certain instructions in a sequential order; one instruction follows another. FPGAs on the other hand can be optimized depending on the processing needs and are not limited by the processor architecture, making them ideal for heavy parallelization: One operation can be carried out by a group of LEs while another operation is carried out by a separate group in parallel. Therefore, an FPGA is an ideal device for tasks which require an incredible amount of data to be processed in real-time, such as the WFI FP design.

The FPM uses as a central element a Space Qualified RTG4 FPGA from Microchip. A functional block diagram of an RTG4 LE is shown in Figure 18. This device is Radiation Hardened by design (RadHard) and complies to the strictest reliability standards for space applications, QML-Class V [39]. It can withstand a Total Ionizing Dose (TID) of > 100 krad, is immune to configuration memory upsets, Single Event Latch-Ups (SELs), and highly resistant to radiation-induced Single Event Upsets (SEUs), caused when cosmic radiation interacts with semiconductor devices. The RTG4 is a flash-based FPGA, meaning that the configuration of its LEs is stored on its internal flash memory cells and does not require an external memory. The RTG4 contains a total of 151,824 LEs with Triple Module Redundancy (TMR), making them even more robust against SEUs in harsh radiation environments. This means that every Flip-Flop is triplicated with a majority voter at its outputs [41].



**Figure 18: Simplified functional block diagram of an RTG4 LE [41].**

Additional to the FPGA fabric, the RTG4 offers specialized logic resources, shown in Figure 19, such as 209 LSRAM memory blocks, 24.5 kb each, and 210  $\mu$ SRAM blocks of 1.5 kb each, distributed throughout the chip. Both have built-in Error Detection And Correction (EDAC) Single Error Correction and Double Error Detection capabilities. Additionally, it offers 462 Math blocks that allow quick computation of simple arithmetic operations and Serializer-Deserializer (SerDes) IOs for high-speed serial interfaces [41]. All the features mentioned above make the RTG4 the ideal device for the WFI Frame Processor. The following chapters delve into details of the FPGA design.



**Figure 19: RTG4 internal logic resources layout overview with labels pointing to the on-chip location of several specialized resources [41].**

### 3.5 THE INSTRUMENT CONTROL SUBSYSTEM

The Instrument Control Subsystem (ICS) serves as the primary interface between the S/C and the WFI instrument. It also handles instrument control, operates all the WFI subsystems, collects health data, and maintains control over operational modes. The ICS is divided into three units [28].

The first unit, the Digital Processing Unit (DPU), receives data streams from the DES via SpW, compresses the data, and sends the compressed packets to the S/C for ground download. As the primary instrument controller, it turns on and turns off the DES, collects HK data, provides mass storage for DES operational parameters, and implements the Fault Detection, Isolation, and Recovery (FDIR) logic. This includes the Disconnection of Non-Essential Loads (DNEL) and the instruments' safe mode operation.

The primary processing element of the DPU is a space-qualified RadHard Gaisler GR740 processor (see Figure 20), developed by the European Space Agency as part of its Next Generation Microprocessor initiative. This high-performance processor is well-suited for handling the data compression tasks required by the WFI instrument [42]. Its multicore design enables it to also perform the monitoring and control tasks necessary for the DPU's operations. To interface the GR740 with mass storage memory, an RTAX2000 anti-fuse technology FPGA is employed. Additionally, it provides interfaces for distributing the Pulse Per Second (PPS) signal used for instrument synchronization and supervises the GR740 processor's operation.

The second unit of the ICS is the Analog Interface Unit (AIU), which features a Gaisler GR718 processor specifically designed for performing WFI analog monitoring tasks. These tasks include collecting temperature data from the instrument, controlling the CHS heaters and FW position, and collecting ICS HK data to communicate it to the DPU via SpW.

The third and final ICS unit is the Power Supply Unit (PSU). As its name suggests, it is responsible for receiving power from the S/C and distributing it to all the systems of the WFI, excluding the DES, which receives its power directly at the GIM to avoid electronics noise [43].



**Figure 20: The GR740 processor, a high-performance processor that enables data compression for the WFI instrument [44].**



## 4 THE FAST DETECTOR FRAME PROCESSOR

The Frame Processor (FP) is a digital electronics design written using hardware description language (VHDL), which is synthesized into a single FPGA chip (the Microchip-RTG4). As discussed in the previous chapter, its main duties include acquiring the digitized signals coming from the WFI CHS performing real-time data analysis and preprocessing, packing and formatting the event data, communicating with the ICS, and driving the CHS Front-End Electronics.

Both the large detector array frame processor (LD-FP) and the fast detector frame processor (FD-FP) are designed to meet the same frame window capabilities, working in frames that are divisible down to windows of 64 x 64 pixels. However, the unique readout features of the fast detector necessitate the development of a separate firmware tailored to its specifications. In this chapter the FD-FP design is described in detail, together with the constraints that lead to its design and the tradeoffs that enable its unprecedented processing performance.

### 4.1 DESIGN PHILOSOPHY

To expedite the development process and leverage the modularity enabled by good FPGA design practice. The LD-FP and FD-FP are designed to be as modular and technology-independent (the code is easily portable to other FPGA technologies) as possible. Making their design portable not only between different FPGA technologies, accounting for the future developments in the space FPGA industry, but also to enable the reutilization of as many firmware modules as possible between the LD-FP and FD-FP. This goal of reusing the firmware units by only modifying their supporting systems and integrating logic enables not only to save development resources (time) but also to achieve a much more robust design, as also the firmware testing and design verification tests reinforce the reliability of both firmware designs.

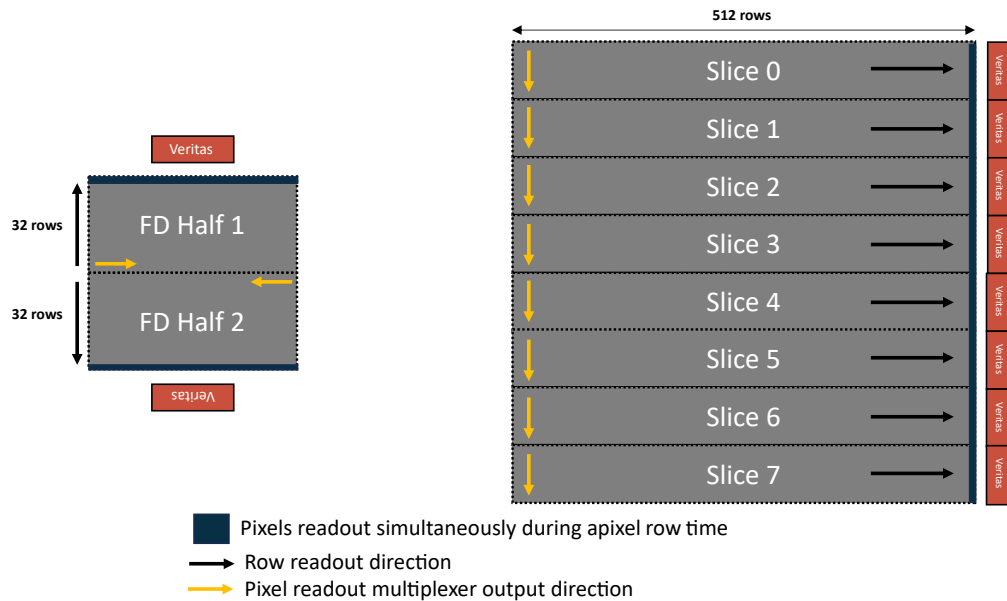
### 4.2 THE READOUT REQUIREMENTS OF THE FD COMPARED TO THE LDA

The LD and FD readout systems, despite using the same readout Veritas ASICs, exhibit a fundamental difference. The LDA sensor's slices are aligned in a single direction (towards the Veritas ASICs) for all readouts. This means that sensor rows are read out in parallel towards the same direction. In contrast, the FD sensor consists of two halves with their readout electronics positioned on opposite sides. Consequently, opposite readout directions read out sensor rows from the center to the edges, as illustrated in Figure 21. Furthermore, since the Veritas ASICs are rotated 180 degrees



from each other, the readout direction from each pixel in a row is also in the opposite direction.

This difference is negligible for most FP pipeline units because both sensors are read on a 64-pixel-per-row basis, and most units operate either pixel-wise or row-wise. However, the disparity in readout directions significantly impacts the Pattern Analysis Unit (PAU). If a photon strikes the border of both FD halves, the pipeline architecture requirements of the LD mean that it cannot detect and discard multipixel events, flag Minimum Ionizing Particle (MIP) neighboring pixels, or promote valid neighbor pixels across this boundary for the FD. Likewise, the FD pipeline working on rows of only 64 pixels would not be able to detect the events between data coming from different Veritas ASICs in the LDA, where rows are 512 pixels. To address this issue for the FD, the data input to the two Veritas ASICs reading out the FD is buffered and organized as a complete single 64 x 64 pixels frame. Once the data is organized, it is processed in the two parallel data pipelines of the FD-FP, enabling the PAU to perform the pattern analysis without limitations on the sensor center. For more details on this architecture, refer to section Memory Buffering Unit of this thesis.



**Figure 21: Schematic showing the line of pixels readout during a pixel-row time by 2 ASICs on the FD (left) and 8 ASICs on the LDA (right).**

As its name suggests, the FD has faster readout requirements compared to the LDA quadrants. This faster readout speed is achieved by reducing the number of pixel rows that need to be read on each frame. A pixel-row time refers to the time required to read a row of pixels in the sensor.

As mentioned in the previous chapter, a single LDA quadrant is read out by eight parallel Veritas ASICs. Each ASIC reads 64 pixels on the same row in parallel, resulting in a total of 512 pixels per row. Since an LDA quadrant is composed of 512 rows, a full frame is read in 512 pixel-row times. To select a row of pixels, eight Switcher ASICs are used. These ASICs activate and deactivate the detector rows that are read out by the Veritas ASICs. The FD sensor, on the other hand, comprises 64 rows, with only 64 pixels on each row. The two Veritas ASICs previously mentioned read each one half of the sensor simultaneously, resulting in the parallel reading of two rows per pixel row time. Consequently, the FD frame time is only 32 pixel-rows, which is 16 times faster than the LDA. Two Switcher ASICs are responsible for selecting the readout pixel rows. A summary of the readout specifications of the LDA and FD can be found in Table 1.

|                        | LDA    | FD     |
|------------------------|--------|--------|
| Number of rows         | 512    | 64     |
| Pixels per row         | 512    | 64     |
| Pixel row readout time | 9.8 ms | 2.5 ms |
| Parallel readout rows  | 1      | 2      |
| Frame time             | 5 ms   | 80 ms  |

**Table 1: Summary on the readout specifications of the LDA and FD.**

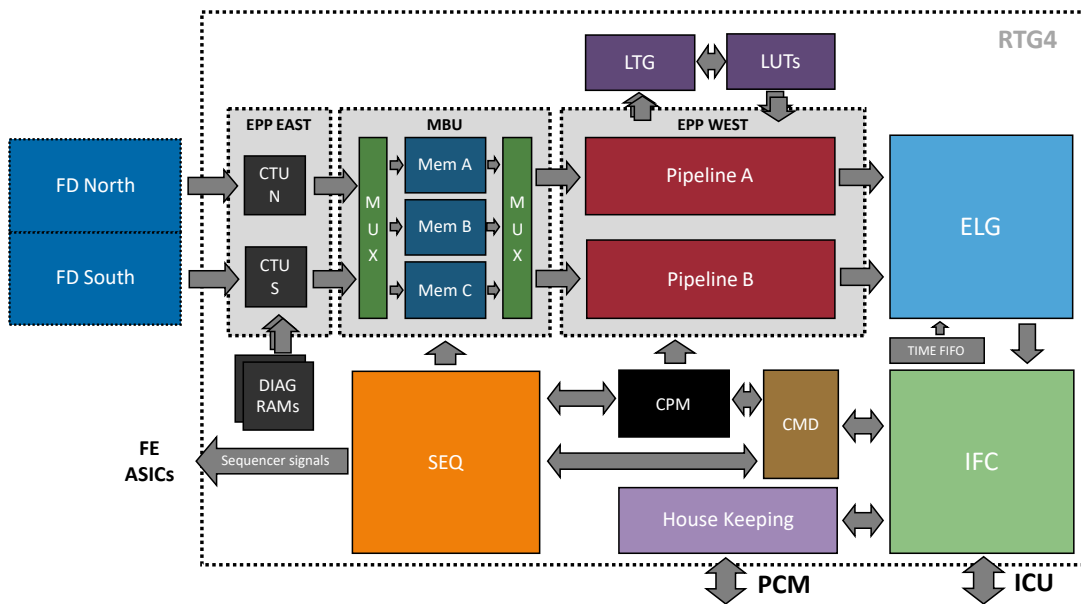
Being significantly smaller, measuring only 64 x 64 pixels in size, the FD contains 64 times fewer pixels compared to the sensors of the LDA. Consequently, the memory required to store the Look-Up Tables (LUTs) utilized by the Pixel Correction Unit (PCU) and Event Filtering Unit (EFU) is reduced by the same ratio. This allows the removal of external memories and enables the utilization of only the internal FPGA memory resources to store the LUTs. This approach results in numerous resource savings, including simplifying the PCB design, reducing the number of components, decreasing weight, reducing power consumption, and eliminating the need for complex memory controllers inside the FP firmware. Furthermore, the reduced pixel count allows for the internal memory to be used for the LUTs Generation Unit (LTG), significantly decreasing the calculation time required to generate the pixels' offset and noise maps.

### 4.3 THE FD FRAME PROCESSOR ARCHITECTURE

The architecture of the FD-FP is discussed in this section. The design aims to meet the requirements summarized in the previous section, but it is important to note that these requirements are derived from the official WFI technical requirements specification document.

### 4.3.1 ARCHITECTURE OVERVIEW

As previously discussed, the FD is more demanding on pipeline timing performance than the LDA. To meet the timing constraints, a double pipeline and buffer system are employed. These pipelines are interleaved to process each a 64 x 64 pixels frame sequentially (64 pixel-rows). However, the FD operates in a 32 pixel-row timing due to the two Veritas ASICs simultaneously reading out one sensor half and is read out from the center to the sides. Consequently, the MBU was introduced to solve this problem.



**Figure 22: General architecture block diagram for the FD-FP firmware.**

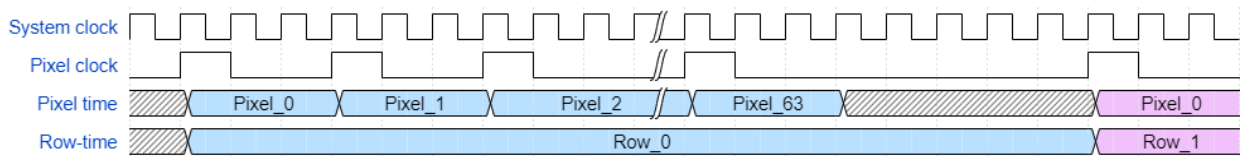
The general architecture of the FD-FP is illustrated in Figure 22. Most of its units are used both in the LDA-FP and FD-FP. With the most significant changes focusing on the general glue logic between the FP data input and the two parallel pipelines. These changes aim to buffer a complete frame, enabling each pipeline to process a full 64 x 64 pixel frame. This allows for alternating consecutive frames between the two pipelines, leaving sufficient processing time while buffering the next frame. The subsequent subsections delve deeper into the details and operations of each unit.

### 4.3.2 TIMING CONTROL AND CLOCK DISTRIBUTION

The previous subsections introduced the concept of pixel-row time. This subsection describes how pixel rows are constructed and how this affects the data preprocessing pipeline.

The FD sensor consists of 64 x 64 pixels. Each of the two Veritas ASICs read out one line of pixels at a time in parallel. However, the data from 64 pixels in a row cannot be transferred in parallel from the CHS to the DES. Therefore, a time multiplexing scheme is implemented. An analog differential voltage signal proportional to each pixel's energy is transmitted, containing the information of the 64 readout pixels in a sequential manner. As depicted in the simplified schematic of Figure 23, the time allocated for the transmission of each pixel is known as pixel-time. Consequently, a pixel-row time consists of 64 pixel-times plus an additional period (of a few pixel-times) between rows.

The concept of pixel-time is fundamental to the design of the pipeline, which is required to preprocess data in real time. This necessitates that each stage or step on the pipeline should be executed within a period shorter than a pixel time to ensure real-



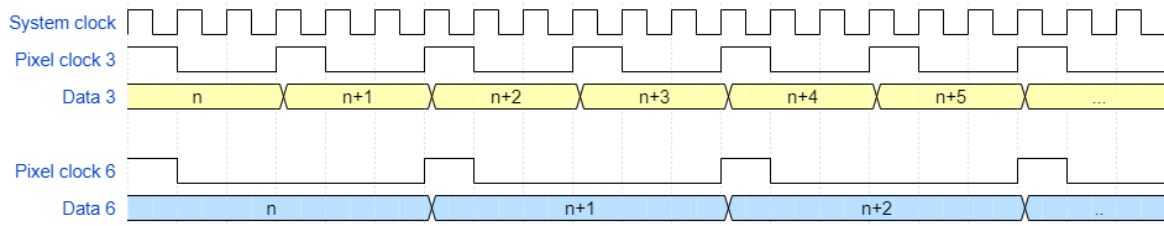
**Figure 23: Time diagram illustrating the pixel clock, pixel-time and pixel-row time.**

time processing capabilities.

To synchronize data while it is processed by the FP pipelines, a signal called the pixel clock is distributed within the FP units and externally to the ADCs that digitize analog signals from the CH.

The base system clock used for the FPGA implementation has a frequency of 80 MHz, corresponding to a 12.5 ns period. This is the minimum time unit used on the FPGA clocking resources and, therefore, the time unit used to measure the pixel clock period.

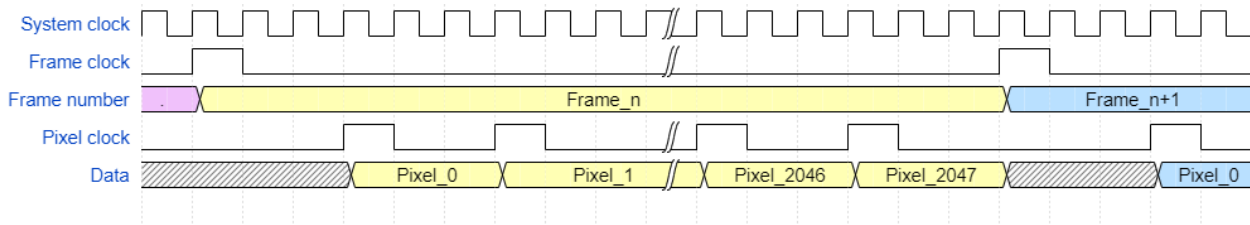
The pixel clock minimum period is limited by the FPGA's maximum speed and the number of computations required within a pixel clock cycle. To ensure reliable performance, a minimum of three system clocks are required within a pixel clock period, setting a minimum processing time requirement of 37.5 ns (three system clocks) for each computation stage on the FP pipelines. This translates to a minimum row readout time of approximately 2.4  $\mu$ s. Considering that a small period of 0.1  $\mu$ s is required between the readout of two rows and that a frame is read in 32 pixel-row times, a complete frame is read out in 80  $\mu$ s, meeting the FD timing requirements. The time diagram showing the pixel-time with respect to the system clock cycles can be seen in Figure 24.



**Figure 24: Time diagram illustrating the pixel-time with respect to the system clock cycles. Data is aligned for 3 system clocks per pixel-clock (above) and 6 system clocks per pixel clock (below).**

However, it should be noted that while three system clocks are the minimum requirement for a single pixel clock, this time can be increased to a higher number of system clocks if necessary. Therefore, care is taken in synchronizing the pipeline stages to the pixel clock, allowing for great flexibility on the pixel clock frequency.

Similarly to the pixel clock case, a signal is distributed within the FP units to ensure frame-wise synchronization. This signal, known as the frame clock (seen in Figure 25), is internally distributed and issued once per frame, marking the arrival of a pixel from the new frame at the FP entry. Since data arrives at different times in each pipeline unit from the beginning of the frame, several synchronization signals with varying delay times, expressed in pixel clocks, are generated in a central clock unit. These signals point to the beginning of a frame, or a row of pixels as required for each individual pipeline stage.



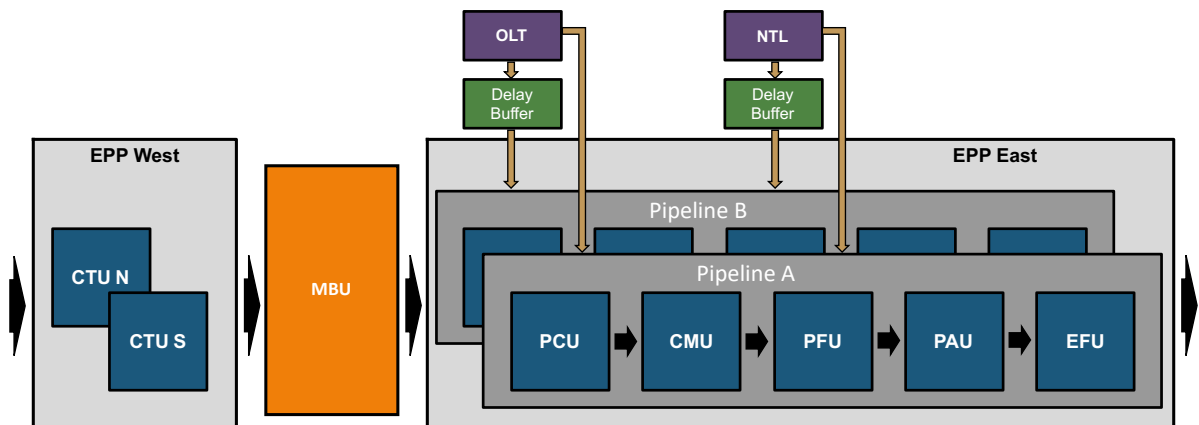
**Figure 25: Time diagram illustrating a frame clock and its alignment with respect to the frame data. It should be noted that a frame is composed of 2048 pixels.**

#### 4.3.3 EVENT PREPROCESSING PIPELINE

As mentioned in the previous section, the design philosophy of the FD-FP aims to use the same units as the LDA-FP, when possible. Given the multiplexing and buffering schemes described in the previous subsection, the FD pipeline units operate under the same time windows of the LDA-FP, enabling an overlap on a significant part of the LDA-FP and FD-FP designs. This allows the re-use of the some units in both pipelines

including the LUT memories. A simplified block diagram of the preprocessing pipelines of the FD-FP is presented in Figure 26. The pipeline is divided into three modules:

- The first module, called the Event Preprocessing Pipeline West (EPP West), contains only the Crosstalk Unit (CTU). Given that the CTU corrects for the crosstalk that affects the consecutive multiplexed pixels coming from the CHS, two CTU modules (CTU North and CTU South) are used inside the EPP West, each of them processing exclusively the signals from one specific half of the FD sensor.
- The second module on the EPP is the Memory Buffering Unit (MBU) which performs the frame buffering needed to sort frames and prepare them to be processed in the third and final module of the EPP, the EPP East.
- The EPP East contains two preprocessing pipelines (pipeline A and pipeline B), each containing a Pixel Correction Unit (PCU), Common Mode Unit (CMU), Pixel Filtering Unit (PFU), Pattern Analyzing Unit (PAU) and Event Filtering Unit (EFU). Unlike the EPP West, the two pipelines of the east module are not specifically related to a sensor half, but instead they work in parallel processing alternating frames each, hence the two pipelines inside the EPP East are known as pipeline A and pipeline B. To synchronize the offset and noise values coming from the external LUTs, a delay buffer line is employed, delaying the LUT inputs to the secondary pipeline (pipeline B) by 32 pixel-row times.

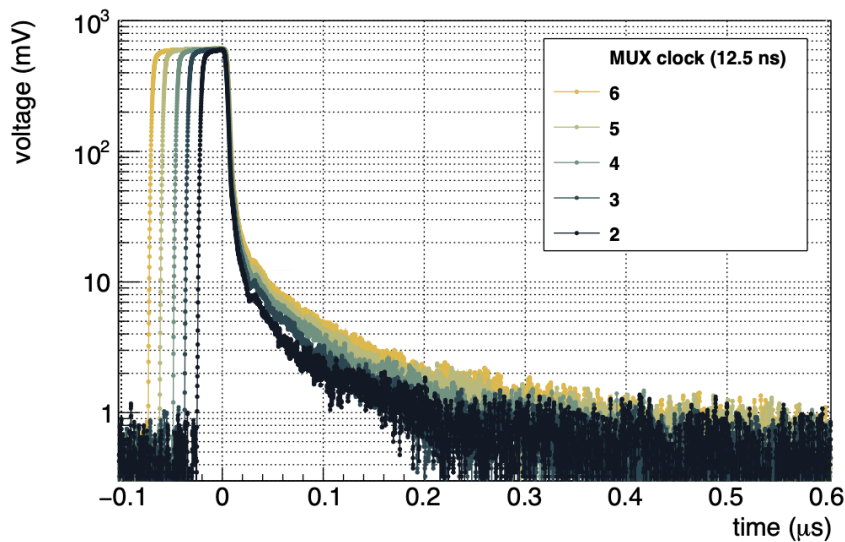


**Figure 26: Structure of the FP pipelines, it should be noted that apart from the MBU, the number of units is the same as for the LDA-FP.**

The following subsections describe in more detail each of the FP-FD units and their algorithms.

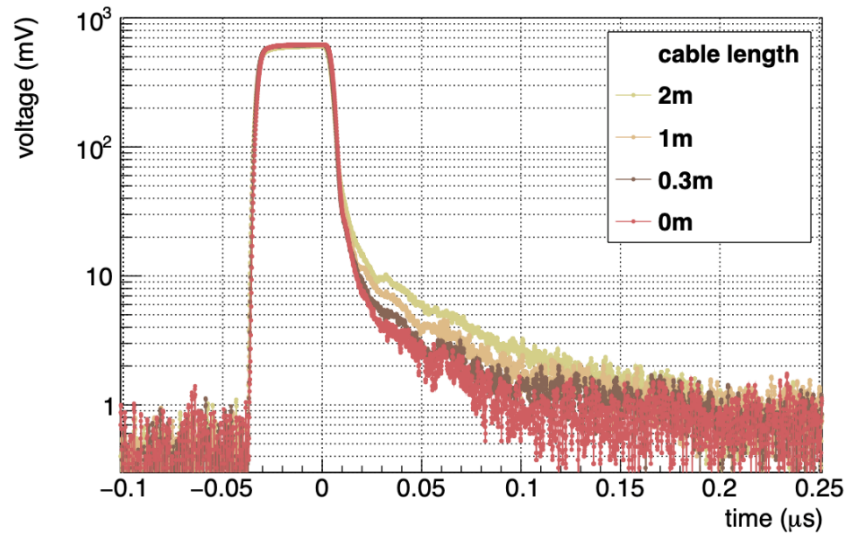
#### 4.3.3.1 CROSSTALK UNIT

The crosstalk unit (CTU) effectively reduces the crosstalk effect caused by bandwidth limitations at the output stage buffer of the Veritas ASICs. This effect occurs when transmitting analog signals from the CHS to the ADCs located at the FPM. It manifests as a noticeable spread of the charge from one pixel to the next consecutive pixel, resembling a tail. Consequently, pixels adjacent to event pixels have higher apparent energy. This effect results in a significant increase in double pixel events and the discarding of more than two pixels events confused as Minimum Ionizing Particle (MIP) events. Figure 27 illustrates the results of a study conducted on this crosstalk effect, demonstrating how the pixel energy fraction spreads into the subsequent pixels and its dependence on the pixel time: shorter pixel times lead to a smaller tail [45].



**Figure 27: Pixel energy fraction spreading to subsequent pixels as a function of the multiplexing clock (pixel-clock) length. Data is aligned to the falling edge for proper comparison [45].**

Similarly to the multiplexing times, shorter cables lead to shorter fall times, an effect most likely arising from the transmission line's capacitance, see Figure 28.



**Figure 28: Pixel energy tail spreading to subsequent pixels as a function of the cable length [45].**

To account for the crosstalk effect, a correction algorithm was implemented in the FP pipeline, which is executed in the CTU. The algorithm involves subtracting a correction factor, which is a fraction of the energy on pixel  $n$ , from the energy of pixel  $n+1$ . Additionally, a second (smaller) fraction is subtracted from the energy of pixel  $n+2$ . It is important to note that in this notation, pixel  $n$  refers to the pixel currently being read out, while  $n+1$  and  $n+2$  represent the following and second consecutive incoming pixels. The calculation factor is determined as follows, where  $R1$  and  $R2$  are the correction coefficients for the two preceding pixels and  $\alpha$  is a common denominator scaling the conversion factors:

$$Output(n) = Input(n) * \frac{R1 * Input(n - 1) + R2 * Input(n - 2)}{\alpha}$$

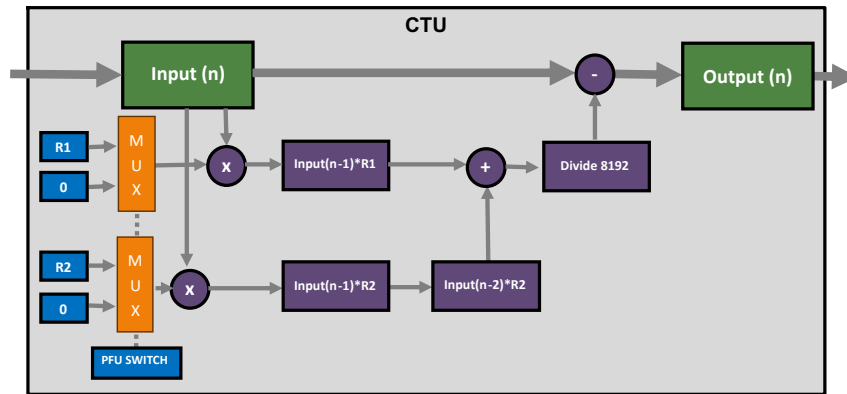
Given that the correction factor is highly dependent on the pixel clock frequency and the final transmission line length, the correction factors are programmable using two registers, `EPP_CTU_R1` and `EPP_CTU_R2`, via the frame processor command interface (CMD). These registers modify  $R1$  and  $R2$  respectively and both accept integer values between 0 and 255. The common denominator  $\alpha$  was chosen to be equal to 8192 to allow an easy division operation on their binary representation by performing a simple bit shift. Table 2 gives some sample values for the correction factors for different register values.



| R1, R2 | Correction (%) |
|--------|----------------|
| 1      | 0.012207       |
| 2      | 0.024414       |
| 3      | 0.036621       |
| 9      | 0.109863       |
| 20     | 0.244141       |
| 41     | 0.500488       |
| 82     | 1.000977       |
| 100    | 1.220703       |
| 127    | 1.550293       |
| 255    | 3.112793       |

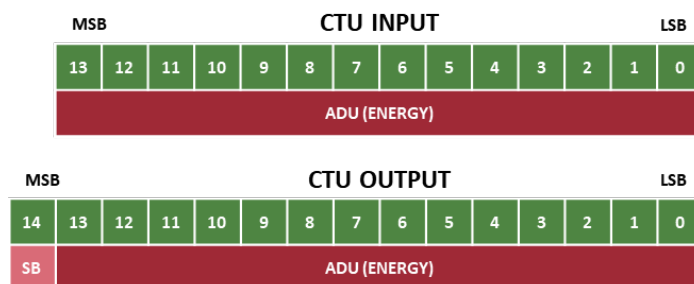
**Table 2: Examples of correction factors in percentage dependent on the R1 and R2 registers.**

The FPGA implementation of the unit utilizes specialized hardware-specific IP core units designed for rapid arithmetic computations, commonly referred to as Multiply-Accumulate (MACC) cores on Microchip FPGAs. These cores are efficiently combined to execute the required subtractions in a resource- and time-efficient manner. Figure 29 illustrates the logic implementation.



**Figure 29: Simplified block diagram of the CTU.**

It is important to note that the subtraction value can result in negative pixel energies, depending on the subtraction factors R1 and R2, the total energy of the preceding pixel, and the currently corrected pixel energy. To address this, a subtraction bit is added to the initial 14-bit register containing pixel energy information, making the output 15-bit wide. Values starting with '1' are considered positive while values starting with '0' are negative, see Figure 30.

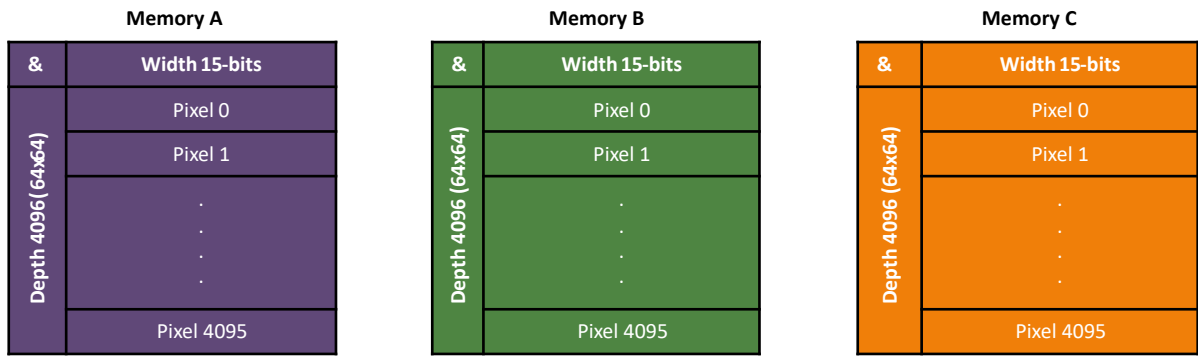


**Figure 30: Internal structure of the CTU input and output bits.**

#### 4.3.3.2 MEMORY BUFFERING UNIT

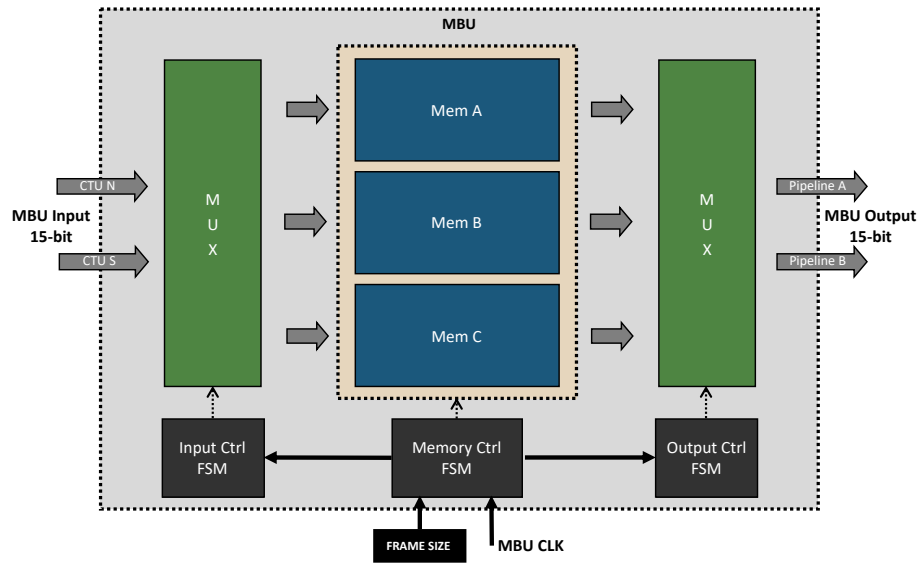
The MBU is responsible for receiving the two data streams from the CTU. It organizes these streams into frames and then synchronously outputs their contents to the two preprocessing pipelines. This section delves into the detailed operation and architecture of the MBU.

As mentioned in The Readout Requirements of the FD Compared to the LDA section, one of the biggest challenges posed by the FD sensor readout is the asymmetric readout of its data from the center of the sensor towards two opposite edges, where the two Veritas ASICs are located. This dual readout system generates two data streams, requiring two parallel pipelines to process the data of alternating frames during a pixel-row time. To manage these challenges, the MBU synchronously buffers the incoming frame data, sorts it, and outputs the information on the corresponding pipelines. To achieve this, the MBU utilizes three memories (A, B & C) each with a 15-bit width and a 4096-address depth (see Figure 31). These memories can store complete 64 x 64 pixel frame information. Each memory block utilizes the LSRAM memory resources of the RTG4 FPGA, with four of these memory blocks needed for each memory buffer.



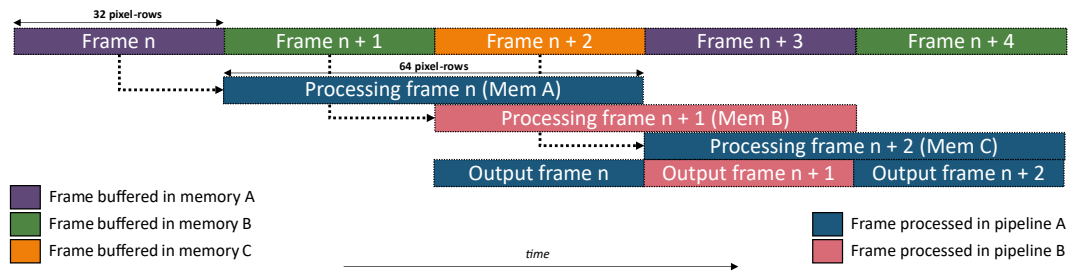
**Figure 31: Schematic showing the memory size and word distribution of the MBU memories.**

The architecture of the MBU is shown in Figure 32 , where it should be noted that the output of the two CTUs is connected to all the three buffer memories, from here on denoted as MA, MB and MC. At the same time the three memories are also connected to the two preprocessing pipeline inputs. This is achieved by exploiting the True-Dual-Port capabilities of the RTG4 LSRAMs. This feature implies that each memory has two access busses with independent address and both with read and write capabilities. This capability is fundamental for the correct functioning of the MBU as it enables the simultaneous storage of both CTU's data streams into a single memory. Once the full frame is acquired and stored in the first buffer memory (MA), its processing begins on pipeline A. Simultaneously, the second frame is acquired and buffered in the second buffer memory (MB). When the buffering process is complete, the second pipeline commences its processing on pipeline B. At the end of the second frame acquisition, MA is still occupied since only half of it has been processed by pipeline A to that point. Therefore, the third frame is buffered in a third memory MC. Once the frame is stored, pipeline A is free and immediately starts processing the data coming from MC. This process continues alternating memories and pipeline in a synchronous manner until acquisition is stopped. A timeline schematic illustrating these processes is presented in Figure 33.



**Figure 32: Block diagram depicting the multiplexing structure connecting the output of the sensors to the three buffering memories.**

To alternate the buffer inputs from the data incoming from the Crosstalk Unit (CTU), a multiplexing switch is introduced at the MBU entrance. This switch alternates the storage between the three memory buffers, ensuring that each line is stored in an organized manner. Similarly, the output of the three memories requires a multiplexing switch that routes the data to the two parallel pipelines. Both multiplexers are controlled by two separate FSMs: an input control FSM and an output control FSM. The synchronization between both FSMs and the memories' read and write operations is driven by a third FSM, the memory control FSM. The memory control FSM uses a clocking signal that points when data of a new frame arrives to the MBU known as MBU CLK. These clocking signals are generated directly on the FPs central clock unit. The synchronization and memory storage are also dependent on the frame size, meaning that when window mode operation is used the memory control FSM adjusts the synchronization accordingly. This process is handled automatically by the MBU depending on the status of the *FRAME SIZE* register.

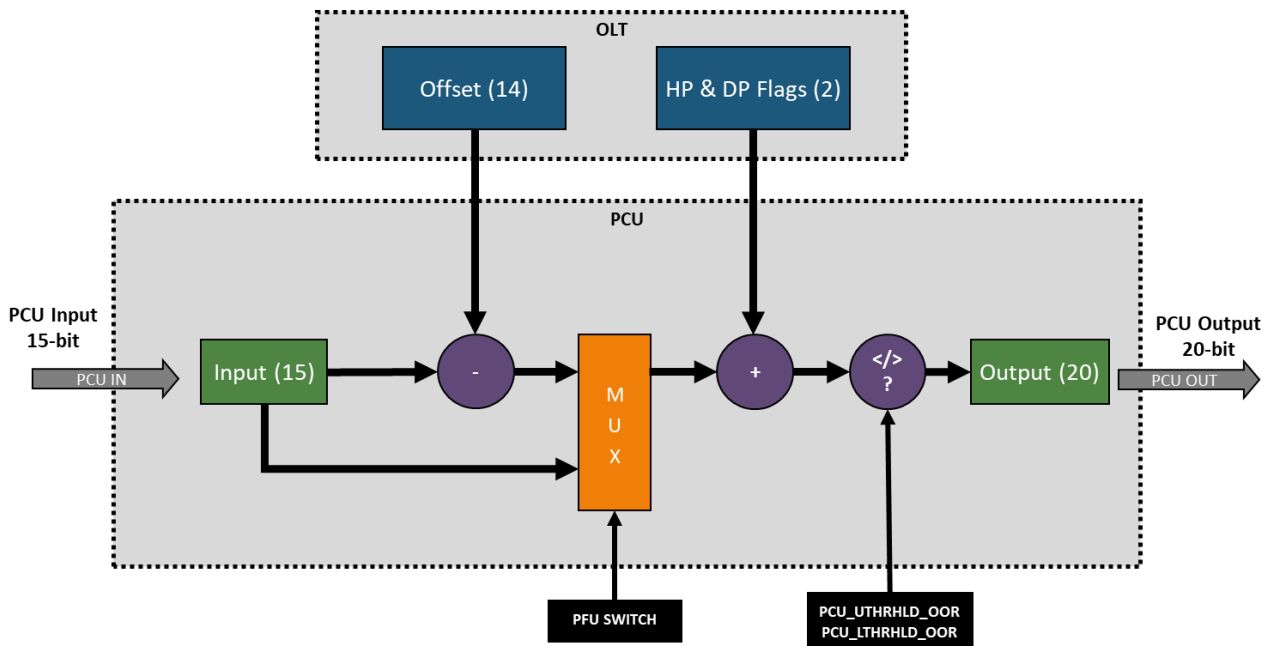


**Figure 33: Timeline schematic showing how consecutive frames are readout in the two pipelines.**

#### 4.3.3.3 PIXEL CORRECTION UNIT

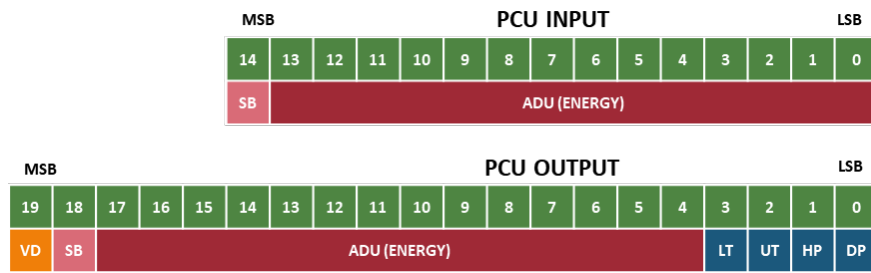
The pixel correction unit is the first stage on the EPP East. Its main function is to correct for the offset energy of each pixel on the detector and flag those pixels known to have a malfunction. The pixels' offset is mainly generated by the thermal fluctuation on the sensor crystals, which enables some electrons to gain sufficient energy to reach the conduction band, generating electron-hole pairs even in the absence of photons. To correct for this offset, the average energy of each pixel is calculated in dark-frame conditions. Once the mean energy value is known for each photon, the value can be subtracted from each incoming pixel, leveling out their energies.

The PCU is in charge of performing the pixels' offset subtraction. In order to do so, the PCU fetches the offset data for each individual pixel in the sensor, which is stored in the Offset Look-up Table (OLT). Once offset data is retrieved, the PCU simply subtracts the offset to each pixel in a sequential manner. In addition to the offset values, the most significant bits of the OLT are used to store information about the pixel health. Specifically, the OLT outputs a 16-bit word. Bits 0 to 13 contain the offset information. Bit 14 is set to '1' when a pixel is a Hot Pixel (HP), while bit 15 is set to '1' when a pixel is a Dead Pixel (DP).



**Figure 34: Simplified block diagram of the PCU.**

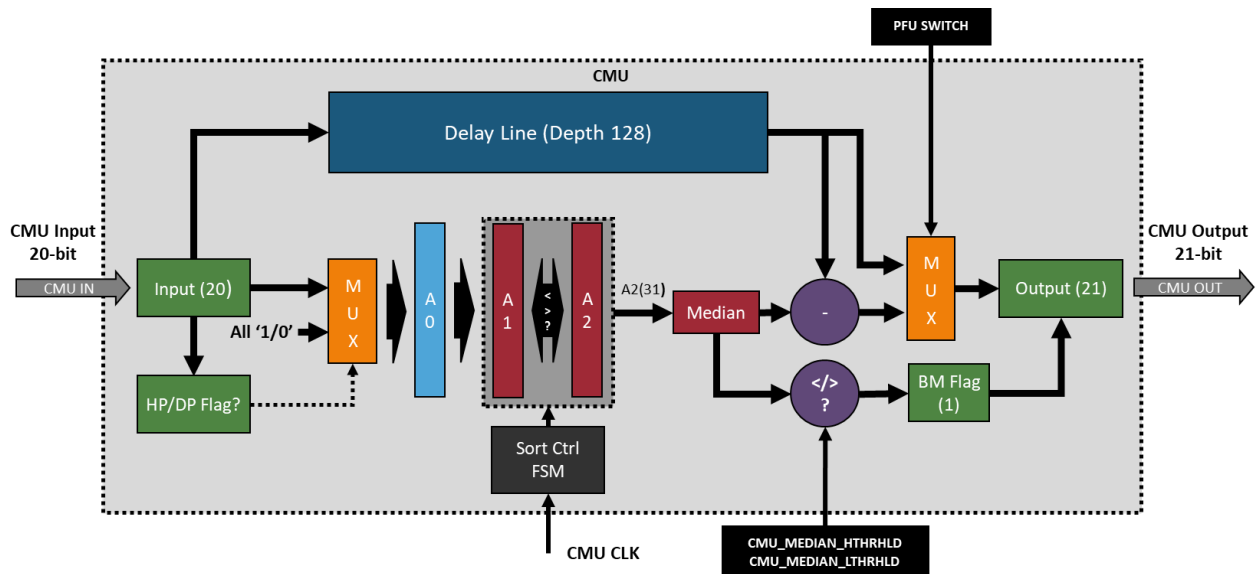
Once the offset has been subtracted and the HP and DP flags have been applied, the PCU applies two threshold comparisons to each pixel and adds a Low Threshold flag (LT) to pixels with values below the MisFit threshold and a High Threshold flag (HT) to pixels above the Minimum Ionizing Particle (MIP) threshold. MisFits occur when a particle hits a detector during readout, causing its readout energy to have “negative” values, while a MIP event occurs when a charged particle hits the sensor depositing a larger amount of energy than an X-ray photon would. It is important to note that the HP, DP, LT, and HT information are appended to the pixel ADU information as additional bits. Therefore, from the 15-bits at the PCU input (14-bit ADU + 1-bit subtraction bit), 4 additional bits are added, plus an extra Valid Data bit (VD) which is used to signal later pipeline units the beginning of the acquisition data stream, leading to a total output of 20-bits (Figure 35). A block diagram of the PCU is shown in Figure 34.



**Figure 35: Internal structure of the PCU input and output bits.**

#### 4.3.3.4 COMMON MODE UNIT

Following the PCU is the common mode unit (CMU), this unit oversees the correction of common mode noise. This noise arises mainly by disturbances such as electronic noise or voltage variations on the amplifying stages of the front-end electronics Veritas ASICs. Such disturbances cause variations in amplitude of the energy signals of all the pixels being read out at each given moment. Since the Veritas ASICs read out 64 pixels in parallel, the energy variation affects in a “common way” the 64 pixels’ energies, hence the name common mode. To correct for this effect, the CMU calculates the average energy offset for a row of pixels and subtracts this common mode value from the 64 pixels involved. Given that there is a good chance that a pixel in a row is hit by a photon skewing the average energy, the common mode value is calculated as the median energy rather than the average energy of the pixels in a row. To perform this task, the CMU accumulates the energy information of the 64 row pixels, sorts them by energy to obtain the median, and then subtracts the median to each pixel. A simplified schematic of the CMU is presented in Figure 36.



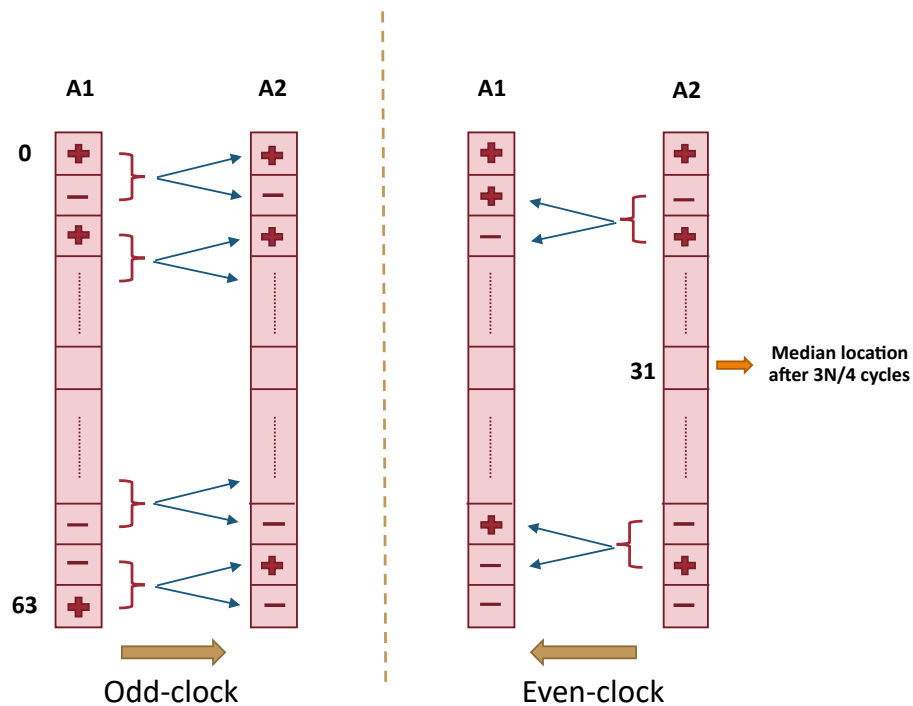
**Figure 36: Simplified block diagram of the CMU.**

At its input, the CMU receives 20-bits for each pixel. The information of each incoming pixel is relayed in parallel to a memory buffer that acts as a delay line and into an input array (A0), containing the 64 pixels of a row. Before storing the pixels' energy into A0, a check is performed whether the pixel has been flagged as a hot or dead pixel. In case this is true, the pixel energy is not directly stored but instead all their bits are set to either all '0' or all '1' in an alternating manner. This procedure aims to avoid distortions on the calculated median. Pixels with only zeros or ones are considered with minimum and maximum "energy values" and therefore the upcoming sorting FSM will automatically sort them out to the edges of the sorting arrays (A1 and A2), keeping the median centered on the array. The process of storing pixels of the same row together in the input array is synchronized by an external signal called the CMU CLK. This signal is generated at the central clocking unit of the FP and marks the beginning of a new row of pixels incoming into the CMU. When a CMU CLK is received, the CMU will start storing pixels into the A0 array from address 0 to 63. Once a second CMU CLK is received, the content of the full input array A0 is copied in a single transaction into a second array A1, where the sorting process begins. While the contents of row n are being sorted, the new row n + 1 is simultaneously being accumulated in A0, making both the storing and sorting processes 64 pixel-clocks in length, and therefore the delay line memory is 128 pixel-clocks deep.

To exploit the FPGA parallel processing capabilities, while still preserving the maximum performance possible, the following sorting algorithm is used. All pixels are initially copied in array A1 in parallel, then on the first pixel clock the contents of each consecutive address are compared in parallel with each other, comparing pixel 0 with

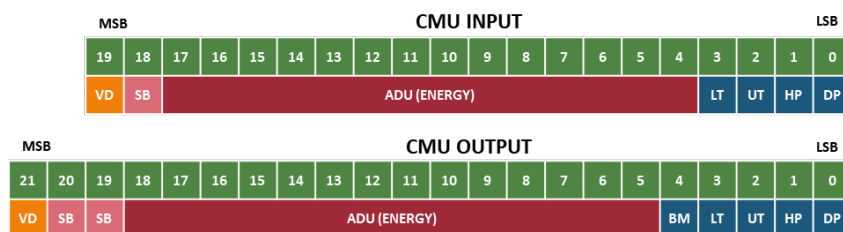


pixel 1, pixel 2 with 3 and so on until pixels 62 and 63 are reached. Then their contents are copied, with the bigger value copied on the lower corresponding address into array A2. On the next pixel-clock a similar process is repeated but, in this case, the consecutive pixels of A2 are compared with each other, skipping addresses 0 and 63 to create an offset which enables values to keep shifting, and the sorted results are copied back into array A1. These processes are repeated until all the pixels' energy values are sorted from high to low; odd clock numbers compare and copy values from A1 to A2 and on even clock numbers A2 is compared and copied into A1. Once the pixels are sorted, the median is located in address 31, the center of A2, its value is copied into a temporary register and this value then is subtracted from the energy of the pixels exiting the 128 pixel-clock delay line.



**Figure 37: Schematic depicting the sorting mechanism of the CMU. It should be noted that on even clock numbers the comparison is offset, skipping address 0 and thus allowing pixels to keep drifting on each comparison until they are sorted.**

If the median value falls below the CMU low median threshold or above the CMU high median threshold, given by the CMU\_MEDIAN\_LTHRLD and CMU\_MEDIAN\_HTHRLD registers, respectively, the pixels in the corresponding row are given a Bad Median (BM) flag, which is appended to the pixels' information as an additional bit. Making the CMU output 21-bit wide (see Figure 38).



**Figure 38: Internal structure of the CMU input and output bits.**

#### 4.3.3.5 EVENT FILTERING UNIT

The third unit on the EPP East is the EFU (a block diagram is shown in Figure 41), in charge of filtering pixels with enough energy to be considered to possibly contain X-ray events or part of an X-ray event.

The EFU adds two threshold flags to the pixels' information, the 1TH flag which indicates an valid event pixel, and the 2TH flag which indicates that the isolated pixel is not itself an event pixel but it could be part of an event pixel if it is next to a valid event pixel. These thresholds are set depending on each individual pixel noise level, as the intention is to be as efficient as possible in identifying event pixels, without accidentally promoting random noise events. Therefore, the EFU uses the Noise Look-up Table (NLT) where individual pixel thresholds are previously calculated and stored. Given that the PCU and CMU use subtraction bits to account for possible negative numbers, the two threshold registers read from the NLT, EFU\_1THRHLD and EFU\_2THRHLD, are added as a subtraction bit according to the PFU SWITCH register status. Once the threshold comparisons are applied, the EFU discards the subtraction bits from the pixels' information.

To prevent dead pixels from being wrongly promoted as event pixels on subsequent units, the TH1 and TH2 flags of dead pixels are set to '0'. Similarly, pixels marked as HPs are identified and their MIP flags are discarded to prevent the subsequent units from discarding neighboring pixels as MIP neighbors. The EFU input and output bits are shown in Figure 42.

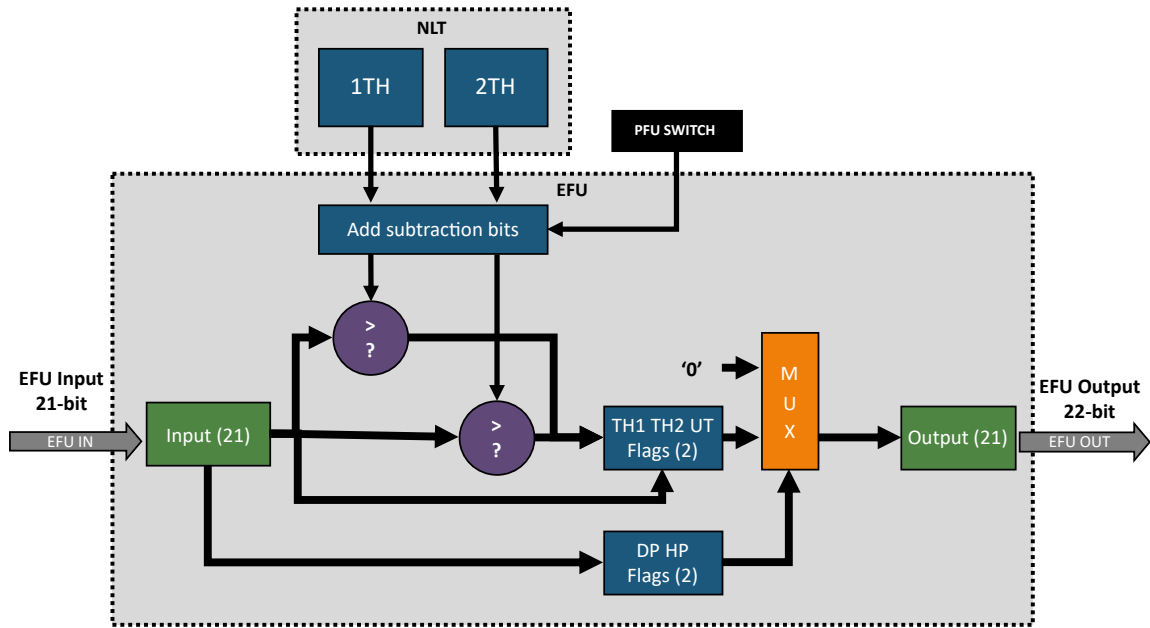


Figure 39: Simplified block diagram of the EFU.

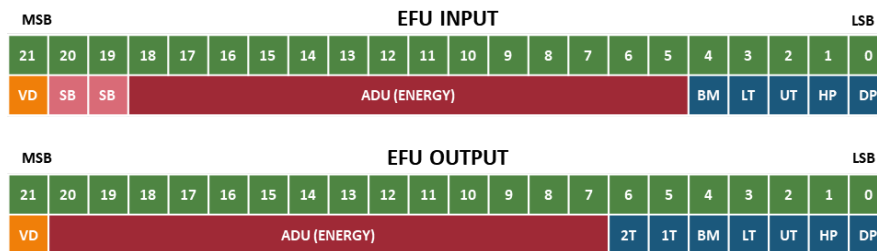
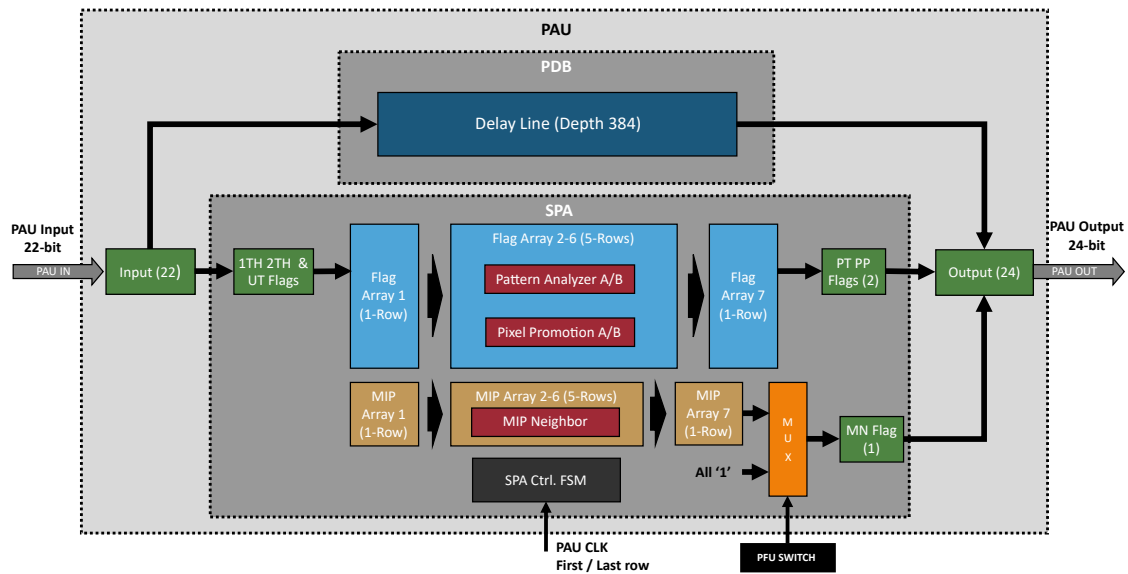


Figure 40: Internal structure of the EFU input and output bits.

#### 4.3.3.6 PATTERN ANALYZER UNIT

The pattern analyzer unit is the most complex and resource-consuming unit of the FP. As its name implies, it is in charge of analyzing the patterns created by the event pixels, neighbor event pixels, and MIP pixels and adding the corresponding flag information to each pixel.

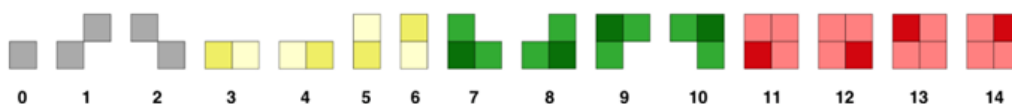
The PAU is subdivided into two sub-units: the Pattern Data Buffer (PDB) and the Slice Pattern Analyzer (SPA). The first unit acts simply as a delay buffer for the pixels' energy information, while the SPA performs the pattern analysis using only the pixels' flag information. At the end of the unit, both data streams get recombined and merged. Figure 41 shows a simplified block diagram of the PAU and its components.



**Figure 41: Simplified block diagram of the PAU.**

The PDB uses a 19-bit wide and 364-address depth RAM memory to temporally store each pixel's information for 384 pixel-clocks or 6 pixel-rows. This delay allows the SPA to perform the pattern analysis. The PDB uses the internal FPGA memory resources to make the storage of this information as efficient as possible, allowing the SPA more resources to process only the relevant information.

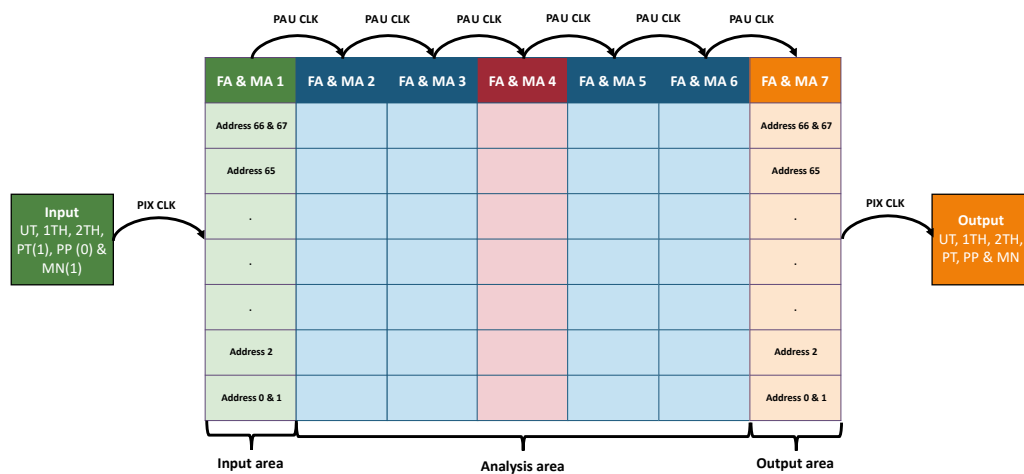
The SPA sub-unit is responsible for performing the analysis and flagging tasks of the PAU. Its operation can be subdivided into 3 main tasks. It performs a pattern analysis, flagging with a bad pattern flag (PT) groups of event pixels that do not fall within the expected point-like patterns generated by X-ray events (see Figure 42); it performs the pixel promotion of neighbor event pixels that are located next to a valid event pixel (PP); and finally, this unit reviews pixels that have previously been flagged as MIP events and marks as MIP neighbors (MN) two pixels around the MIP pixel.



**Figure 42: Pixel patterns corresponding to valid X-ray events allowed by the PAU.**

In order to perform these tasks, the SPA stores the 1TH, 2TH, and UT flags of seven pixel rows into two array groups. The first group contains 7 arrays known as the flag

arrays 1-7 (FA). Each array contains the UT, 1TH, 2TH, and two new flags for bad pattern, PT, and pixel promotion, PP, making each of the seven arrays 5-bit wide and 68 addresses deep (64 pixels per row + 4 extra addresses aiding the analyzer operation). The second array group, the MIP neighbor array (MA), is composed of seven arrays 2-bits wide. The two bits are used intercalating between frames to avoid propagating MIP neighbor information between two different frames. Therefore, the MAs are each 2-bit wide and 68 addresses deep. Pixels transit through the arrays in the following order: the pixels' UT, 1TH, and 2TH flags are stored into flag array 1, one pixel per pixel-clock consecutively as they arrive at the SPA input. Addresses 0, 1, 66, and 67 are prefilled with valid flags and skipped. In this way, the 64 pixels in a row are stored from address 2 to 65. When a PAU clock, signaling a new row arriving in the PAU, is received, the content of the full FA 1 is copied into the FA 2. In parallel, arrays 2 to 6 are copied into the arrays 3 to 7. The contents of FA 7 are read out sequentially on each pixel-clock and are mixed with the PDB output to constitute the PAU output. Similarly, the MIP array is prefilled when FA 1 is being filled. This array is completely prefilled since there are no pixels with MIP neighbor flag before this EPP stage, and on each PAU CLK, MA 1 to 6 are copied into arrays 2 to 7. A schematic of this procedure is shown in Figure 43.



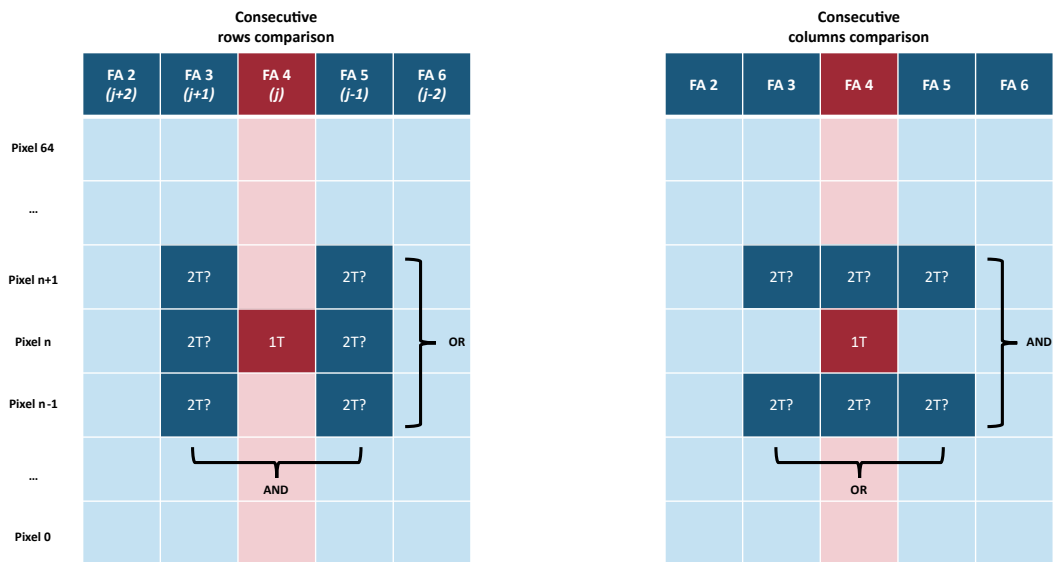
**Figure 43: Schematic depicting the clocking and moving of pixels inside the PAU arrays. Data enters on each pixel clock, once the first array is filled, the full arrays get copied to the following array synchronized with the PAU CLK.**

As the schematic shows, the analysis tasks are carried out using the information of FAs 2-6, with flag array 4 being the central element. It can also be observed that the arrays are carrying a pixel-row each; therefore, it can be said that if array 4 is the current row  $j$ , then arrays 5 and 6 are the previous rows  $j-1$  and  $j-2$ , while arrays 3 and 2 and the subsequent rows  $j+1$  and  $j+2$ , respectively. This array organization allows the PAU to

analyze two pixels around a central pixel in all directions: same row up and down and previous and subsequent rows, right and left.

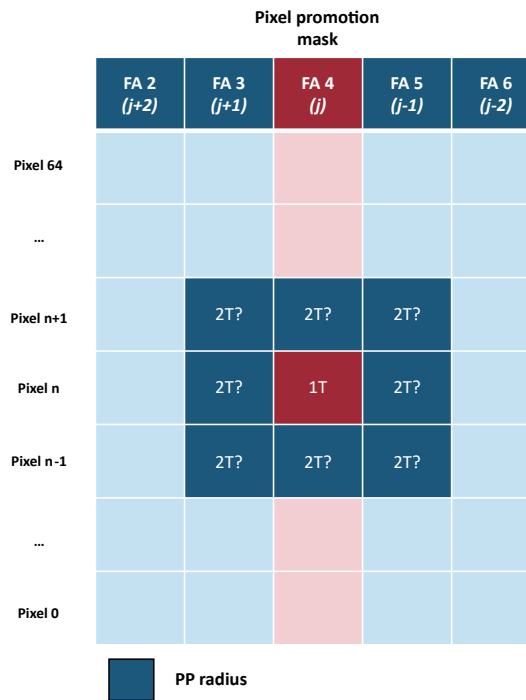
To perform the three main analysis tasks, three different algorithms are running in parallel, always with the flag array 4 in the center: pattern analysis, pixel promotion, and MIP neighbor analysis.

The pattern analysis, responsible for adding the bad pattern flag PT to all pixels that fall outside the allowed pattern shapes, works by using a comparison mask mechanism. This mechanism sits on top of the analyzed pixel on FA4. If the analyzed pixel is a valid event pixel (1TH = '1') and at least one pixel in row  $j+1$  and one in  $j-1$  are neighbor event pixels (2TH = '1') or at least one pixel in location  $n+1$  and one in  $n-1$  are neighbor event pixels, then those pixels are marked with the bad pattern flag (PT = '0'). Figure 44 shows a visual representation of the analysis mask.



**Figure 44: Comparison mask used by the pattern analysis unit to flag bad patterns, the masks translates vertically over FA4 performing the comparisons on each pixel.**

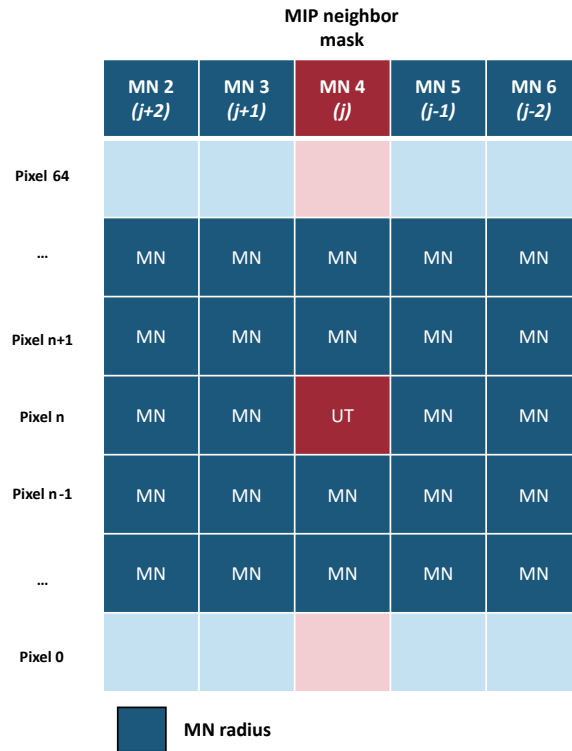
A similar procedure takes place in the pixel promotion analyzer, although in this case the algorithm is simpler. The analyzer mask sits on the central pixel. If this pixel is marked with a valid event flag (1TH = '1'), then any pixel around it is promoted with a valid pixel promotion flag (PP = '1') if they have a valid neighbor event flag (2TH = '1').



**Figure 45: Schematic showing the pixel promotion mask radius.**

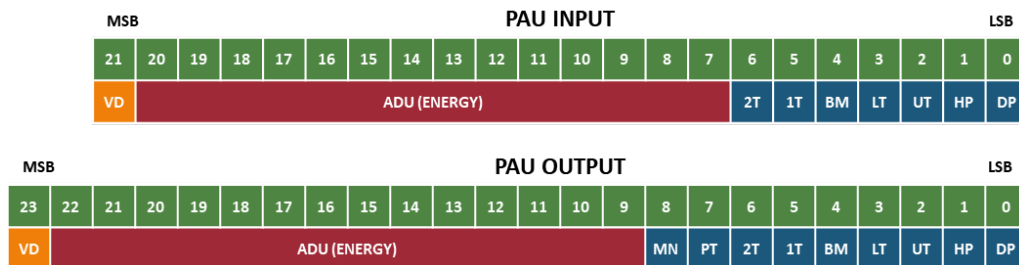
In order to ease the timing constraints on the FPGA, two pattern analyzer and pixel promotion masks are run in parallel, both covering 32 pixels, each covering pixel 0-31 and 32-63, leaving two pixel-clocks for the analysis of each pixel. Both analyses use the first row and last row signals to limit the analysis to the concerned frame. When the last row signal is active, the analysis only takes care of FA4 and FA5. Similarly, when the first row is active, the analysis masks only consider FA4 and FA3.

The final analysis in the PAU is the MIP neighbor analyzer. This procedure again uses the information stored in the FA to perform the MN flagging. Specifically, the unit uses the UT flag to identify MIP events. Once the mask is centered over a MIP event, two pixels around the MIP event pixel will be flagged as MIP neighbors (MN = '0'). To avoid the MN flag propagating between pixel-rows belonging to different frames, the MN flag is stored in one of the two bits of the MA, alternating between bit 0 and bit 1 on each frame. Simultaneously, the output of the PAU is synchronized to copy the right bit corresponding to the output frame when preparing the PAU output data.



**Figure 46: Schematic showing the MIP neighbor mask radius.**

At the PAU output, data of the SPA and PDB units is merged into a single data stream. The MN flag is activated or deactivated depending on the PFU SWITCH register state. The PP flag gets merged into the 1TH flag, and the MN and PT flags get appended to the pixels' information, making the PAU output 24-bits wide, see Figure 47.



**Figure 47: Internal structure of the EFU input and output bits.**



#### 4.3.3.7 PIXEL FILTERING UNIT

The last unit on the EPP East is the PFU. This unit oversees the final filtering, selecting which pixels are going to be packed and transmitted. Additionally, the PFU adds the frame and pixel number information to each pixel.

The FP is a completely synchronous design, and none of the pixels' information gets discarded until the end of the EPP. The selection on which pixels are going to be flagged as valid pixels (VD flag) is taken in the PFU, where all the flags appended to the pixels' information are compared against the PFU MASK register, which contains the flag mask criteria that a pixel needs to meet in order to be considered a valid pixel (VD='1'). The PFU MASK can be bypassed using the PFU BYPASS register, which enables the bypassing of a certain flag or flags so that they are not considered for the valid pixel selection, see Figure 48. Prior to these comparisons, the PFU unit checks for the FPU SWITCH register to activate/deactivate the "raw data mode," which, as its name implies, allows for all the FP data to be flagged as valid pixels for their transmission. It is this unit that makes use of the VD added to the pixels' information on the PCU. Without this flag, the FP would bypass as raw data any invalid information if the sequencer were actively sending pixel-clock signals.

| PFU MASK |    |    |    |    |    |    |   |    |    |    |    |    |    |    |    |    |
|----------|----|----|----|----|----|----|---|----|----|----|----|----|----|----|----|----|
| BIT      | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FLAG     | X  | X  | X  | X  | X  | X  | X | MN | PT | 2T | 1T | BM | LT | UT | HP | DP |

| PFU BYPASS |    |    |    |    |    |    |   |    |    |    |    |    |    |    |    |    |
|------------|----|----|----|----|----|----|---|----|----|----|----|----|----|----|----|----|
| BIT        | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| FLAG       | X  | X  | X  | X  | X  | X  | X | MN | PT | 2T | 1T | BM | LT | UT | HP | DP |

**Figure 48: Internal structure of the PFU MASK and PFU BYPASS registers bits.**

A simplified block diagram of the PFU is shown in Figure 49. The pixel number and frame number are generated in the FP central clocking unit and received at the PFU entry, synchronized with the arrival of the corresponding pixels. It should be noted that the pixel number is 15-bit wide, while the FD only requires 12-bits to account for the 4096 pixels of a frame. However, the 15-bit scheme was kept, maintaining the same data format as the LD-FP, which does require 15-bits for the maximum number of pixels on each slide per frame. The PFU input and output bits are illustrated in Figure 50.

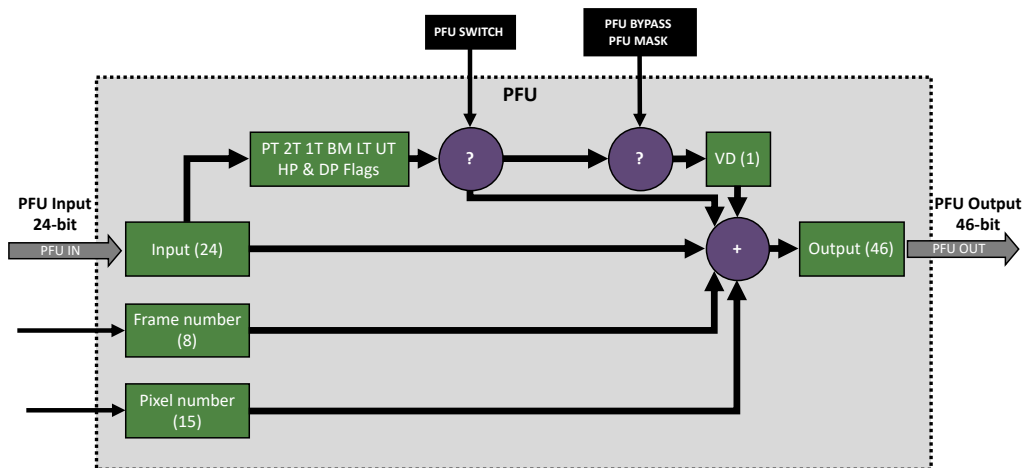


Figure 49: Simplified block diagram of the PFU.

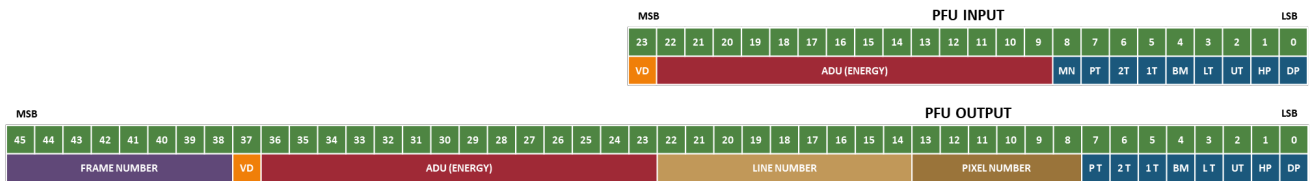
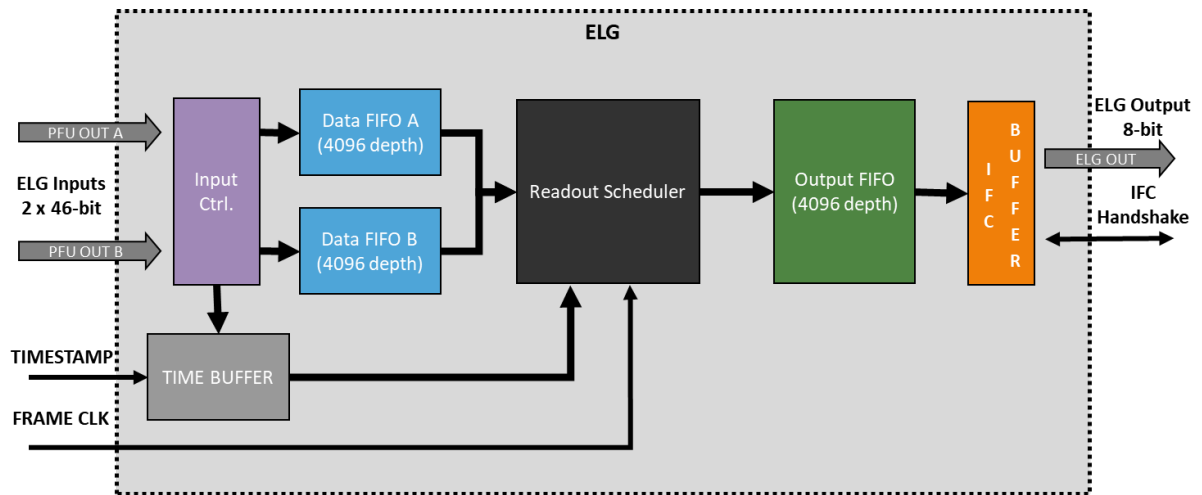


Figure 50: Internal structure of the PFU input and output bits.

#### 4.3.4 EVENT LIST GENERATOR

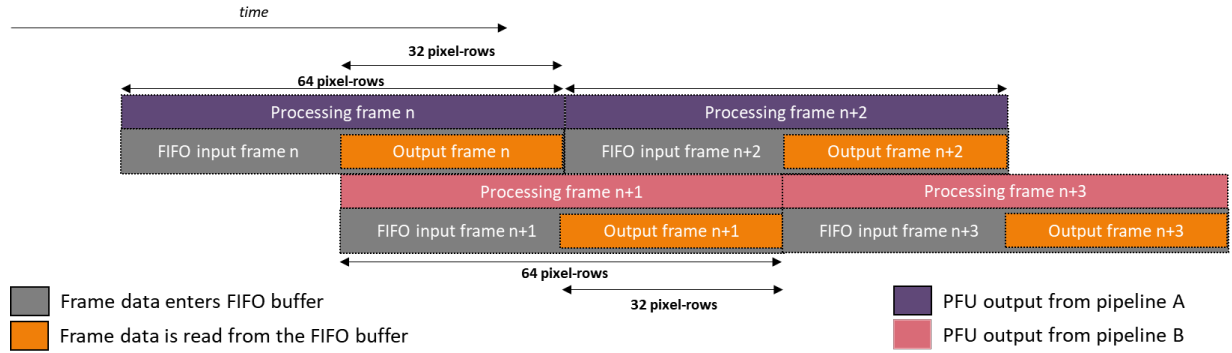
The ELG is the unit in charge of packing the event information on a frame-by-frame basis and communicating the information to the Interface Controller (IFC) for its transmission.

This unit contains an input controller which stores the information of any arriving pixel with a valid event flag (VD = '1') into its corresponding data First-In First-Out (FIFO) buffer, one buffer for each pipeline (DFA and DFB). At the same time, the input controller checks the frame number of the valid pixel, and if the pixel belongs to a new frame number, saves the timestamp information and frame number into a buffer register, the time buffer.



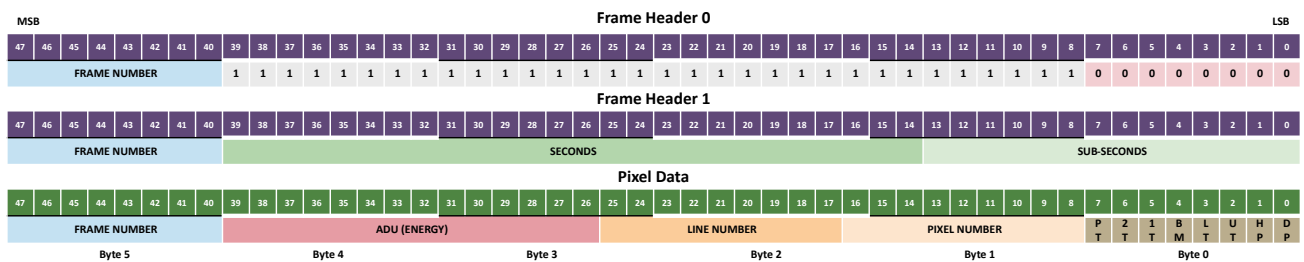
**Figure 51: Simplified block diagram of the ELG.**

As seen in Figure 51, an FSM called the readout scheduler reads in a synchronous manner the contents of the two data FIFOs, and places the frame data into a third FIFO, the output buffer. To output the data in a frame-wise order, the readout scheduler assigns 32 pixel-row times to each pipeline according to a frame clock signal generated from the central clocking unit. Before writing pixel data, the readout scheduler checks for the time buffer comparing the current pixel frame number with the time buffer, and if the pixel corresponds to a new frame, writes the frame header into the output FIFO, followed by the information of all the pixels belonging to the frame. It should be noted that the readout scheduler performs the DFs readout at a faster rate than they are written, taking less than 32 pixel-rows time to perform the readout. This ensures that the readout scheduler is always able to readout a full frame from the DFs before the new frame data needs to be read out. A schematic depicting this process is shown in Figure 52 where it can be appreciated that while the readout of the PFU takes 64 pixel-row times, the readout of the DFs only takes 32 pixel-row times. It should also be noted that outputs of the PFU are synchronous and therefore an exact time allocation can be safely be given to the readout of each of the two DFs. Therefore the readout of the DFs of each pipeline only begins when the information of the first half of the frame has already been stored in the corresponding DF.



**Figure 52: Timeline depicting the order in which data is fetched by the readout scheduler FSM.**

The output FIFO memory has a depth of 4096 addresses, making it capable of storing a complete FD frame. This buffer acts as an interface between the DFs and the Interface Controller Buffer, which is in charge of reading the output FIFO and transmitting, in an 8-bit format, the output FIFO information as requested by the IFC. Given that the word information is composed of 48 bits, the IFC buffer divides the contents of each output FIFO word into 6 chunks of 8 bits. Once the IFC has read out the six chunks corresponding to a word, the IFC buffer automatically loads the next pixel information from the output FIFO. The schematic in Figure 53 shows the structure of the 48-bit output FIFO word, both for frame headers and pixel information. Both the pixel and line number require fewer bits to comply with the FD data requirements, but the additional bits are kept mirroring the LD-FP data format.

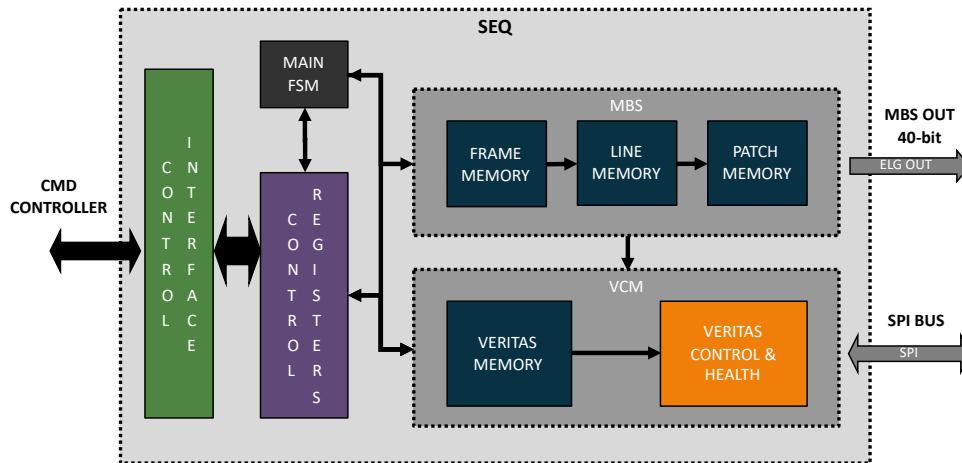


**Figure 53: Data format of the Event Lists and the internal bit structure of each word.**

#### 4.3.5 SEQUENCER UNIT

The Wide Field Imager (WFI) FP firmware is responsible for programming, driving, and sequencing the WFI FEE Switcher and Veritas ASICs. These tasks are performed inside the FP in the sequencer unit. This unit outputs a set of periodic switching signals

activating and deactivating at periodic intervals to drive the FEE ASICs and contains an SPI interface which can program and monitor the Veritas ASICs during operation. The architecture of the sequencer unit is depicted in the simplified block diagram in Figure 54.



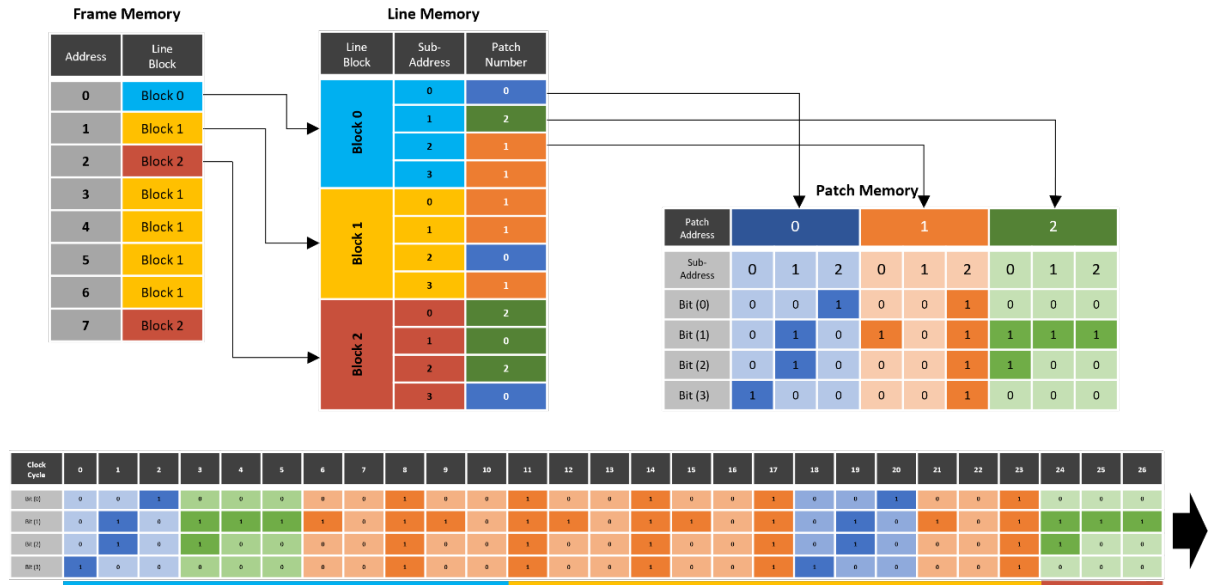
**Figure 54: Simplified block diagram of the sequencer unit.**

The unit is divided into three modules: the sequencer control module, memory-based sequencer (MBS), and the Veritas control module. The control module is in charge, as its name suggests, of controlling the sequencer unit and interacting with the command controller of the FP to transmit and receive information outside of the sequencer unit. It is subdivided into three blocks: the control interface, receiving and transmitting commands to the command controller, the main FSM, and control registers which govern the operation of the sequencer. The MBS generates the sequencers that drive the FEE; it mainly consists of three memory blocks which contain frame-sequence, line-sequences, and patch-sequences. Finally, the Veritas control module drives the SPI communications with the Veritas ASICs and monitors the correct operations of the ASICs.

#### 4.3.5.1 MEMORY BASED SEQUENCER

The MBS primarily comprises three memory blocks that interact in a manner similar to how pointers function in programming languages. The first memory, the frame memory, addresses different locations in the line memory. Each address corresponds to one of up to 16 different line sequences available for output. These sequences represent the readout sequence for one or more rows of pixels in the LAD or FD. By sequentially outputting different line sequences in the order specified by the frame memory, the entire sequence to read a complete frame can be constructed. Similarly,

the line memory addresses locations in the patch memory, which in turn store addresses pointing to different sequence-patches in the patch memory. The patch memory contains recurring small sequence patches, each containing up to 16 system clock cycles of sequence information. These patches serve as the fundamental building blocks for the full frame sequence. Figure 55 below presents a simplified schematic illustrating how the FP MBS operates.



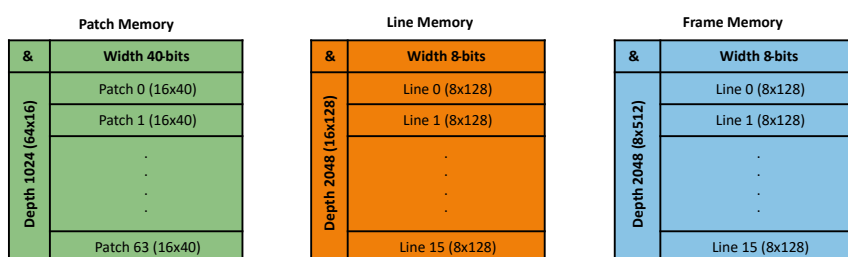
**Figure 55: Simplified schematic of the memory base sequencer, generating the first 27 clock cycles of a 4-bit width bus.**

The frame memory is an 8-bit wide, 512-bit depth LSRAM. Each address stores the address of a line memory sequence that is output sequentially to form a complete frame sequence. The number of lines required to generate a complete frame is programmed using the FRAME SIZE register. When the sequencer is in working mode, the addresses of this memory are read sequentially in ascending order, outputting the line sequences specified by the information stored at each address. Once the count reaches the maximum address, specified by the FRAME SIZE register, the sequence returns to its first address, which is zero, and starts counting again. Each cycle of the entire memory corresponds to a complete sequencer frame.

The line memory is an 8-bit wide, 2048-bit depth LSRAM. It is divided into 16 blocks, each with 128 sub-addresses. Each block stores a different sequence of patch addresses to generate line sequences. Address 0 of the line memory is reserved for the initialization sequence, which is output only once at the beginning of the sequencer operation but is skipped when continuous frame outputs are ongoing. When the frame memory selects an address from the line memory, the corresponding block is read out sequentially, starting from its lower sub-address and moving up to the maximum

address specified by the LINE SIZE register. Once the maximum address is reached, the frame memory moves to a new block, and the process repeats. The maximum address of block 0 is given by the INIT SIZE register.

The patch memory is a 40-bit wide, 1024-bit depth LSRAM. It is divided into 64 blocks of 16 bits each. Each block contains a patch that is sequentially output based on the line memory. Just like the frame memory addresses the line memory, the line memory addresses the patch memory to form a line sequence. When the line memory points to a block address, the corresponding patch sub-addresses are output one by one until the maximum block address is reached. The maximum sub-address for the patch memory blocks is determined by the PATCH SIZE register, see Appendix A: FD Interface Controller Operation for more information on the mentioned internal registers operation. A summary of the three memory sizes and divisions is shown in Figure 56.



**Figure 56: Simplified block diagrams with the sequencer memories internal divisions.**

The line memory's different line sequences are as described in Table 3. Columns 2 and 3 show the maximum line length in units of patches and clock cycles that the unit allows (a patch can be up to 16 clock cycles). If the FP operates with a clock speed of 80 MHz, the maximum line length is calculated as:  $(12.5 \text{ ns/clock}) * (2048 \text{ clocks/line}) = 25.6 \text{ us/line sequence}$ .

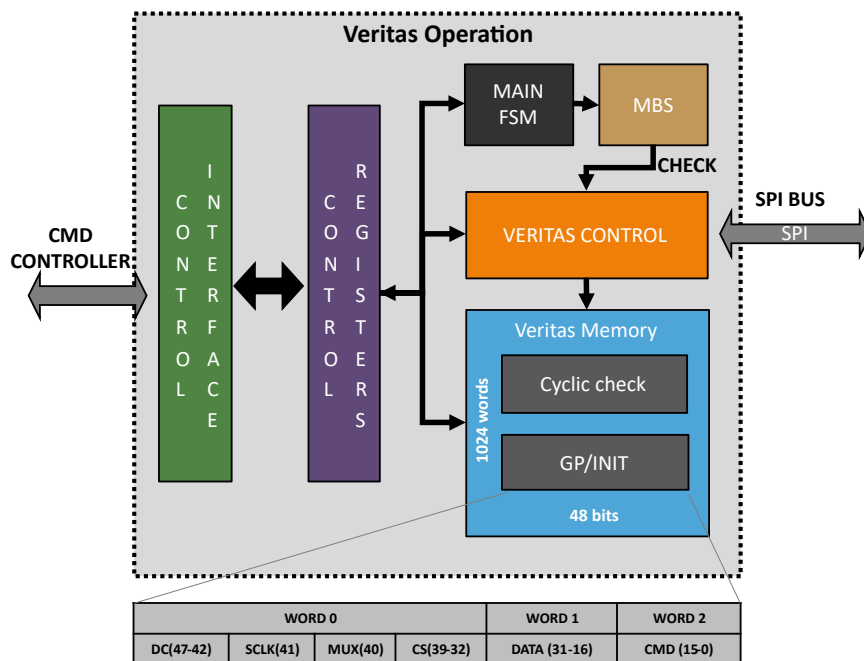
| Line             | Line length (patches) | Line length(clocks) |
|------------------|-----------------------|---------------------|
| Initial Sequence | 128                   | 2048                |
| Line 1           | 128                   | 2048                |
| Line 2           | 128                   | 2048                |
| Line 3           | 128                   | 2048                |
| Line 4           | 128                   | 2048                |
| Line 5           | 128                   | 2048                |
| Line 6           | 128                   | 2048                |
| Line 7           | 128                   | 2048                |
| Line 8           | 128                   | 2048                |
| Line 9           | 128                   | 2048                |
| Line 10          | 128                   | 2048                |
| Line 11          | 128                   | 2048                |
| Line 12          | 128                   | 2048                |
| Line 13          | 128                   | 2048                |
| Line 14          | 128                   | 2048                |
| Line 15          | 128                   | 2048                |

**Table 3: Line memory internal address mapping.**

#### 4.3.5.2 VERITAS CONTROL MODULE

The Veritas control module manages the programming and reading of Veritas ASIC internal registers through an SPI interface. This interface is divided into two primary sub-modules: the Veritas memory and the Veritas control module (shown in Figure 57). The Veritas memory houses all the necessary information required to define and execute SPI operations, which can include SPI write, read, or read-write operations. The control module reads the data from the Veritas memory and executes the corresponding SPI signals by driving and receiving the appropriate signals.



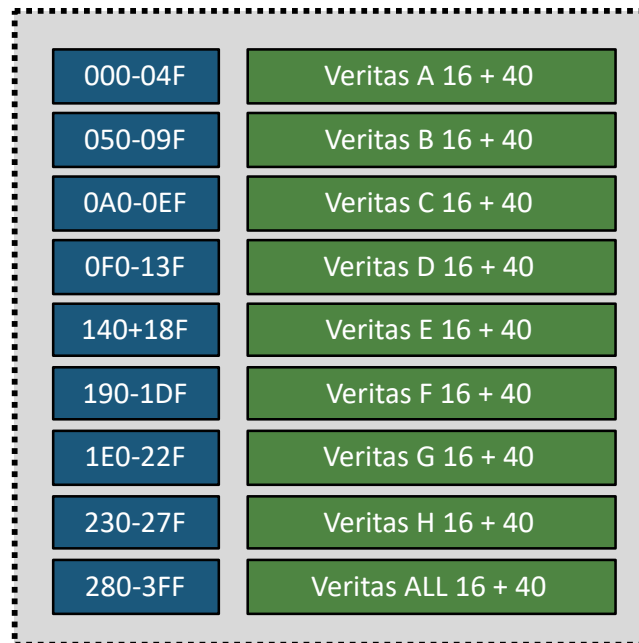


**Figure 57: Simplified block diagram of the veritas control module.**

The Veritas memory enables the user to store up to 1024 48-bit command words. These memory addresses are divided into two primary sections. The first section, the cyclic check section, is used to store commands that will be activated during the FP operation. It also monitors the state of the Veritas internal memories. The second section, the initialization section, stores a series of words that can be directly programmed into a specific Veritas register or used to initialize all the Veritas registers before the FP operation begins. During operation, the interface automatically uses the cyclic check commands to monitor the Veritas ASICs, while the initialization section is used during programming mode to set the Veritas registers before the FP transitions into operation mode.

The cyclic check section of the memory is divided into eight sections, each containing information about up-to eight Veritas ASICs that can be connected to the interface (the FD implementation only uses the first two). This allows the user to program the register with different information for each ASIC and independently monitor each one. A diagram illustrating the memory address organization is provided in Figure 58. Please note that each word is 48-bit wide, while the main FP command interface can only handle 16-bit data words. This means that reading or modifying a single word of the Veritas memory requires three read/write operations on the FP command interface. For

instance, to write the address 0 of the Veritas memory, addresses 3000h, 3001h, and 3002h would need to be written via the FP command interface.



**Figure 58: Word addresses organization inside the veritas memory. The fast detector implementation only uses the A and B blocks of the cyclic check memory allocation.**

As mentioned earlier, the Veritas control submodule utilizes the Veritas memory to perform three distinct operations: Veritas memory check, Veritas memory initialization, and Veritas memory direct commands. The first operation, Veritas memory check, is executed automatically and sequentially during FP operation whenever the check-signal (MBS bit 38) transitions to logic '1'. The number of checks per ASIC that can be performed and the number of ASICs to be checked are programmable using the sequencer unit control registers (see section Command Interface for more information on the registers operation). The Veritas memory initialization command is executed when a Veritas command is addressed to the last memory word (word 3FFh). Lastly, direct commands are executed when a word within the initialization memory range (280h-3FFh) is directly addressed for execution.

#### 4.3.6 LOOK-UP TABLE GENERATOR

The look-up table generator is the ancillary unit responsible for calculating and storing the corresponding offset and noise values for each pixel in the FD sensor that are used on the PCU and EFU.

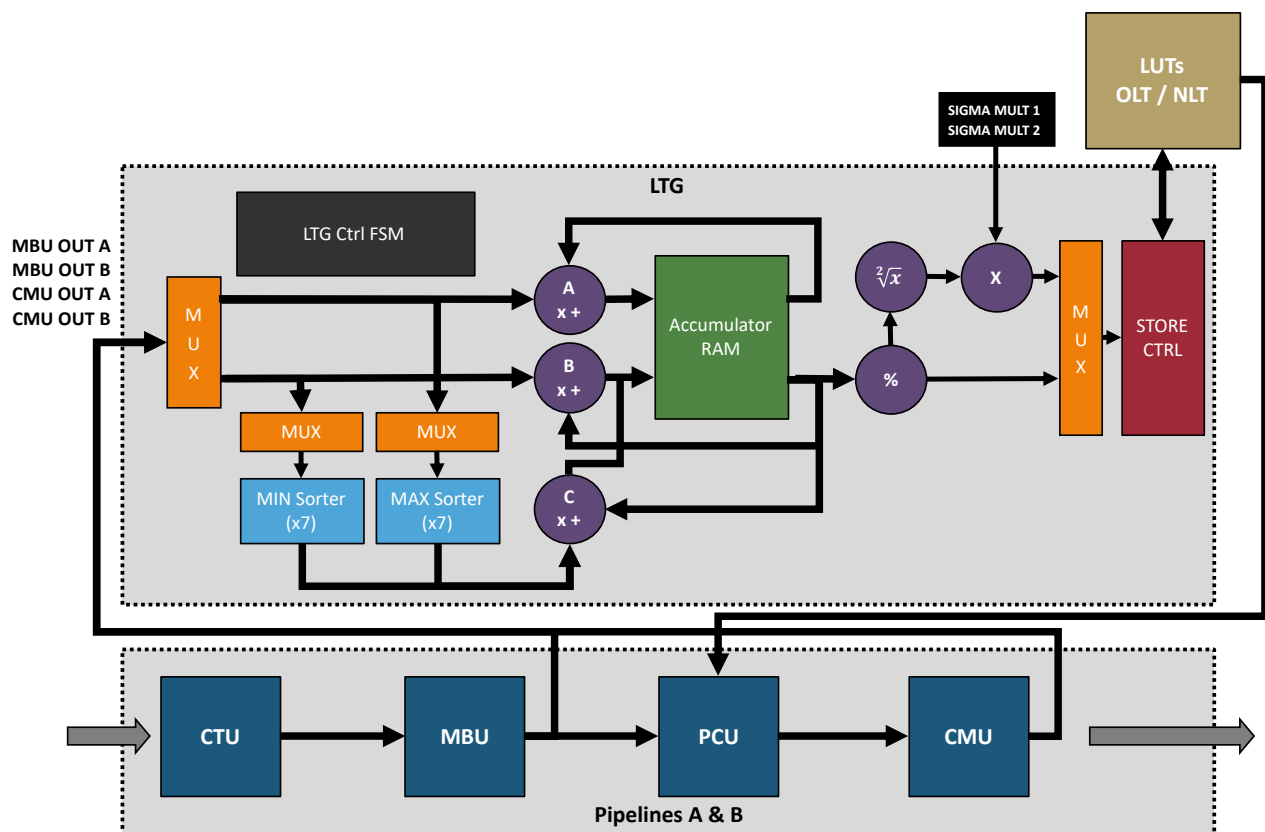
The OLT contains the ADU offset values for each pixel; this offset is simply calculated as the average energy for each pixel over a certain number of frames, in our case 256 frames. On the other hand, the NLT contains the noise value for each pixel; these noise values are calculated as the standard deviation on each pixel's energy. Once the standard deviation is calculated, it gets multiplied by two programmable multiplication factors, given by the LTP SIGMA1 and LTP SIGMA2 registers, to calculate each pixel's event and neighbor thresholds. The NLT is also performed using 256 frames. The number of frames chosen for the calculation is a trade-off between better statistics, FPGA resources, and calculation time based on the sensor's performance stability over several frames. The equations below show the mathematical representation for the calculation of the OLT and NLT maps, where  $\epsilon_1$  and  $\epsilon_2$  are the 1TH and 2TH multiplication factors.

|                                                         |                                                                               |
|---------------------------------------------------------|-------------------------------------------------------------------------------|
| OLT                                                     | NLT                                                                           |
| $\mu = \frac{1}{n} \left[ \sum_{i=1}^{i=n} x_i \right]$ | $\sigma = \sqrt{\frac{1}{n-1} \left[ \sum_{i=1}^{i=n} (x_i - \mu)^2 \right]}$ |
|                                                         | $1T = \sigma \times \epsilon_1 \quad 2T = \sigma \times \epsilon_2$           |

To perform the OLT and NLT calculations, the LTG needs to account for the intercalating frames between the two preprocessing pipelines while attempting to reuse FPGA resources whenever possible, allowing higher performance. A simplified block diagram of the LTG is shown in Figure 59.

The LTG has two data access points, one used for the OLT calculation and the other used for the NLT. The first input gets its data directly from the MBU output, just after the frame pixels have been sorted, while the NLT calculation input uses the CMU output. Each input fetches data from the two parallel pipelines in parallel, meaning that data from the two data streams correspond to pixel addresses offset by half a frame, 2048 addresses for full frame mode, and 1024 addresses for window mode. Data is accumulated in a central accumulation memory, the accumulation RAM, capable of storing the accumulated data for up to 4096 pixels composing a FD sensor frame. Parallel acquisition is achieved thanks to the true dual-port capabilities of the FPGAs' internal memories, and therefore the two access ports of the accumulation RAM are used independently by the two data streams. A central FSM, the LTG control FSM, handles the synchronization of the LTG operation, controlling the addresses, pixel count, and cycle counts for the data accumulation. In parallel to the data acquisition, the LTG keeps track of the four minimum and four maximum values of energy received

by each pixel. This process is used to remove pixel data outliers from the accumulated data and requires a sorting mechanism that compares the new incoming data with the previously stowed information. Given that the sorting mechanism requires several system clock cycles, a parallelization approach was used. Therefore, the MIN/MAX sorting units in charge of keeping track of the 4 minimum and 4 maximum values for each pixel are heavily parallelized. Eight instances of each unit are used. A multiplexer at their inputs directs the incoming pixels, alternating consecutive pixels into the different sorting units, cycling back every eight pixels. This allows for each unit, eight pixel-clock cycles or twenty-four system clocks, to perform the sorting and comparing functions. The sorting units also make use of the FPGAs' RAM memories, multiport capabilities, and therefore each unit supports input on two parallel addresses per operation. Accumulation, subtraction, and multiplication operations are performed using the FPGAs' MACC cores, which are specific IP cores dedicated to such arithmetic operations. Storage of the calculation results into the OLT and NLT is handled by the store control FSM, a simple FSM that performs the read/write operations on the desired addresses of the LUTs.



**Figure 59: Simplified block diagram of the LTG architecture.**

---

#### 4.3.6.1 OLT CALCULATION

The OLT calculation procedure is briefly described in this sub-section. When the OLT generation command is received, the process begins by initializing the internal memories of the LTG module. This includes resetting the accumulator memory to zero and setting the MIN/MAX sorters memories to their initial values. This sets all bits as zeros for the MAX sorter and ones for the MIN sorter. This ensures that in the MAX sorter any incoming pixel will definitely be bigger and the opposite applies for the MIN sorter. Following initialization, the LTG aligns its internal counters to the FP data using the PCU CLK signal. After the system is synchronized, the acquisition phase begins. During this phase, two parallel processes are carried out. Pixel data will be read from the accumulator RAM, and the new incoming corresponding pixel data will be added to the accumulated sum. While data is accumulated, the MIN/MAX sorters in parallel keep track of the 4 maximum and 4 minimum energy values. The process of acquisition is repeated for 264 frames. These are 256 frames of data that need to be accumulated plus 8 extra frames of the 4 minimum and 4 maximum value data frames that are going to be removed to avoid data skew. When the accumulation phase is over, the system stops acquisition and starts the subtraction phase. During this phase, the accumulator energy address for each pixel is read, and the corresponding 8 min/max values are subtracted. Once the final accumulated value is acquired, the preparation and storage of each pixel's offset take place. During preparation, the accumulated energy value for each pixel gets divided by 256, which was conveniently chosen as a division factor as it corresponds to an 8-bit right-shift on the accumulation register output. The resulting value is the mean energy for the corresponding pixel. The storage controller reads the corresponding address on the OLT and retrieves the DP and HP flags, appends this information on the MSB, and stores back the flags with the new calculated offset. The preparation and storing process gets repeated 4096 times for all the FD pixels.

---

#### 4.3.6.2 NLT CALCULATION

The NLT calculation procedure is similar to the OLT procedure and reuses most of the calculation units. The process begins also with the initialization process where all memories get reset/set to their initial values. After initialization is completed, the synchronization process begins; in this case, the NTG CLK signal is used for synchronization since the NLT calculation takes the data coming from the CMU instead of the PCU. Data is taken from the CMU to account for the mean subtraction term on the standard deviation calculation formula ( $x_i - \mu$ ), the PCU is effectively subtracting the mean from each pixel's energy data. Data gets squared simply by duplicating the register and multiplying it and gets accumulated in the accumulation RAM. In parallel as well, the MIN/MAX sorters keep track of the outlier values. Once the accumulation

phase is over, the subtraction of min/max values takes place; in this case, the min/max values get first squared and are then subtracted. In the NLT calculation, data is accumulated for 257 frames; this accounts for an extra term that would allow to perform the  $(n-1)$  division of the standard deviation formula using a bit shift as in the case of the OLT. After subtraction, the data preparation and storage take place. Data preparation has two additional steps compared to the OLT calculation. After dividing the corresponding pixel accumulation value by 256, the square root is calculated. The algorithm uses a brute force approach where a number  $x$  is squared by itself, increasing its value by 1 on each step, and then gets compared to the reference value; the last integer number that does not exceed the reference value is the square root. The square-rooted value is stored, multiplied by the two sigma multiplication factors, obtaining the event and neighbor event thresholds, and finally, a store control FSM saves the corresponding information on each of the NLT addresses. The preparation and storage processes get repeated 4096 times until all the pixels are covered.

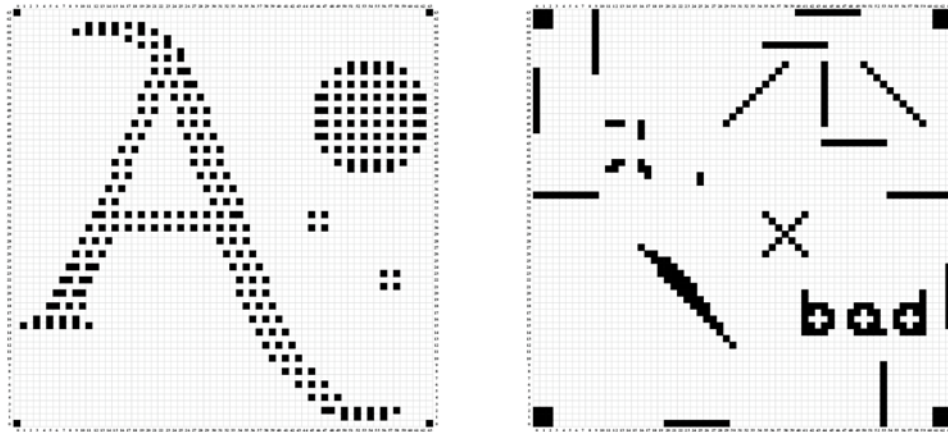
It is worth noting that the OLT and NLT generation procedures are performed using only internal FPGA memory resources, accumulating data from consecutive frames, unlike in the LTG of the LD-FP. This difference means that the calculation of both memory maps can be performed in less than a second, marking a milestone in performance considering the number of calculations performed.

---

#### 4.3.7 DIAGNOSTIC UNIT

The diagnostic unit (DIAG) is an ancillary system that allows testing the FP functionalities without the need for a sensor, by outputting test “frame data”, that is processed by the EPP.

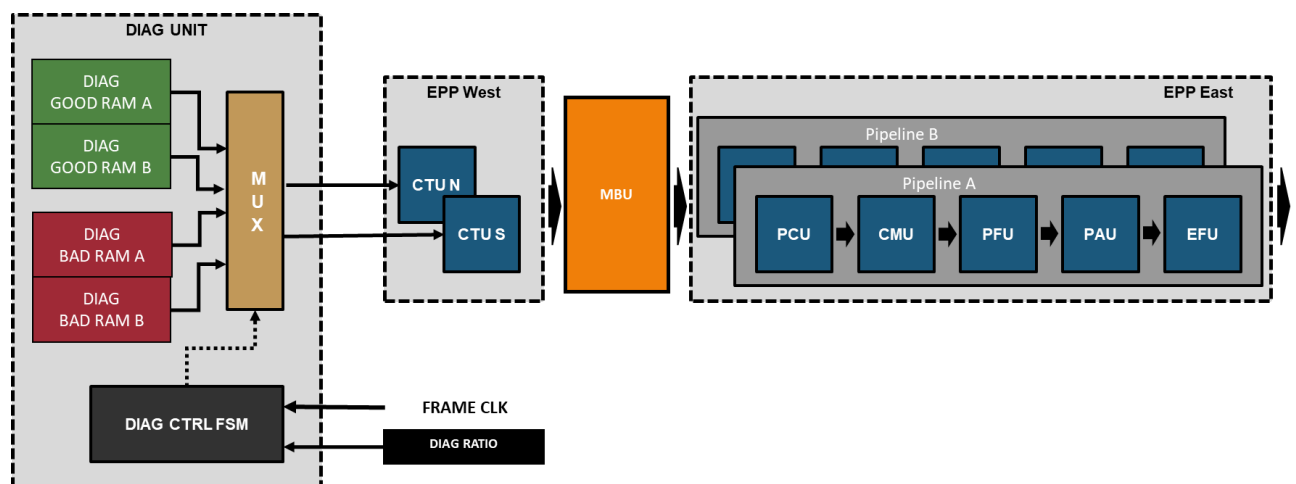
The unit uses four RAM memories, 16-bit wide, 2048 pixels deep, storing each half a FD frame. Therefore, two memories are used to compose a complete frame, allowing the DIAG system to store two complete FD frames, from here on known as GOOD RAM and BAD RAM memories. These memories work together in an intercalated manner, with a configurable ratio given by the DIAG RATIO register, which indicates how many times the BAD RAM memory is output per GOOD RAM cycle. The main intention of this separation, apart from having two separate test frames, is to limit the data output at the FP exit, which could generate data overflows. Hence, GOOD RAM is used to store data with “valid events” while BAD RAM data is filled with non-valid or discarded events. By increasing the DIAG RATIO, the data output of the FP can be lowered, enabling the SpW interface to avoid data overflow. Figure 60 shows a GOOD and BAD frame example.



**Figure 60: GOOD FRAME (left) and BAD FRAME (right) default test examples.**

As mentioned before, both the GOOD RAM and BAD RAM are composed of two separate memories: GOOD RAM A/B and BAD RAM A/B, respectively. Each half is used to output in parallel the “sensor data” into the two CTU units, emulating the accusation of the real sensor. With the A half corresponding to input of CTU N, and B with CTU S. A multiplexer at the output of the DIAG unit takes care of alternating and sending the corresponding frame-half to the correct CTU.

The input of data in the FP should be synchronized with the pixel and frame clocks, just as with the real FD sensor. Also, the count between GOOD/BAD data frames should be tracked to output the correct ratio. To oversee this process, a simple FSM is the main controller of the DIAG unit. A block diagram showing the overall architecture of the DIAG unit is shown in Figure 61.



**Figure 61: Simplified block diagram of the diagnostic unit.**

#### 4.3.8 HOUSEKEEPING UNIT

The HKU specifications are still to be defined. This unit is in charge of ensuring a smooth and efficient operation of the FP, monitoring and reporting critical operation data such as currents and voltages via the IFC. Given that many aspects of the final HKU are yet to be defined, a versatile design that would allow add-on developments was developed.

The HKU is composed mainly of a memory that contains the relevant register data, an access controller which is in charge of reporting the memory contents to the IFC for its transmission, and another FSM called the memory update unit that operates on loading register contents into the HKU memory. The memory exploits the true dual-port capabilities of the RTG4 LSRAM blocks and serves as an asynchronous interface between the IFC and the various registers linked to the memory update unit as well as the Power Conditioning Module (PCM) interface. This critical capability allows it to develop behind the asynchronous barrier without affecting any functionality of the IFC interface, allowing for future upgrades and developments.

As mentioned before, a critical feature of the HKU is that it serves as an interface with the PCM and therefore this unit oversees powering up and down the power supplies for the CHS. The PCM interface is envisioned to consist of a simple UART serial interface, but its details remain to be concluded. Furthermore, the HKU registers should include interfaces capable of reading out supervising ADCs that will monitor the temperatures, voltages, and currents of the FPM electronics, as a baseline is considered that the interface most probably will require an SPI interface. A simplified block diagram of the HKU is shown in Figure 62, the dotted line denotes the asynchronous barrier.

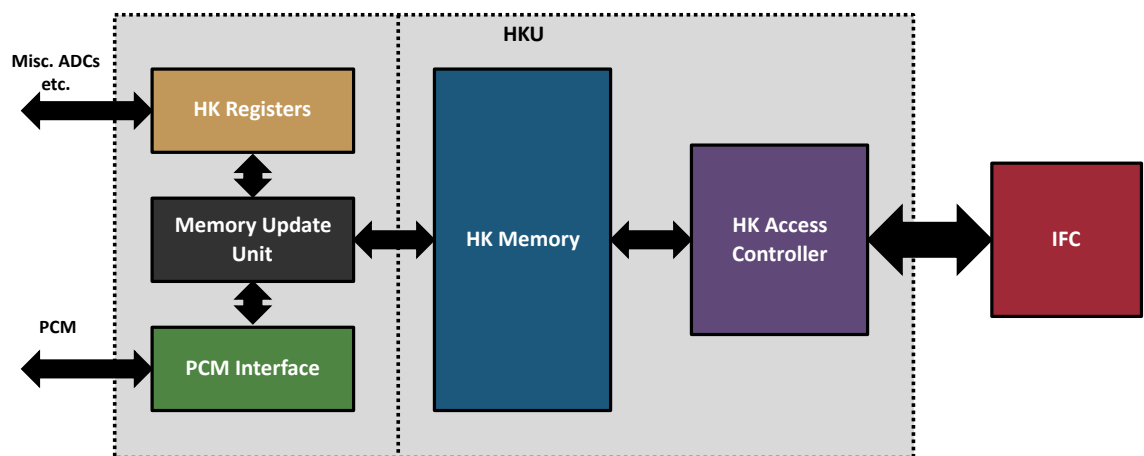
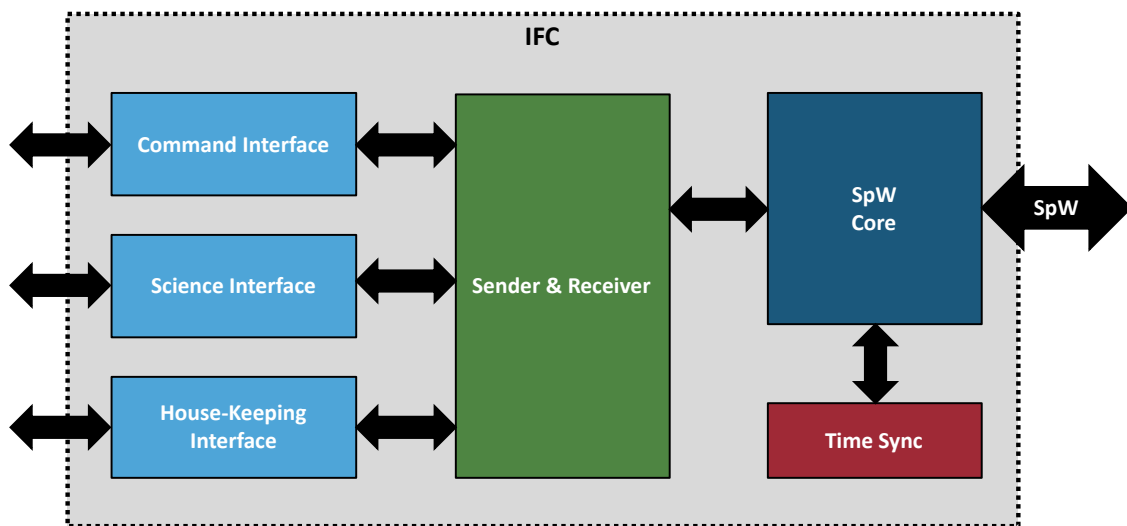


Figure 62: Simplified block diagram of the current "dummy" HK interface.



#### 4.3.9 INTERFACE CONTROLLER

The interface controller (IFC, Figure 63) is the component responsible for transmitting the FP's data to the ICS through a SpaceWire (SpW) interface. It comprises three communication interfaces: the command interface, the science interface, and the housekeeping interface. Additionally, it includes a SpW communication IP core and an FSM that manages and coordinates access to the SpW interface among these three interfaces. The IFC design is very modular and address-based, with each interface being assigned an address used to identify to which interface a TM/TC command belongs. These addresses are kept the same as in the LDA-FP design, simplifying the communication protocols by making them completely compatible.



**Figure 63: Simplified block diagram of the interface controller.**

#### 4.3.10 HIGH SPEED SERIAL INTERFACE

The High-Speed Serial Interface (HSSI) is a unit designed for testing and development of the FP firmware; it allows output of the raw data exactly as received from the ADCs as it enters the CTUs. Strictly speaking, the HSSI is not part of the FP design, since it is used for Electronic Ground Support Equipment (EGSE) purposes only. However, for an overview of awareness and given the criticality of this unit for the development and testing of the FP, the HSSI is briefly presented in this section.

To enable the output of the raw data coming from the FD sensor, a high-speed interface is used, given the fast readout of the sensor at 12,500 fps, with 4096 pixels/frame and 14 bits each, a minimum data transfer rate of 716.8 Mbps is required.

The Gigabit Ethernet protocol was chosen for this purpose, given that it is widely used and would allow connecting the FP output to any computer with an ethernet port.

The HSSI takes the data input as coming from the CTU, prepares the data for its transmission via Gigabit Ethernet, handles the protocol, and transmits it at high-speed using the RTG4 Serializer/De-serializer IP cores. An FPGA Mezzanine Connector (FMC) board is plugged into the FPGA with a Marvel PHY chip that transmits via differential copper lines the information to the host PC. The activation of the interface and number of frames to be transmitted upon activation are configured using the *ETH\_PWR* and *ETH\_BURST\_SIZE* FPGA internal registers. A block diagram with the HSSI architecture is shown in Figure 64.

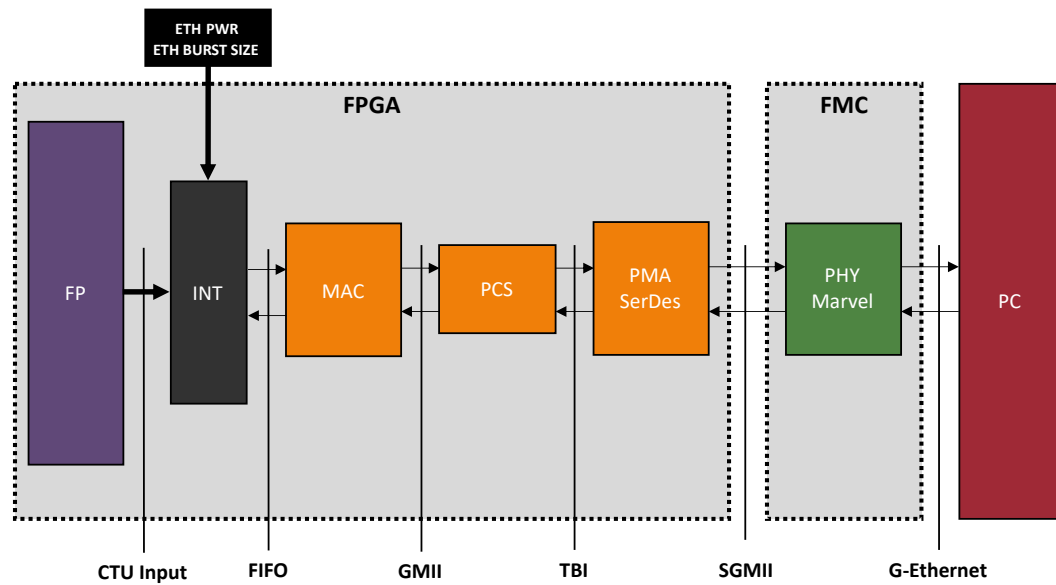


Figure 64: Simplified block diagram of the HSSI.

The details of the Gigabit-Ethernet protocol are out of the scope of this thesis and the FP development. However, given the required time and effort to develop each of these sub-modules, a brief description of its units is given below:

**Frame Processor (FP):** As its name implies, this is the FD-FP. Data is fetched before entering the CTUs and output via parallel interface at each pixel clock.

**Interface Module (INT):** This module oversees reading-out the data as it passes through the frame processor pipeline and sends it to the MAC module to be sent. Its responsibilities also include the task of reading commands coming from the MAC module to activate or deactivate the data interface. This module corresponds to the

Logical Link Control or Mac Client on the Open Systems Interconnection (OSI) reference model according to the IEEE 802.3 standard.

**Media Access Control and Reconciliation (MAC):** The MAC sublayer, as described in Clause 2 of the IEEE 802.3 standard, allows the MAC client to interchange data units with peer entities. These data packets are exchanged to the PCS module through the Reconciliation Sublayer (RS), which uses the Gigabit Media Independent Interface (GMII), as described in Clause 37 of the IEEE 802.3 standard. In simple terms, the MAC layer receives packets coming from the INT module using a FIFO interface. The module then packs the information into ethernet frames, which are then output to the PCS module using the Gigabit Media Independent Interface (GMII) interface.

**Physical Coding Sublayer (PCS):** This module, as described in Clause 36 of the IEEE 802.3 standard, serves as an interface between the RS of the MAC module (GMII interface) and the PMA module using the Ten Bit Interface (TBI). In simple terms, the layer performs encoding (decoding) of GMII data octets to (from) ten-bit code-groups for communication with the underlying PMA. Also, the PCS oversees the Auto-Negotiation (AN) process (as described in Clause 37 of the IEEE 802.3 standard), which allows for the PCS and the PHY module to agree on the capabilities of the link. Finally, this module performs the word alignment of the information received on the TBI.

**Physical Medium Attachment (PMA):** The PMA layer, described in Clause 36 of the IEEE 802.3 standard, provides medium-independent means for the PCS to support serial-bit-oriented physical media. It performs the mapping of transmit and receive code-groups between the PCS and PMA. It is also in charge of serializing and deserializing such code-groups into the Serial Gigabit Media Independent interface (SGMII) for communication with the physical (PHY) module. This layer is technology-dependent and, in our case, is contained in the SerDes Module of Microsemi FPGAs.

**Physical layer Marvell module (PHY):** This commercial FMC module provides RJ45 connectors through its Marvel Alaska (88E1240) device. The transceiver performs all the physical layer functions for the 1000BASE-T Ethernet on CAT 5 twisted pair cable. It communicates with the PMA module through SGMII, and it supports the IEEE 802.3 standard, including its auto-negotiation (AN) functions.

The data output format of the HSSI follows the structure of Figure 65.

| Address | 0          | 1  | 2          | 3  | 4          | 5  | 6          | 7  | 8         | 9  | A         | B  | C         | D  | E        | F  |
|---------|------------|----|------------|----|------------|----|------------|----|-----------|----|-----------|----|-----------|----|----------|----|
| 0000    | FF         | FF | FF         | FF | FF         | FF | 00         | 01 | 02        | 03 | 04        | 05 | 08        | 00 | 45       | 00 |
| 0010    | 05         | CE | 00         | 00 | 00         | 00 | FF         | 11 | CRC       |    | C0        | A8 | 00        | 00 | FF       | FF |
| 0020    | FF         | FF | 00         | 80 | 00         | 20 | 05         | BA | 00        | 00 | 00        | 00 | CTU N n   |    | CTU S n  |    |
| 0030    | CTU N n+1  |    | CTU S n+1  |    | CTU N n+2  |    | CTU S n+2  |    | CTU N n+3 |    | CTU S n+3 |    | CTU N n+4 |    | CTU S +4 |    |
| ...     | Continue   |    |            |    |            |    |            |    |           |    |           |    |           |    |          |    |
| ...     | CTU N 2046 |    | CTU S 2046 |    | CTU N 2047 |    | CTU S 2047 |    | FF        | FF | FF        | FF | CTU N 0   |    | CTU S 0  |    |
| ...     | Continue   |    |            |    |            |    |            |    |           |    |           |    |           |    |          |    |
| 05A0    | CTU N m    |    | CTU S m    |    | CTU N m+1  |    | CTU S m+1  |    | CTU N m+2 |    | CTU S m+2 |    | 00        | 00 | 00       | 00 |
| 05B0    | 00         | 00 | 00         | 00 | 00         | 00 | 00         | 00 | 00        | 00 | 00        | 00 | 00        | 00 | 00       | 00 |
| 05C0    | 00         | 00 | 00         | 00 | 00         | 00 | 00         | 00 | 00        | 00 | 00        | 00 | XX        | XX | XX       | XX |
| 05D0    | XX         | XX | YY         | YY | YY         | YY | YY         | YY | 00        | 00 | 00        | A1 |           |    |          |    |

Ethernet headers

Padding

Frame number

F-Version

Frame marker

Pixel data

Packet number

**Figure 65: Structure of the HSSI packets.**

The data packet begins with the Ethernet, IP-UDP protocol headers. The following 2 bytes of padding are added to align the pixel data to 32-bit words. Pixel data follows with a pixel from each side of the sensor, each using 16 bits. The output follows the output format from the FD sensor and is left to the operator to account for the sensor's asymmetric readout when analyzing the data. After the frame data, the footer begins; it is composed of padding, the frame number, packet number, and the f-version data.

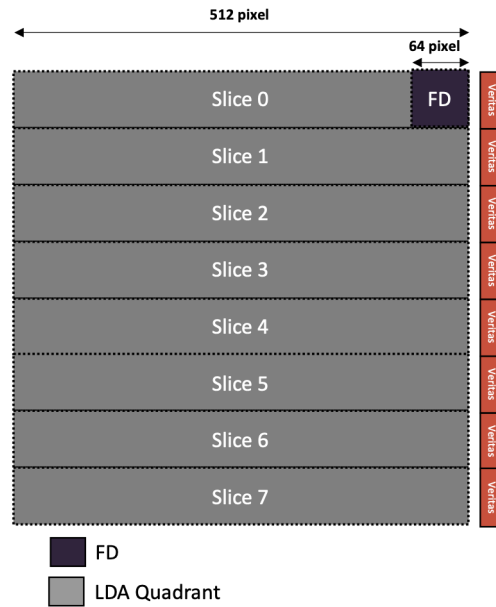
## 5 THE LARGE DETECTOR ARRAY FRAME PROCESSOR

In a similar manner to the FD sensor, the LD-FP comprises a digital electronics design written in VHDL, which is implemented in the RTG4 FPGA. While the LD-FP and FD-FP share similarities beyond implementation technology, they also have a significant overlap in processing modules and architectural features. However, due to key functional and architectural differences between the LD and FD sensors, the LD-FP incorporates a unique set of features tailored to address its specific requirements and challenges, enabling successful sensor readout.

In this chapter, the LD-FP design is delved into in greater detail. Given the shared resources and similarities to the FD-FP, this chapter primarily focuses on the key differences between the two designs. Only the aspects that differ from the FD-FP design are provided in detail.

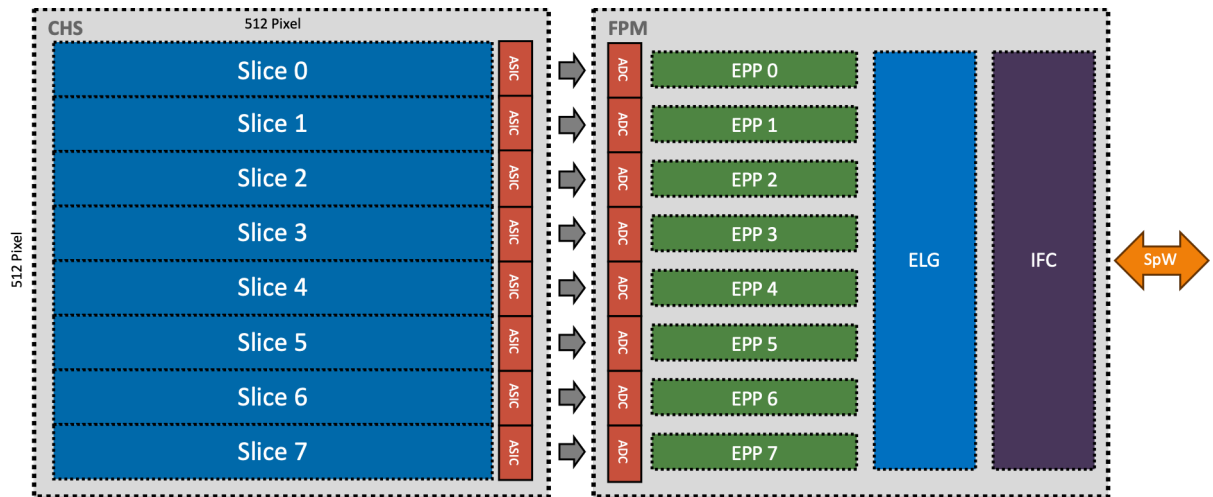
### 5.1 THE LARGE DETECTOR ARRAY REDOUT REQUIREMENTS

As was described in the previous chapter, both the LD and LDA sensors use the same readout Veritas ASICs and therefore the pixel-wise sensor-readout as well as its required processing features are almost identical. However, key differences arise when comparing the main properties and key characteristics of both sensing instruments. The FD is geared towards very high-speed readout performance enabling the observation of point-like bright sources, while the LDA sensors aim to perform imaging of a wide area in the sky with less strict requirements on readout speed performance. While this means that the LD-FP does not need to operate at such high-speeds on a frame basis, the large area requirement means that the sensors on each LDA quadrant are 64 times larger than the FD sensor (see Figure 66), imposing a great challenge regarding the vast number of pixels per frame to be processed when compared to the FD-FP.



**Figure 66: Size comparison between the FD sensor and a quadrant of the LDA. The LDA quadrant is 64 times larger than the FD.**

To address the raw-processing performance challenge, the LD-FP leverages the FPGA's capabilities, particularly its large parallelization capabilities. Unlike the FD-FP, which processes data alternating two pipelines, the LD-FP employs eight parallel pipelines, mirroring the eight Veritas ASICs reading out an LDA quadrant. This approach, illustrated in Figure 67, enables the processing of over a million pixels of the LDA at an astonishing rate of 200 frames and up to 800 frames per second. First, the data is divided into four quadrants, each with its own readout and DE. Then, each LDA sensor's data is processed into one FPGA chip containing eight parallel pipelines. In other words, the 1024 x 1024 pixels of the LDA are processed in four RTG4 FPGA devices, each with eight parallel pipelines, effectively processing a slice of 512 x 64 pixels each. This unprecedented combination of parallelization and real-time processing far surpasses the capabilities of any previous generation mission.



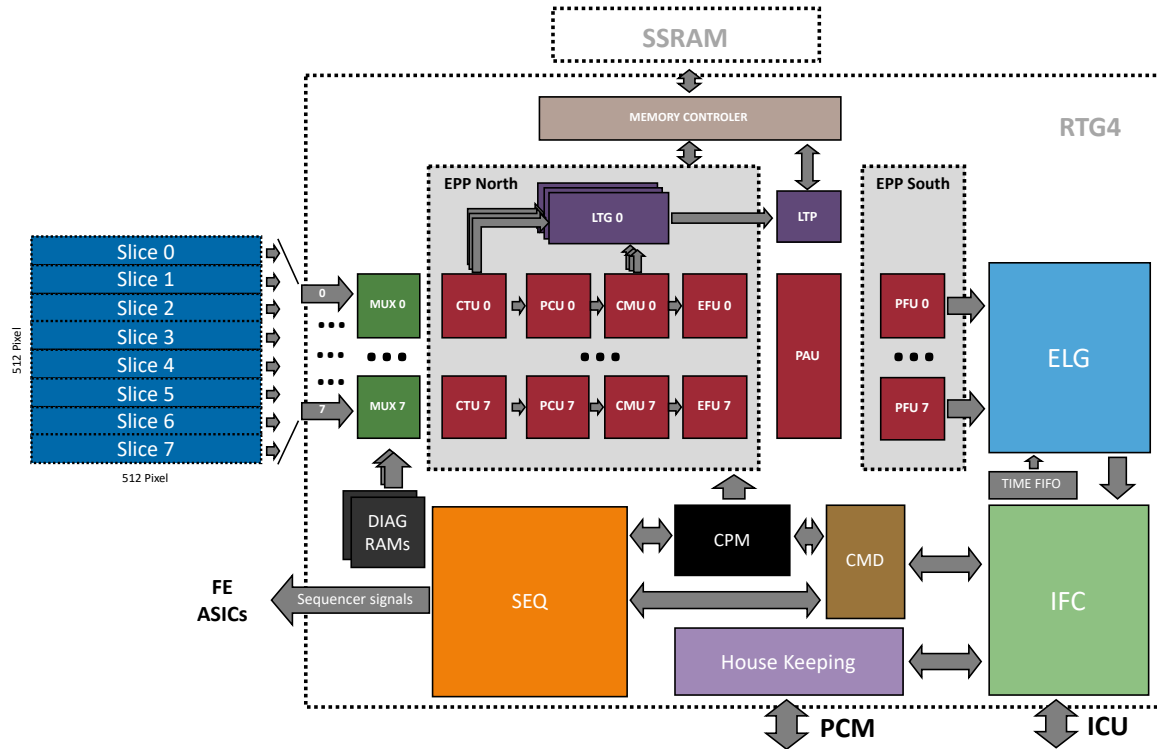
**Figure 67: Block diagram showing the data processing flow in the LD-FP. Data is read out and converted in parallel coming from the eight Veritas ASICs. In a mirrored manner, data is preprocessed in eight parallel pipelines. Afterwards, data is merged into an event list and sent to the ICS via a SpW interface.**

After the data has been preprocessed in each of the eight parallel pipelines, a mutual unit called the ELG collects the relevant pixels' data and generates a frame-wise event list. This list is then transmitted to the ICS through its SpW communication interface.

## 5.2 THE LD FRAME PROCESSOR ARCHITECTURE

The LDA requires a greater degree of parallelization when compared to the FD. Therefore, its architecture expands parallelization to eight parallel EPP pipelines instead of two. These pipelines, unlike in the FD-FP, work together synchronously and in parallel thanks to the LDA readout architecture. Data is acquired from the LD sensors in a single direction with all the Veritas ASICs reading out the same pixel-row at the same time, concurrently. This characteristic avoids the need for the MBU, as data is already read out in an organized manner. Figure 68 depicts a block diagram with the LD-FP architecture. The eight parallel pipelines are subdivided into two separate areas with the PAU in between, the EPP-North and EPP-South sections. Given that the PAU shall be capable of analyzing pixel patterns over the whole sensor, regardless of the readout slice, the PAU works as a single structure for the eight parallel pipelines. The ELG unit occupies a more complex scheduling system given that data in the eight parallel pipelines corresponds to pixels belonging to the same frame, contrary to the FD-FP where the pipelines output synchronously alternating frames. A major difference with the FD-FP design lies in the necessity of an external memory, where the LUTs necessary for the PCU and EFU operation are stored. Therefore, a memory controller is utilized to interface the external SSRAM memories with the LD-FP EPPs. The LTG unit

accommodates to these differences as well, by dividing its functions into two separate structures, eight LTG units, one on each EPP, and one Look-up Table Programmer (LTP) unit, which interfaces the LTGs with the memory controller and oversees the storage of the LUTs in the external SSRAM during offset and noise maps generation.



**Figure 68: General architecture block diagram of the LD-FP firmware.**

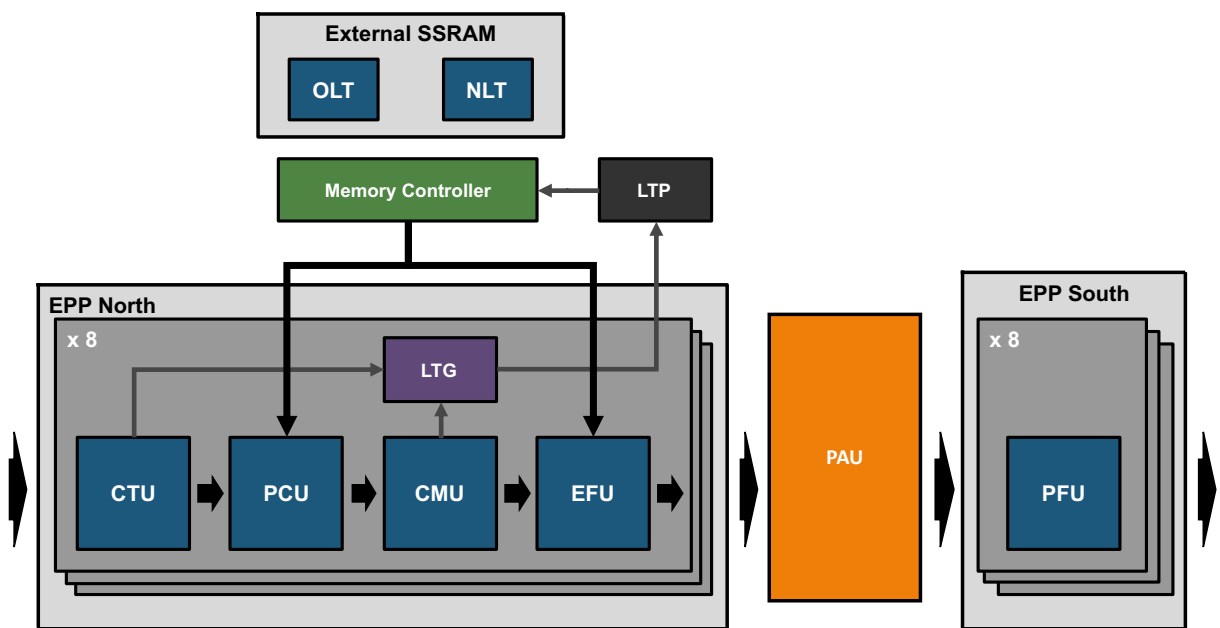
As stated before, good design practices and modular design enabled by VHDL properties allow for reuse of most of the modules between the FD-FP and LD-FP. In the EPPs in specific, the CTU, PCU, CMU, EFU, and PFU are completely identical in both designs. Most of the ancillary units are also shared, including the SEQ, IFC, and HK units. Therefore, the following subsections will only focus on the units where changes were necessary: these are the PAU, LTG/LTP, ELG, and a brief description of the memory controller.

### 5.3 EVENT PREPROCESSING PIPELINE

The EPP (a block diagram is shown in Figure 69), the brain of the FP, sequentially preprocess data belonging to read-out pixels in a series of stages. These stages perform pixel-energy calibration, enabling accurate classification of valuable X-ray data events for transmission to the ground.



The pipelines are divided into three areas. The first area is the EPP-North, which includes the CTU, PCU, CMU, and EFU. Additionally, the EPP-North houses the LTG, an auxiliary unit responsible for calculating the Offset and Noise maps used by the PCU and PFU. Next is the PAU, which is internally divided into three subunits: the Slice Pattern Analyzer (SPA), the Border Pattern Analyzer (BPA), and the Pixel Data Buffer (PDB). More details about the PAU are provided in the subsequent subsection. Finally, after the PAU processes the data, it is transmitted to the EPP-South, which contains only the PFU, the final stage of the pipeline where valid pixels are filtered and selected for ground transmission.



**Figure 69: Simplified block diagram showing the EPPs architecture.**

Given that, with the exception of the PAU, all units in the EPP are identical to the FD-FP units, only a brief description of each unit is provided in the form of a list:

- **Cross-Talk Unit (CTU):** This unit corrects for the cross-talk effect caused by bandwidth limitations when transmitting the analog pixels' data from the CHS to the DES. The correction process involves subtracting the previous readout pixels' energy multiplied by a programmable coefficient.
- **Pixel Correction Unit (PCU):** This unit performs the offset correction of the pixels' energy. The offset is pixel-specific, so the PCU uses offset data stored in external LUTs. The process involves subtracting the offset (mean energy)

for each pixel. Hot and dead pixels are identified, and a threshold discriminates between MIP and MISFIT pixels.

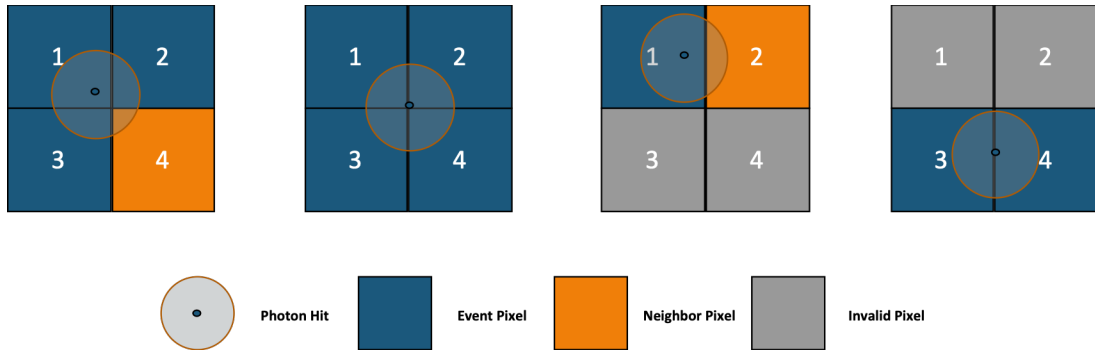
- **Common Mode Unit (CMU):** This unit corrects for common mode noise that arises when a pixel-row is readout. The process involves calculating the median energy of all pixels in the same row (excluding hot and dead pixels) and subtracting this median from the energy of all pixels in the same row.
- **Event Filtering Unit (EFU):** In the EFU, pixels with enough energy to be considered X-ray events are flagged as event pixels. Likewise, pixels that could belong to an event (neighbor pixels) are identified and flagged accordingly. The process involves comparing each pixel with its individual threshold, which is calculated as the pixel noise (standard deviation) multiplied by a fixed coefficient. The coefficient for event pixels is larger than the one used for neighbor pixels.
- **Pattern Analyzing Unit (PAU):** The PAU analyzes the patterns created by event and neighboring pixels. Whenever a pattern is not point-like (up to four clustered pixels) as expected by an X-ray event, the pixels are flagged with an illegal pattern flag. Additionally, the PAU flags two pixels around MIP pixels and promotes neighboring pixels that are contiguous to an event pixel to event pixels. Detailed information about the PAU's operation is provided in the following subsection.
- **Pixel Filtering Unit (PFU):** The PFU selects pixels based on programmable criteria by comparing its information to a programmable mask. Pixels that meet these criteria are marked as valid pixels and are sent to the ELG for packaging and subsequently to the IFC for transmission.

---

#### 5.3.1 PATTERN ANALYZER UNIT

The PAU analyzes the patterns created by events, neighboring pixels, and MIP pixels in the DEPFET sensors. It is crucial to reduce the amount of background events data transmitted to the ground by removing illegal event patterns and MIP neighboring pixels. Additionally, the PAU aids in increasing energy resolution by avoiding a redshift skew of the data when a small portion of incoming photons' energy is shared by neighboring pixels. This promotes neighboring pixels to event pixels, preventing data loss. When a photon hits a DEPFET sensor, its energy is deposited in a point-like manner in the sensor's silicon bulk. These point-like patterns can be centered in the border of two or four pixels, with a maximum of four pixels when the hit occurs directly between four

pixels. Always in a point-like manner and never occupying more than two pixels across (see Figure 70), these patterns allow us to discriminate X-ray photon events from those of charge particles. Charge particles interact with the silicon bulk differently due to their electric field, causing them to scatter and produce worm-like traces on the detector surface, expanding several pixels. The PAU exploits this difference in patterns to differentiate photon events from background events.

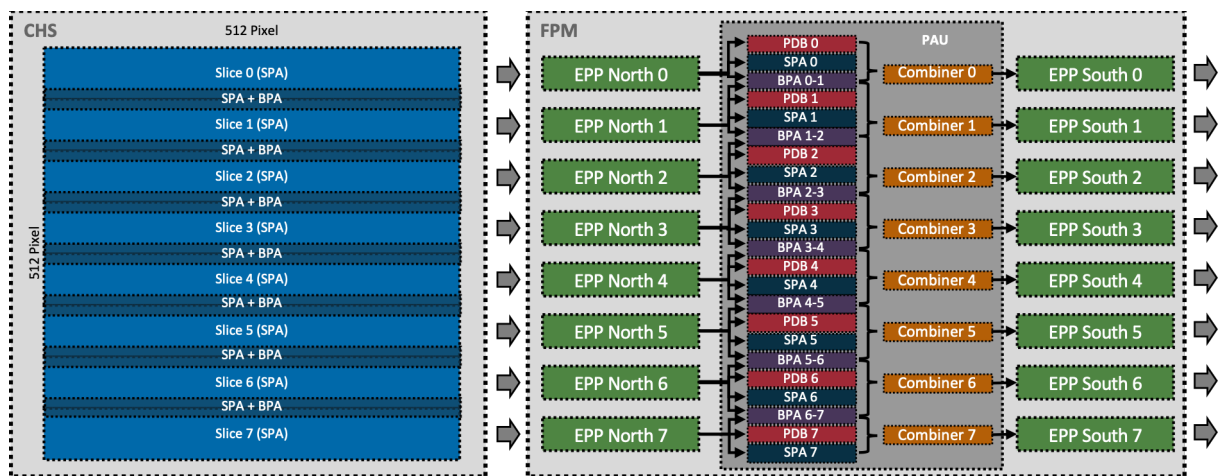


**Figure 70: Examples of the patterns generated by X-ray events. The PAU allows to identify neighbor pixels and promotes them to event pixels to prevent data lost.**

A problem that arises with the eight parallel pipeline structure used in the LD-FP is the fact that while sensor readout is performed in slices and most pipeline units operate either pixel-wise or row-wise, and therefore, not affected by the segmented readout. The traces generated by background events occur sensor-wise in any position, independent of the readout slice. For instance, a trace can begin in, e.g. the border of slice 0 and propagate to slice 1. If the pattern left in slice 0 is composed of less than four pixels, a PAU operating in the same segmented manner as the other units would not be able to detect it.

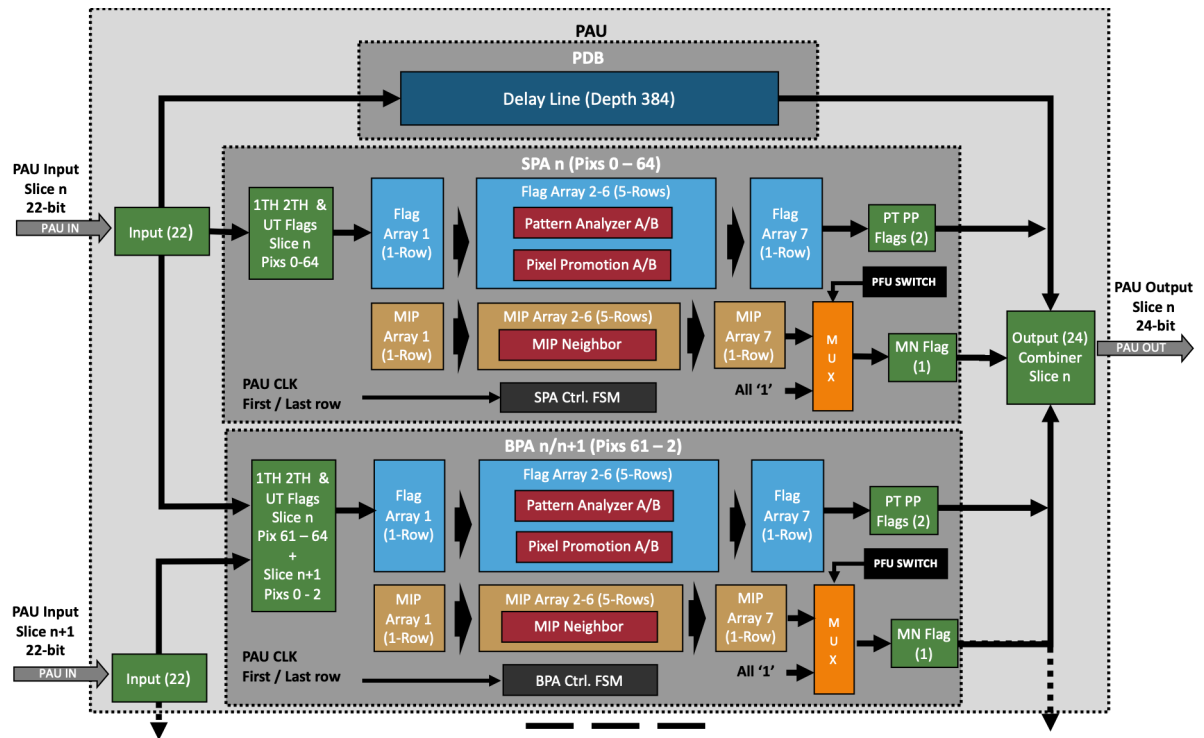
To address this challenge, the PAU was designed as a single unit that serves as a common component for the eight parallel pipelines. This unit is further divided into three distinct modules: the Slice Pattern Analyzer (SPA), Pattern Data Bypass (PDB), and Border Pattern Analyzer (BPA). The SPA and PDB function similarly to those found in the FD-FP. Consequently, the SPA performs pixel pattern analysis, pixel promotion, and MIP neighbor pixel flagging tasks on the flag array vectors corresponding to the pixel data of rows of 64 pixels, the entire sensor width for the FD, and a readout slice for the LD. The PDB buffers the energy information for these pixels while the analysis tasks are performed on the flag arrays. Since each SPA and PDB covers a slice 64 pixels wide, eight of these units are utilized in parallel to cover the 512-pixel sensor width. The remaining unit, the BPA, operates similarly to the SPA, with two key differences: it operates at the border of two contiguous readout slices, comprising six BPA units in

total. Additionally, it analyzes flag arrays with a width of six pixels, with three pixels corresponding to each of the adjacent readout slices where the BPA is centered. The BPA performs parallel analysis of the overlapping border pixels alongside the SPA. At the exit of the SPA and BPA, the pixels' energy information is merged with the flags generated by both the SPA and BPA. In other words, a pixel only needs to be flagged as illegal by either the SPA or BPA for the output of the PAU to indicate the illegal flag result. This ensures that the pattern analysis handles any illegal pattern regardless of the sensor location. Figure 71 illustrates a schematic showing how the sensor pixel analysis is distributed in the SPA and BPA.



**Figure 71: Schematic showing the distribution of the sensor sections (left) and their corresponding processing units inside the PAU (right).**

Figure 72 illustrates a simplified block diagram that reveals the internal structure of the PAU. Due to the unit's high complexity, only one SPA and one BPA are depicted, along with their corresponding PDB. It is important to note that the BPA receives pixels from two separate adjacent slices. Additionally, at the output, the results from both the SPA and BPA are merged, with the BPA flags information being sent to the two contiguous slices. It can also be noted that the MIP neighbor flagging can easily be switched off using the *PFU\_SWITCH* register which disconnects the output of the MIP array via multiplexing switch.



**Figure 72: Simplified block diagram showing the internal structure of the PAU, to aid readability only one SPA, one PDB and one BPA module are shown.**

## 5.4 EVENT LIST GENERATOR

The ELG in the LD-FP, similar to the FD-FP, oversees the packaging of pixel information into “Event Lists.” These Event Lists contain information about all valid pixels in a single frame. The Event Lists in the LD-FP share the same format as those in the FD-FP, with the only difference being that the LD-FP uses the extra address bits previously not utilized by the FD-FP. This is because the LDA containing more pixels naturally requires more address bits.

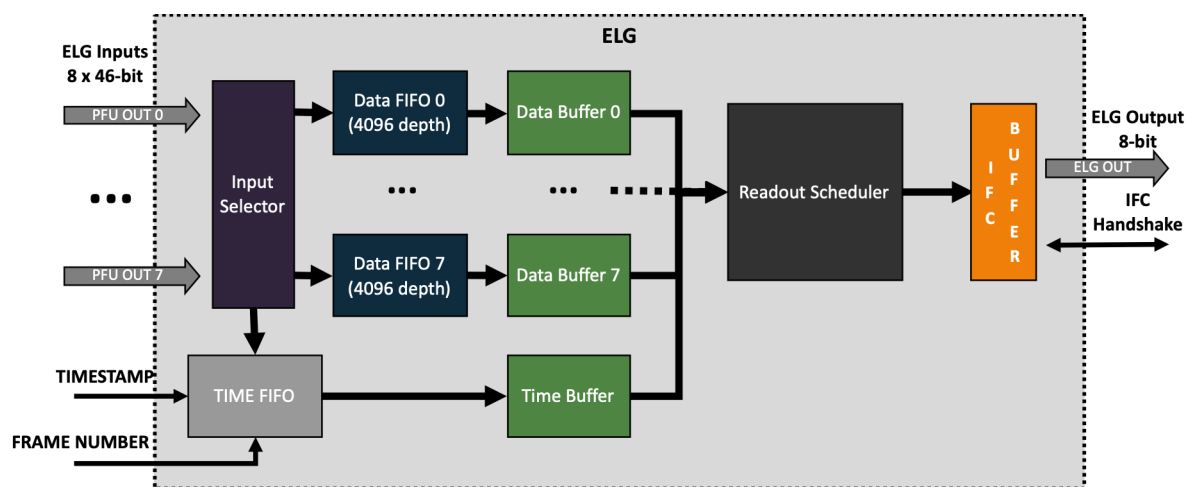
The architecture of the LD-FP ELG shares some similarities with the FD-FP. However, given that the ELG must fetch data from eight pipelines instead of two, the algorithm of the LD-FP ELG is significantly more complex. The general architecture of the ELG is depicted in the block diagram of Figure 73.

The outputs of the eight parallel pipelines are connected in parallel to the input selector module. This module reviews the Valid Pixel flag assigned by the PFU to determine if the incoming pixels are valid. When a valid pixel is detected, the input selector saves the pixel information into the corresponding pipeline data FIFO buffer. This action signals the time FIFO to store the valid pixels’ frame header. The frame header contains a consecutively increasing event number that cycles back every 256

frames (eight bits) and a unique 38-bit frame timestamp generated by the central clocking unit of the LD-FP.

Each of the eight data FIFOs has a depth of 4096 words and a width of 45 bits, enabling it to store the complete information of a single pixel on each word. The time FIFO has a depth of 256 words and a width of 48 bits, storing the frame header information of each frame that contains a valid pixel on each data word.

Following each of the nine FIFOs is the data buffer module, which simply consists of a state machine responsible for reading out and preloading the data into an output buffer, making it readily available for readout by the readout scheduler module.



**Figure 73: Simplified block diagram of the ELG.**

The readout scheduler module executes the most intricate algorithm within the ELG. It employs a finite state machine (FSM) to systematically read pixel data from the data buffers and time buffer and store it in the interface controller buffer (IFC buffer). In simpler terms, the readout scheduler FSM begins by reading the time buffer, storing the information in the IFC buffer, and recording the corresponding frame number. After the frame header is transmitted, the readout scheduler compares the information in the first data buffer with the pixels' frame number in the transmitted frame header. If a match is found, the corresponding pixel data is fetched and sent to the IFC buffer. The readout scheduler repeats this process with the subsequent data buffers, fetching matching pixel data and sending it to the IFC buffer. Once no more pixels matching the frame number are detected, the readout scheduler retrieves the next frame header from the time buffer and resumes the process for the next frame. It is important to note that while the algorithm description is theoretically simple, its implementation poses significant challenges due to the tight timing constraints imposed on the readout

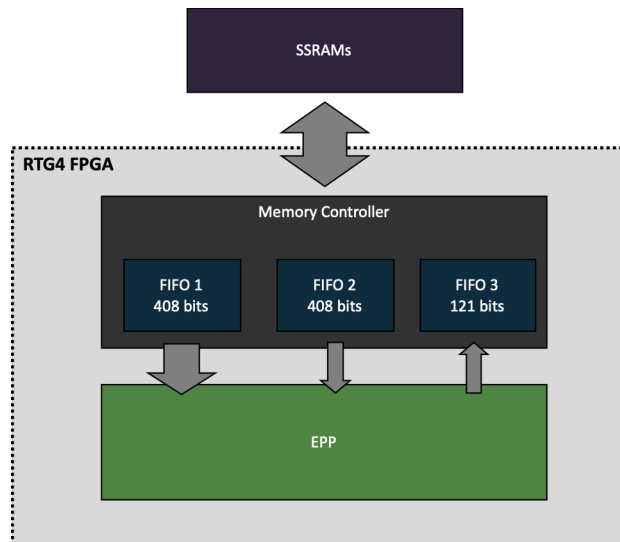
scheduler to maintain data organization and prevent data loss. This not only makes the implementation challenging but also the verification process, given the diverse scenarios and methods through which valid data is output.

The final module in the ELG is the IFC buffer, where the readout scheduler stores the outgoing data. This module is a simple buffer equipped with a handshake mechanism that enables the IFC to read out the six data chunks comprising a 42-bit data word and transmit their bytes via the IFC's SpW interface. Additionally, the IFC buffer is responsible of signaling the readout scheduler whenever new data can be stored, helping to maintain a consistent transmission rate and prevent data loss.

## 5.5 MEMORY CONTROLLER

The memory controller is the unit responsible for interfacing the LD-FP with external SSRAM memories that contain the LUTs necessary for the PCU and EFU's operation. These external memories are required because the LDA quadrant has a large number of pixels ( $515 \times 515$ ), and each pixel requires 48 bits of storage. Therefore, the total storage required would exceed 12 Mbits, which is more than twice the resources available in the RTG4 FPGA.

Since the SSRAM memory selected for the FP PRE-EM is expected to be obsolete by the Athena launch date, the current memory controller will not be included in the final flight configuration. Moreover, this unit was developed externally by engineers at MPE, with only its interfaces being relevant to FP development. As a result, a comprehensive review of this unit falls outside the scope of this work. Instead, this subsection provides a brief overview of its interfaces to enhance the reader's awareness and offer essential information that supports an understanding of the design choices made for the LTG, more details on the operation of the SSRAM interface registers can be found in Appendix B: LD Interface Controller Operation.



**Figure 74: Simplified block diagram of the memory controller interface with the EPP.**

As shown in the block diagram of Figure 74, the memory controller interface consists of three FIFO buffers: FIFO 1, FIFO 2, and FIFO 3. These buffers function as an asynchronous interface to the memory control logic, ensuring smooth communication despite the frequency disparity between the external SSRAM memory, which operates at a 125 MHz Double Data Rate (DDR), and the main system clock of the FP, which runs at 80 MHz. The FIFO buffers act as a protective barrier, mitigating clock domain crossing issues that typically arise when exchanging data between asynchronous clock domains.

Each of the three FIFOs has a distinct function, with FIFO 1 featuring a notably wider 408-bit interface compared to FIFO 2 and FIFO 3, which are both 121 bits wide. Each of the three FIFOs has a distinct function, with FIFO 1 featuring a notably wider 408-bit interface compared to FIFO 2 and FIFO 3, which are both 121 bits wide. FIFO 3 operates as a control input buffer, enabling the FP to load data into the SSRAM while also managing commands and configurations for the memory controller. FIFO 2 serves as an output port, allowing the memory controller's internal registers to be read and, upon commands received via FIFO 3, facilitating the retrieval of specific data words from the SSRAM.

The primary buffer, FIFO 1, plays a crucial role when the LD-FP is active, as its interface accommodates the three 16-bit data words needed for processing eight pixels simultaneously, totaling 384 bits. Due to the high-speed demands of SSRAM operation, the memory controller functions in a clocked continuous readout mode during FP operation, pre-loading the necessary 384-bit data into the FIFO buffer. This ensures that the FP can access the required information to perform corrections and flagging for each pixel efficiently. To prevent synchronization errors that could result in data



mismatching, the FIFO 1 buffer output includes a 16-bit address word, which the FP verifies during readout.

## 5.6 LOOK-UP TABLE GENERATOR AND LOOK-UP TABLE PROGRAMMER

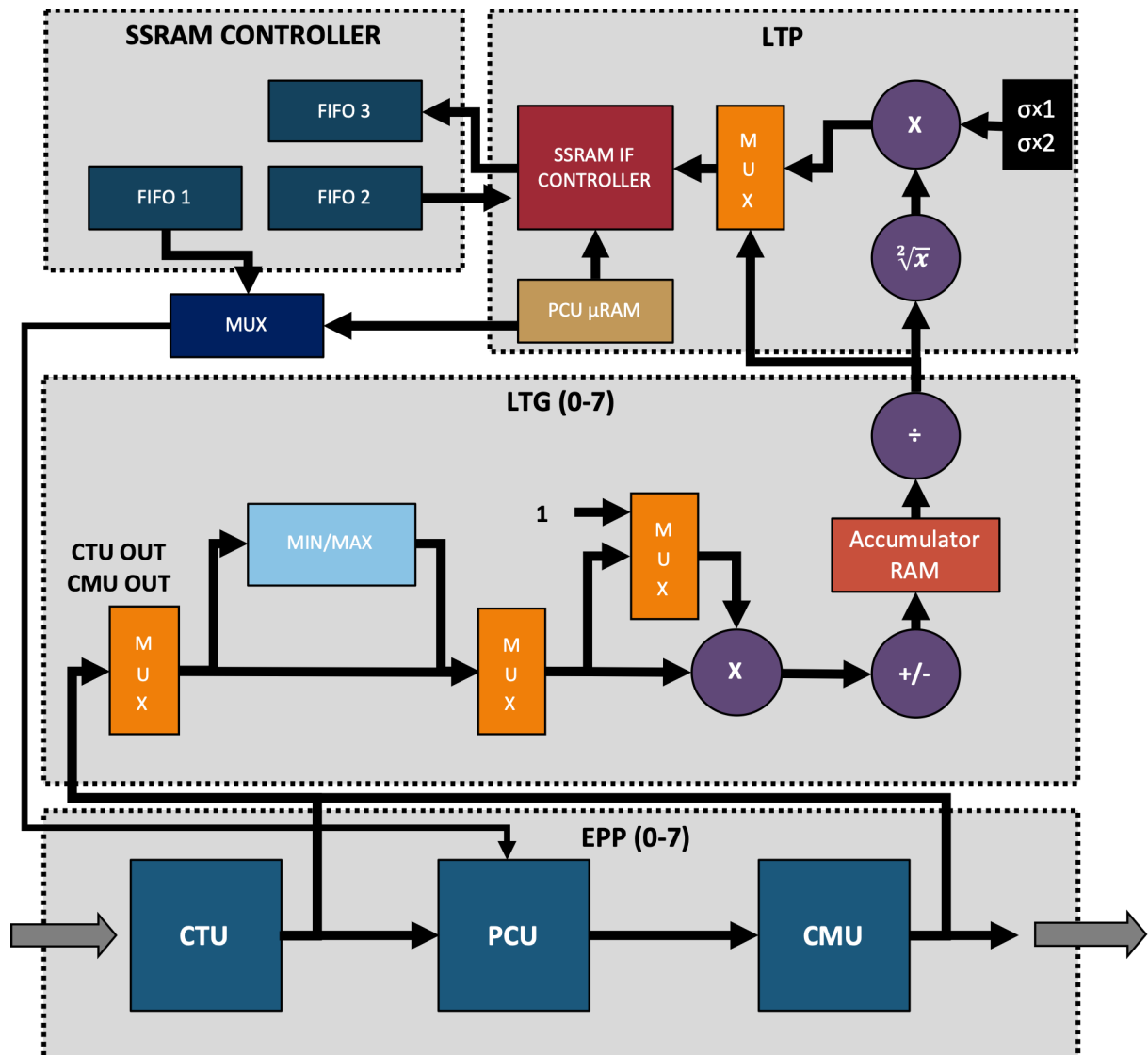
Just like in the FD-FP, the generation of Look-Up Tables (LUTs) containing pixel offset and noise maps, OLT and NLT respectively, essential for the operation of the PCU and EFU units, takes place within the FP itself. This eliminates the need to download raw data to the ground, a necessary measure given the vast amount of data produced by the LDA sensors. Transmitting the 256 frames required for LUT calculations would be nearly impossible due to bandwidth limitations in the SpW interface and the satellite downlink. Reducing the frame speed is not an option either, as the offset map accounts for pixel dark current, which increases with exposure time. To maintain the energy resolution requirements, the LUTs must be computed at the same readout speed as used during data acquisition observations.

The calculation requires performing the same mathematical computations as for the FD-FP. However, in the case of the LD-FP, compared to the FD-FP, the challenge shift is not due to synchronization issues but rather the sheer volume of pixels involved. This prevents the calculation buffers from fitting within the FPGA memory resources, necessitating the use of external SSRAM memories for processing. Consequently, the LTG must be capable of operating these SSRAM memories via FIFO interfaces, prompting the development of an interfacing module, the Look-up Table Programmer (LTP), that manages SSRAM memory read and write operations.

Further complicating matters, the LD-FP requires a greater share of FPGA resources than the FD-FP, as it needs more internal registers to manage the movement of a higher number of pixels. This makes efficient FPGA resource allocation just as critical as functional development for the LTG. To address this, the same resources used for creating the offset map are also leveraged for noise map generation. By cleverly integrating modules and utilizing multiplexers, their functionality is altered using the same combinatorial logic resources.

Due to memory constraints, the OLT and NLT generation procedures are carried out in segments of four sensor rows at a time rather than over the entire frame. The process begins with the OLT calculation over 256 frames for all pixels in the first four rows of the sensor. Once completed, it advances to the next four rows and repeats 128 times until all 512 rows have been processed. Similarly, the NLT calculation follows the same four-row incremental approach. This method can also be applied to smaller detector windows, down to 64 rows, facilitating LUT generation for the LDA window mode operation.

Figure 75 presents a simplified block diagram illustrating the interaction between LTGs, LTP, and the Memory Controller. Eight LTG blocks, one per EPP, retrieve data from their respective pipelines, performing accumulation and preparation. Once accumulation is complete, the LTP stores the processed data within the memory controller. A more detailed explanation of Offset LUT Generation (OLG) and Noise LUT Generation (NLG) follows in the upcoming subsections.



**Figure 75: Simplified block diagram showing the structures of the EPP, LTG, LTP and memory controller. All interfacing together to facilitate the OLG and NLG tasks inside the FPGA.**

#### 5.6.1 OLT CALCULATION

The OLG generation process begins upon receiving the OLT generation command. It starts by initializing all LTG memory units, setting MIN-RAM, which tracks the four lowest energy pixel values, to its maximum value (all 14 bits set to one); MAX-RAM to its minimum (all 14 bits set to zero); and the accumulation RAM to zero. Simultaneously, the LTP retrieves OLT values for all pixels in the first four rows of the detector, or area window under evaluation, and stores them in FPGA internal SRAM, as the external SSRAM lacks asynchronous read/write capabilities. This step ensures the LTP retains flags for hot and dead pixels, bits 14 and 15 of the OLT word, even when a new map is generated.

Following initialization and synchronization with the start of a new frame, the accumulation phase begins. Data flows from the CTU to the LTG, where two operations occur: pixel data is compared to the fourth lowest and fourth highest recorded values and stored in the respective MIN/MAX RAM addresses if lower or higher; meanwhile, the data passes through a multiplying accumulator unit, with a factor of one, and is summed with previously accumulated pixel data. Initially, accumulated data is zero, and each new value is stored in its corresponding address. After processing the four in-focus rows, accumulation halts, allowing the MIN/MAX RAM sorter algorithm to arrange values such that the fourth lowest and highest are readily accessible for future comparisons. The cycle repeats upon the next frame's start and runs 264 times.

Once accumulation concludes, the MIN/MAX RAM data is read and processed through the multiplying accumulator unit, where the four minimum and maximum pixel values are subtracted from the total, leaving a sum of 256 frames. This subtraction mitigates data skew caused by sporadic pixel events or MISFITs. Subsequently, data from the eight LTGs is sent to the centralized LTP, which stores it in external memory via the memory controller. During transmission, the LTP truncates eight least significant bits, effectively executing an eight-bit right shift operation, equivalent to dividing the accumulated sum by 256, ensuring the LTP receives mean pixel energy values.

During storage, the LTP retrieves the offset map recorded at initialization and appends hot and dead pixel flags to the newly generated map. This cycle, comprising initialization, accumulation, subtraction, and storage, repeats up to 128 times to process all 512 detector rows, thereby completing the OLT calculation procedure.

---

### 5.6.2 NLT CALCULATION

As described at the beginning of this section, the NLT calculation utilizes the same units as the OLT calculation. The procedure is very similar, with a few key differences. In this case, the NLT calculation involves determining the standard deviation of pixel energies, or pixel noise. This subsection provides a summary of the procedure.

The first step is the initialization process, which again includes initializing the MIN/MAX RAMs, accumulation RAM, and reading the OLT values of the four in-focus pixel rows. The objective is not to retain hot and dead pixel flag data but rather to enable the operation of the PCU during NLT generation, effectively incorporating the mean subtraction term in the standard deviation formula. This step is necessary due to the limitations of the external SSRAM memory, which does not support dual-port read and write asynchronous operation, unlike the FPGA's internal SRAM. To ensure accuracy, the OLT calculation must be performed before the NLT calculation.

After initialization, the accumulation phase begins, tracking the four minimum and maximum pixel values in the MIN/MAX RAMs while accumulating data in the accumulation memory. However, when the data passes through the multiplier accumulator unit, it is multiplied by itself, implementing the squaring term of the standard deviation formula. The accumulation process temporarily halts while the rest of the frame is read out and the MIN/MAX RAM sorting algorithm completes its tasks. It then resumes until 265 frames have been accumulated. Once this threshold is reached, the subtraction of the minimum and maximum values takes place. In this stage, the values also undergo squaring when passing through the multiplier accumulation unit before being subtracted from the accumulated sum, leaving 257 accumulated frames at the end of the operation. This count includes an additional frame compared to the OLT calculation procedure. The extra frame accounts for the  $n - 1$  term in the standard deviation formula while maintaining a division factor of 256 at the LTG output, which simplifies calculations in binary representations.

Following subtraction, the truncated data is sent from the LTGs to the LTP. Unlike in the OLT process, this data is not stored immediately but undergoes two additional processing steps. First, the square root is computed, completing the standard deviation calculation. Second, the standard deviation is multiplied by two programmable factors,  $\sigma_1$  and  $\sigma_2$ , typically corresponding to five-sigma and three-sigma thresholds, which are stored in the NLT and used by the EFU to determine event and neighboring pixels.

Once the data is stored, the process repeats for the remaining rows until all 512 sensor rows have been processed, marking the completion of the NLT calculation procedure.

## 6 PERFORMANCE AND TESTS

The FPGA development requires a diverse range of verification and testing methods to ensure the design meets specifications. These tests range from fundamental development stages, such as successful VHDL synthesis and timing closure, typically handled by Electronic Design Automation (EDA) tools within the FPGA vendor's design flow, to advanced functional validation tests conducted directly on hardware.

This section provides a summary of the tests performed for the development of LDA-FP and FD-FP, highlighting the key outcomes of each stage. FPGA hardware testing can be just as demanding as the design process itself. To maximize resource efficiency, most tests were conducted on either LDA-FP or FD-FP rather than both. Given the highly modular design approach and the significant overlap in design modules, this methodology is both practical and successfully implemented.

### 6.1 IMPLEMENTATION

The implementation of a VHDL design onto the target FPGA technology is a multistage process, typically aided by FPGA vendor-provided EDA tools. For Microchip RTG4 FPGAs, these tools are integrated into the Libero SoC Design Suite, which offers a comprehensive environment to transition a design from VHDL source code to FPGA programming. This process, though not entirely linear, consists of the following stages:

1. **VHDL Code Development:** This stage involves developing the source code that describes the design. Designers typically use a text editor, define design requirements, and create block diagrams, flowcharts, and I/O port mappings. During development, designers may also decide to incorporate IP cores, which are pre-designed reusable logic blocks that accelerate development by providing specialized functionalities such as memory management or communication protocols. FPGA vendors offer a variety of IP cores, which facilitate the instantiation of block memories and arithmetic units.
2. **Simulation:** Throughout development, designers use simulation tools to verify expected behavior. These simulations help validate the design by constructing logic based on inputs and displaying results via simulation reports or graphical signal plots over time. While simulation is not mandatory, it is essential for medium to highly complex designs. More details on simulation tools are covered in the next section.

3. **Synthesis:** Once verified, the VHDL code is synthesized into a netlist, converting the design into FPGA-specific components such as LUTs and FFs.
4. **Design Constraints:** Separate constraint files define implementation details, including:
  - **Timing constraints:** Specify operational frequencies and expected timing behavior.
  - **I/O constraints:** Assign input and output ports to specific pins, including voltage levels and speed settings.
  - **Floorplanning constraints:** In advanced designs, these constraints guide EDA tools to place logic elements in specific FPGA regions. Proper constraint definition is crucial, as mismatches can lead to design malfunctions or hardware damage.
5. **Place and Route:** Once synthesized and constrained, the EDA tools perform automatic placement and routing. This process assigns FPGA resources such as LUTs, FFs, memory blocks, and clock managers, ensuring optimal interconnectivity while satisfying timing and I/O constraints. Multiple iterations are often performed to enhance efficiency.
6. **Gate-Level Simulation:** In some cases, further simulations are required to verify hardware behavior compared to previous behavioral simulations. Gate-level simulation considers propagation delays and distances between logic elements, making it computationally demanding. For complex designs, full gate-level simulation may not be feasible, requiring designers to target specific critical units for analysis. Fortunately, RTG4 tools exhibit strong alignment between behavioral simulations and actual hardware performance.
7. **Bitstream Generation:** After routing, the finalized netlist is compiled into a bitstream file, a binary file containing the programming information for FPGA resources.
8. **FPGA Programming:** The bitstream is loaded onto the FPGA using a hardware programmer, typically via JTAG protocol. Since the RTG4 FPGA is flash-based, its configuration remains stored in non-volatile memory, allowing it to retain its programmed state across power cycles. Some FPGA technologies require external memory to reload the design upon startup.

The synthesis and resource utilization results provided by the EDA tools confirm that both FD-FP and LDA-FP designs successfully achieved timing closure at 80 MHz, ensuring they meet the required specifications. Furthermore, the timing analysis was conducted under worst-case conditions, including high temperature, low FPGA core voltage, and the effects of FPGA degradation due to a Total Ionizing Dose (TID) of 100 krad, validating the robustness of the designs in extreme environments.

On the resource usage side, both designs fit within the resources available in the RTG4 FPGA, with the LDA-FP being considerably more constrained, occupying almost 75% of the FPGA resources. While this leaves a small margin to accommodate additional modules within the design, it should be noted that the FP design is a highly advanced architecture with few new modules left to integrate, except for the PCM interface, which is a simple module and occupies minimal FPGA resources. Additionally, the RTG4 is a hardware TMR FPGA, offering superior robustness against radiation effects, eliminating the need for extra fabric resources for radiation mitigation. The FD-FP, on the other hand, occupies only about 30% of the FPGA resources, making its implementation highly reliable and the synthesis process remarkably efficient, achieving timing closure in less than two hours without requiring additional routing iterations. The FPGA resources utilization used by both FP designs are shown in Table 4 and Table 5 for the LDA-FP and FD-FP, respectively.

| LDA-FP Resources Utilization (RTG4) |        |        |       |
|-------------------------------------|--------|--------|-------|
| Type                                | Used   | Total  | %     |
| 4LUT                                | 112438 | 151824 | 74.06 |
| DFF                                 | 81924  | 151824 | 53.96 |
| RAM64x18                            | 67     | 210    | 31.90 |
| RAM1K18                             | 166    | 209    | 79.43 |
| MACC                                | 30     | 462    | 6.49  |

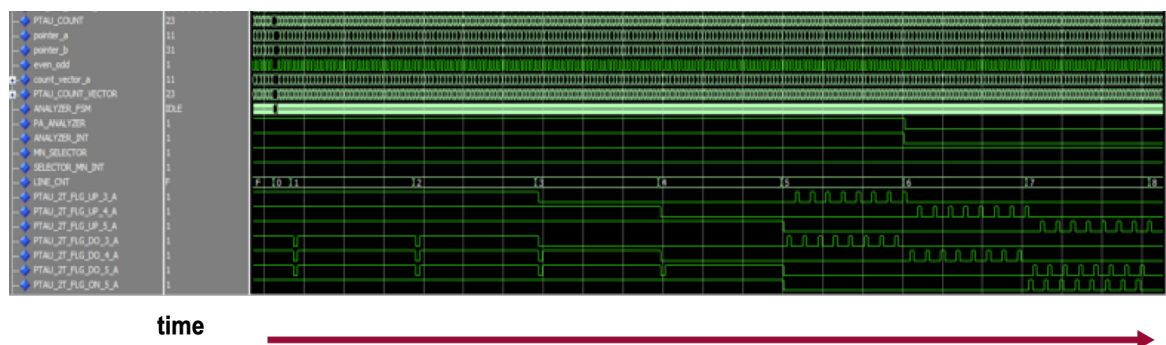
**Table 4: LDA-FP resource utilization report summary.**

| FD-FP Resources Utilization (RTG4) |       |        |       |
|------------------------------------|-------|--------|-------|
| Type                               | Used  | Total  | %     |
| 4LUT                               | 39963 | 151824 | 29.74 |
| DFF                                | 26120 | 151824 | 20.62 |
| RAM64x18                           | 3     | 210    | 1.43  |
| RAM1K18                            | 127   | 209    | 62.68 |
| MACC                               | 10    | 462    | 2.16  |

**Table 5: FD-FP resource utilization report summary.**

## 6.2 TESTBENCH SIMULATION AND VERIFICATION

As discussed in the previous subsection, simulation is a crucial step in complex VHDL designs. It allows the designer not only to test the outputs of each module but also to gain insights into the logic behavior within each module. Since VHDL is a hardware programming language rather than software, its logic executes in parallel, with each unit functioning simultaneously. This contrasts with software, where instructions are executed sequentially. The logic elements performing VHDL instructions operate concurrently. To provide an overview of the behavior of each signal inside the FPGA, simulation tools typically represent signal states graphically over time, as shown in Figure 76. As the plot progresses, the simulation reveals how each signal evolves over time, enabling the visualization of activity cycles, latching states, and, crucially, the signal behavior in case of unintended operations.



**Figure 76: Screenshot showing a typical VHDL simulation example. In the image, the SPA unit signals (one signal per row) are shown evolving over time, the simulation tool allows to observe several signals in parallel to verify the concurrent operation of the different logic units.**

The interdependence of numerous parallel signals makes the debugging and verification process of FPGA designs as complex as their development. Depending on the complexity of the unit and simulation parameters, simulating just a few milliseconds of FPGA behavior may require hours. For this reason, integrating simulation throughout the development of each individual VHDL module is essential. Thorough testing should be conducted before incorporating modules into a larger design. This process involves a combination of software instructions and HDL units that emulate the interconnections a module will encounter during operation while generating the necessary signals for its function. These testing units are known as testbenches, and when specifically designed for VHDL units, they are referred to as VHDL testbenches. The best practice in FPGA development is to alternate between module development and testbench creation. Testing each module before integrating it into a broader structure ensures that by investing time in simple tests early on, unnecessary computational and time-intensive simulations of the entire FPGA design can be avoided. This does not imply that full design simulations are avoidable, but a good balance should be found to make the development process more efficient while keeping its reliability and robustness.



Once the design units were integrated, a complete simulation environment with VHDL testbenches was developed, as shown in Figure 77. This testbench allowed the FP design to be controlled using the SpW command interface, which loaded inputs from simple command text files. These files contained the same command sequences used for hardware tests, with the difference that to reduce simulation time, the LUTs and Good/Bad RAM memories of the diagnostic unit were preloaded in the simulation.

**Figure 77: Testbench canvas for the LDA-FP. The full LDA-FP design (D3C\_IO\_FRAME) is connected to a stimulus unit which generates SpW commands, data patterns and the pixel clocks. To emulate the external SSRAM memory the SSRAM\_CONTROLLER\_TB was added creating a complete virtual environment.**

The second method utilized a larger external diagnostic unit outside the FP logic. This test primarily assessed input data synchronization with the system and pixel and frame clocks to ensure the FP could align its inputs with ADC outputs. The third method involved simple programmable patterns generated by state machines connected to the FP input. These patterns, such as basic counting sequences, enabled full-frame testing to evaluate latencies and confirm that data was neither lost nor incorrectly assigned to different frames. This method was particularly useful for assessing FP operation at varying pixel clock periods, ensuring that the pipeline remained synchronized with the pixel clock.

As mentioned previously, full FP design simulations require significant computational resources. In the case of the LDA-FP, a single millisecond of computation could take over an hour. Therefore, simulation time needed to be optimized and supplemented with hardware tests. By passing more test data through the FP pipeline, simulations were primarily used when a bug or unexpected behavior emerged. In such cases, the simulation tools allowed the replication of abnormal behavior, facilitating bug correction and helping determine whether the observed behavior was an expected part of FP operation under specific conditions.

## 6.3 HARDWARE BREAD-BOARDING

The development of the frame processor encompassed multiple stages of hardware and firmware advancements aimed at creating a robust and efficient architecture. Throughout these stages the frame processor was tested on various hardware models, each representing a progressive step towards future flight hardware. This subsection provides a concise overview of several hardware bread-boarding stages along with their most significant outcomes.

---

### 6.3.1 SMARTFUSION 2 TESTS

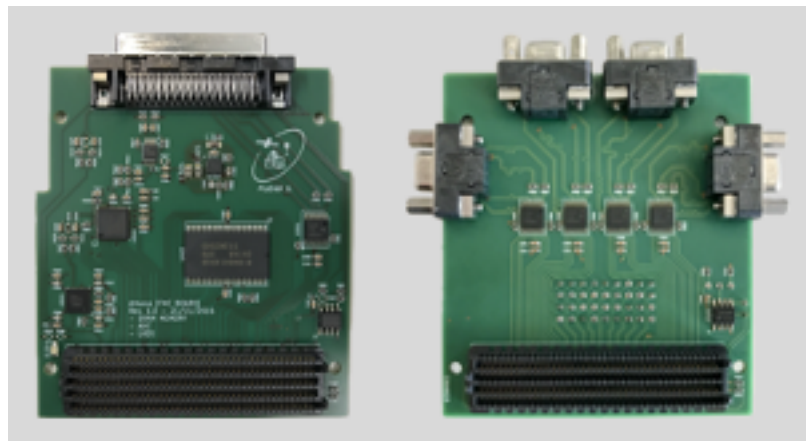
The enhanced processing platform of the frame processor traces its origins to the IAAT Digital Camera-Controller-Chip, developed for the Simbol-X mission proposal in the late 2000s [46]. This design was initially intended to become an ASIC but, as is common in ASIC development, was first implemented using FPGA technology specifically a Xilinx Virtex 4 FPGA. As this technology became obsolete and considering the planned launch of the Athena mission in 2028, the decision was made to develop the new design using the newly released RTG4 FPGA in 2016.

The transition to the new FPGA technology required evaluating the adaptability of the D3C VHDL code to a different FPGA and learning to use the newly introduced Libero SoC IDE software from Microsemi, now known as Microchip. Due to the high risks and costs associated with using a space-grade FPGA for the first time, an initial decision was made to port the D3C design to the SmartFusion 2 (SF2) FPGA which had been released alongside the RTG4 FPGA. This FPGA was manufactured by the same vendor and utilized 65 nm Flash-FPGA technology while offering the same amount of FPGA resources within its fabric. In addition to these technological similarities, the SF2 was available as a commercial product with a development kit costing less than one twentieth of that of the RTG4 FPGA while maintaining similar ports and functionalities.

This made it an optimal choice for familiarization tasks before transitioning to space-qualified hardware.

Although FPGA development kits provide a variety of useful connectivity options, it is ultimately the responsibility of the end user to develop tools that ensure application compatibility. In the case of the frame processor development, modifications were required to increase the board's memory capacity and enable SpW-like communication by attaching Micro-D connectors. Additionally, the availability of simple General Purpose Input Output (GPIO) interfaces was considered a valuable feature for gaining deeper insights into the FPGA's internal signals.

To address these requirements, two daughter boards were designed to connect to the FPGA Mezzanine Connectors (FMC) available on the SF2 Advanced Development Kit. The first board was intended to demonstrate the use of a small SRAM memory with behavior similar to the internal memory of the RTG4. This board also facilitated testing of the LVDS components necessary for incorporating a SpW interface. A second board featuring these LVDS drivers was later developed with multiple GPIO header pin ports useful for debugging, LEDs to indicate FPGA status and four SpW connectors designed to evaluate the integration of a SpW interface. Figure 78 illustrates both developed FMC boards.

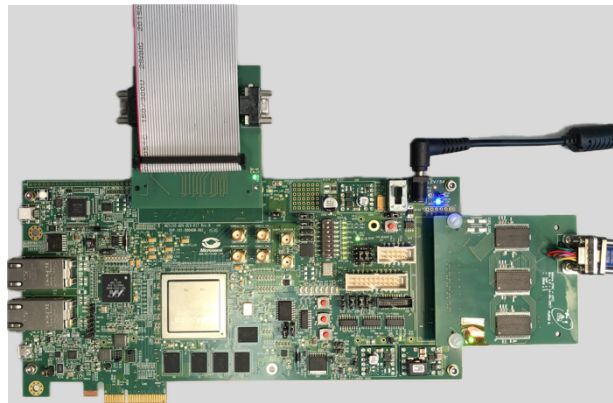


**Figure 78: Pictures of the two initially developed FMC boards: on the left the memory testing board, on the right the GPIO board.**

The two boards enabled the implementation of simple test designs using SF2 technology and the Libero SoC design flow methodologies. After successfully integrating the necessary tools, the focus shifted to porting the existing D3C pipelines into the SF2 FPGA. Due to the smaller size of the Virtex 4 FPGA, it only had sufficient resources to support four D3C pipelines. Consequently, the porting tasks involved modifying the architecture to accommodate the eight parallel pipelines required for the LDA-FP, incorporating the SpW interface, replacing all Xilinx proprietary IP cores with

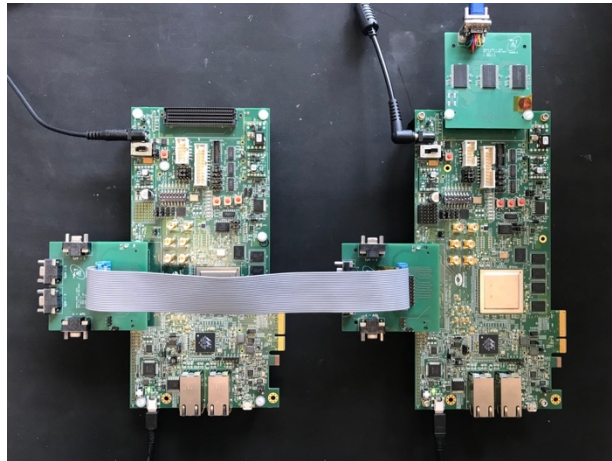
Microsemi equivalents, and making several adjustments to the original D3C module code to enhance efficiency and align it with the new FPGA fabric.

Following the successful testing of the two developed FMCs, a third FMC was designed with additional memory and an integrated SpW core. These external memory components allowed the LUTs to support a full LDA slice of 64 by 512 pixels, whereas previously the internal FPGA memory could only accommodate 64 by 64 pixels within the LUTs. With all these elements in place, the SF2 successfully completed the initial porting tests and established a solid foundation for further development.



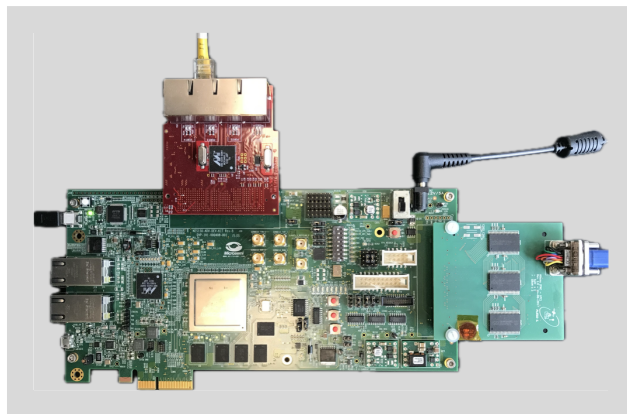
**Figure 79: Picture of the SmartFusion 2 development kit with the SpW-Memory board connected on one FMC port (right) and the GPIO board connected on the other port (top).**

To further enhance the complexity of these tests and enable monitoring of data transfers within the SF2 pipelines, a second SF2 FPGA was introduced. This FPGA was programmed with a simplified version of the diagnostic memory system from the frame processor design and equipped with a basic UART serial interface. It was configured to transmit test frames to the primary SF2, allowing for the first-ever validation of the pipeline using an external input. This setup ensured the capability to synchronize with an incoming pixel stream. Figure 80 presents an image of the experimental arrangement.



**Figure 80: Testing setup with two SF2 development kits, one SF2 contains a simple diagnostic system (left) and sends data to the FP board to be processed and output the processed data via SpW (right).**

Additionally, leveraging the Serializer-Deserializer (SerDes) interfaces integrated into the SF2, which facilitate high-speed data transmission, the FPGA was utilized to develop an initial version of the High-Speed Serial Interface. This implementation allowed the transfer of raw input data from up to two pipelines in parallel, significantly enhancing the debugging and testing capabilities of the frame processor while laying the groundwork for more advanced future evaluations.



**Figure 81: The SmartFusion 2 development kit with the SpW-Memory board (right) and the Gigabit-Ethernet FMC board used for the HSSI (top).**

---

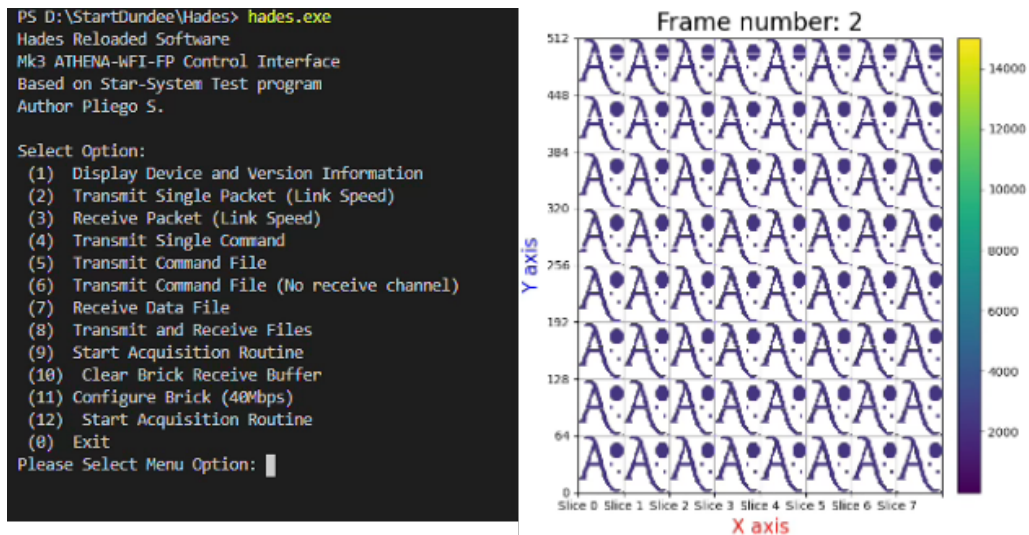
### 6.3.2 RTG4 DEVELOPMENT KIT TESTS

After conducting tests with the SmartFusion 2 and gaining a thorough understanding of the Libero SoC development flow, the design was ported to the RTG4 using the commercially available RTG4 Development Kit. This development board shares many features with the SF2 kit, including two FMC ports, but contains an RTG4-Proto FPGA. The RTG4-Proto has the same silicon as the space-grade RTG4 FPGA but lacks the space-grade screening, making it functionally identical to the RTG4 and an ideal platform for testing and developing digital designs for RTG4 hardware.

Porting between FPGAs from the same vendor and with similar fabric technologies is generally simpler than porting between different vendors. However, the RTG4 features a triple-module-redundant architecture, where each flip-flop is triplicated to enhance reliability. This redundancy, while beneficial for fault tolerance, introduces a penalty on maximum operating frequency due to additional flip-flops occupying more space and the inclusion of majority voters at the outputs of these modules. As a result, although the silicon theoretically maintains the same speed at 65 nm, the extra embedded logic causes the RTG4 to perform at a lower effective speed. While the frame processor operated at 80 MHz on the SF2, consuming half of its resources, it only reached 50 MHz on the RTG4 and required extensive place and route optimization.

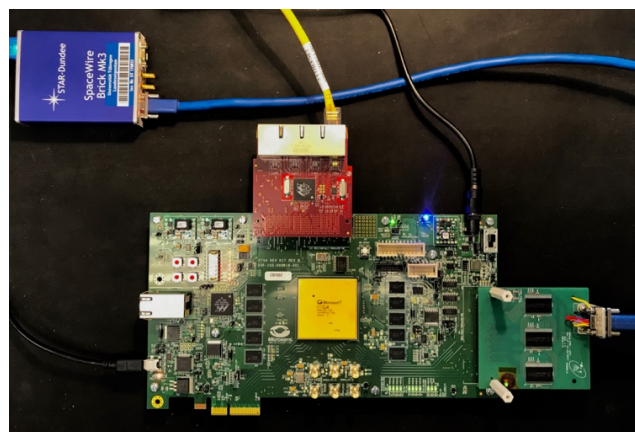
This transition necessitated a major redesign of all pipeline units. Additionally, some components such as the LTG and CTU, along with several new features, were yet to be integrated into the frame processor. The redesign involved modifying every algorithm in the VHDL code, with the exception of the PCU and EFU units, which rely on simple comparisons. These improvements also led to significant advancements in development and testing tools, including the migration of the SpW interface to the industry-standard Star Dundee testing hardware. To facilitate easy and efficient communication with the frame processor, a new control software known as Hades was developed, featuring rapid visualization tools that allow quick status monitoring. Figure 82 presents an illustration of the updated system.





**Figure 82: Early version of the hades command line interface (left), the quick frame visualizer showing diagnostic data with the Athena logo.**

Integrating the High-Speed Serial Interface into the testing setup allowed for hardware-in-the-loop-style evaluations, where test patterns were fed into the frame processor. The input data was captured via the High-Speed Serial Interface, while the output data, transmitted through the SpW interface, was analyzed to verify the performance of the frame processor. Figure 83 depicts the testing arrangement, with the RTG4 Development Kit positioned at the center. The High-Speed Serial Interface, featuring an FMC Gigabit Ethernet modem highlighted in red, records the input data, while the SpW brick, shown in blue, receives the processed output using the SpW-Memory FMC.



**Figure 83: The RTG4 Development Kit board with the Gigabit-Ethernet FMC (top) and SpW-Memory FMC (right) connected. The SpW brick (top left) receives the preprocessed data.**

### 6.3.3 FRAME PROCESSOR MOTHER-BOARD TESTS

Having gained extensive experience with the RTG4 and confirmed its selection as the primary hardware, the next phase of testing required specialized equipment capable of evaluating the frame processor's features in greater detail. This setup needed to accommodate all necessary peripherals and ports for its operation, including memory large enough to store the LDA-FP lookup tables, differential transceivers for sequencing the camera head electronics, and analog-to-digital converters for digitizing the camera head signals.

To support this phase, engineers at MPE, with input from IAAT, developed the Frame Processor Mother Board (FP-MB). Designed specifically for LDA-FP development and testing, this board incorporated six FMC connectors, each serving a distinct purpose: a Star-Dundee SpW FMC equipped with LVDS transceivers and SpW connectors, an SSRAM FMC containing space-grade memory sufficient to store LDA-FP lookup tables, an ADC FMC with eight analog-to-digital converters for processing camera head signals, a buffer FMC for interfacing with the camera head ASICs, and two FMCs each hosting a Gigabit Ethernet PHY for extracting raw input data via the HSSI.

Two identical FP-MSs were produced, one stationed at IAAT (Figure 84) for the frame processor development team and the other at MPE for the testing team. This arrangement not only enhanced efficiency in debugging and testing by utilizing a shared hardware platform but also ensured compliance with an important ESA requirement mandating that design development and verification be conducted by independent teams.

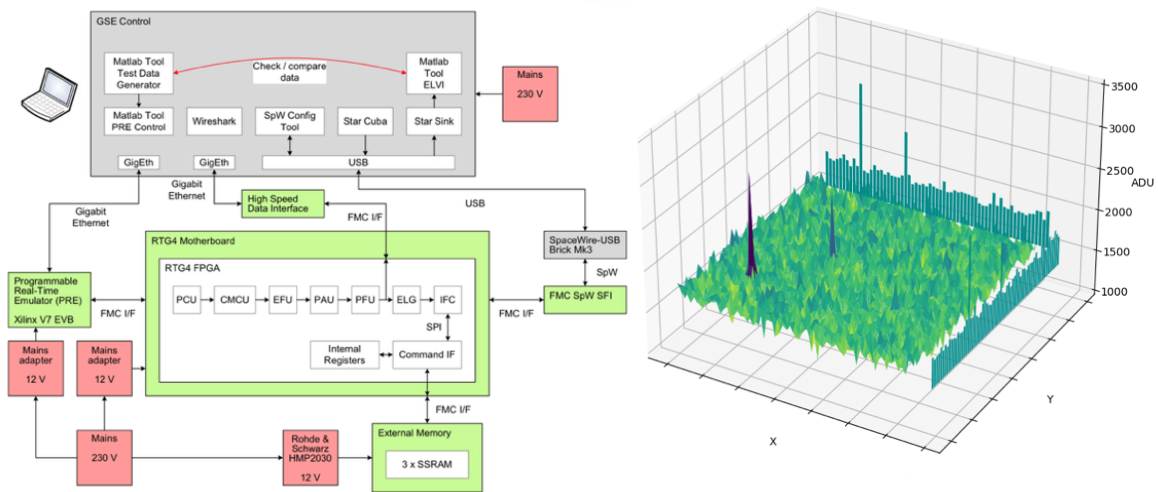


**Figure 84: Real-time performance testing setup operating at IAAT. The FP-MB receives data from the Virtex 7 development board (top) through a flex-rigid PCB connector. Output data is received through the SpW interface for its analysis.**



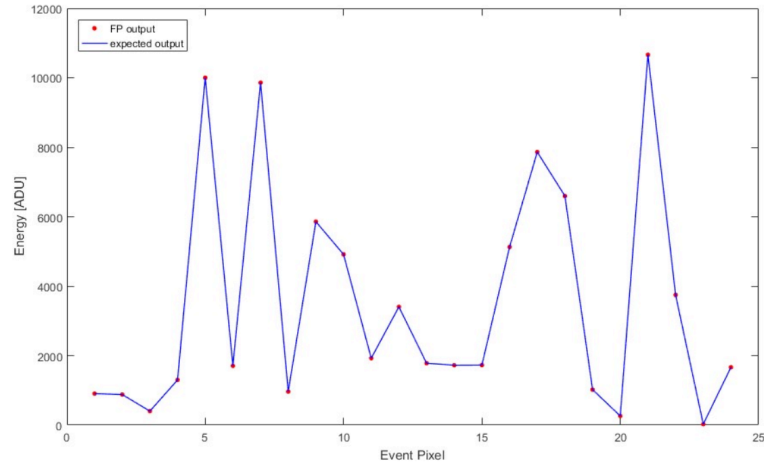
### 6.3.3.1 REAL-TIME PERFORMANCE VERIFICATION

Expanding the development environment further, an external Virtex 7 development board was employed to host a Programmable Real-time Emulator (PRE) for the camera head [39]. The programmable real-time emulator allows realistic data to be inserted into the frame processor without requiring an actual detector, providing controlled testing conditions. A MATLAB program was used to simulate various observational scenarios that the LDA-FP and FD-FP might encounter, considering different data rates and incorporating MIP events derived from the WFI requirements. The memory within the emulator stored up to eight distinct realistic LDA frames, which were cyclically transmitted to the FP-MB. With data rates reaching 195 pixels for the LDA-FP and 31 pixels for the FD-FP, this setup enabled the validation of the real-time processing capabilities of the frame processor using hardware with the same expected speed performance as the flight model, Figure 85 shows a block diagram of the testing setup. Additionally, it confirmed that the memory bandwidth required for reading external memory for the LDA-FP was achievable with available flight-ready technology.



**Figure 85: Block diagram showing the test setup used for the real-time performance verification tests of the FP (left). Sample 3D visualization of the frame data generated by the MATLAB software (right).**

The Matlab program also generated event lists containing expected output data, including preprocessing pipeline corrections. These results were compared with data received through the frame processor's SpW interface, allowing for verification of both real-time performance and output accuracy. The findings showed complete alignment between expected and actual output (see Figure 86), demonstrating the frame processor's ability to operate in real-time conditions.



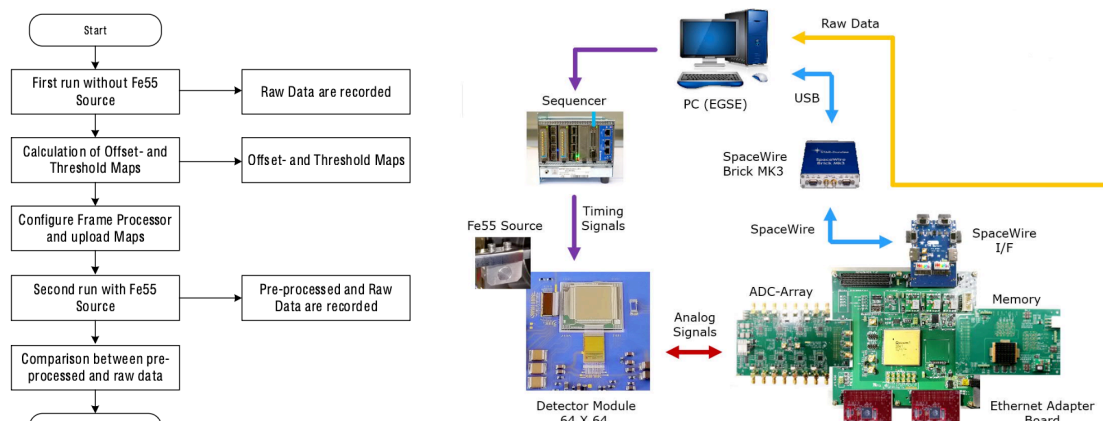
**Figure 86: Comparison between the FP output and expected output data, showing complete agreement [39].**

#### 6.3.3.2 DETECTOR RAW-DATA PERFORMANCE VERIFICATION

To further expand the verification tests on the frame processor, an Electrical Ground Support Equipment system was designed to evaluate its operation with a real detector exposed to an X-ray source. The test setup utilized the FP-MB to process data received from a 64 by 64 prototype DEPFET detector subjected to an Fe55 source. An ADC FMC board, containing commercial versions of the selected space-grade ADC was employed to digitize the signals from the detector.

To manage the front-end readout electronics, an external sequencer module was integrated into the system. This module also generated clock signals for the ADCs, which were then relayed to the frame processor to establish its internal clock, ensuring a unified synchronous clocking source throughout the setup. Incoming data was recorded in real-time using the HSSI, allowing transmission of the raw input data. The data was then processed through the frame processor's preprocessing pipeline and transmitted via SpW to the SpW brick, which forwarded the processed output to the same computer receiving the original input data.

The test followed five distinct steps. First, raw data was recorded using the HSSI without the Fe55 source. This dataset was then used to generate offset and noise maps for the preprocessing pipeline, which were stored in the external memory of the FP-MB via the SpW interface. Subsequently, new data was recorded with the Fe55 source, capturing both the raw input and the preprocessed output simultaneously. As the FD-FP was still under development at the time, the LDA-FP operated in a 64 x 64 window-mode. Finally, the raw input data was processed using offline tools and compared with the frame processor's output, following the same validation approach as previously conducted with the programmable real-time emulator.



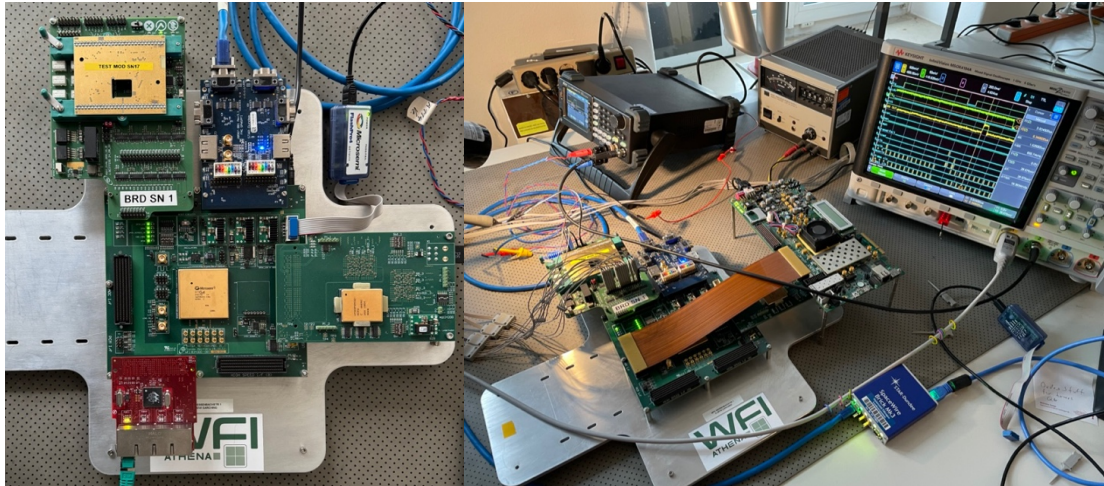
**Figure 87: Testing procedure followed for the performance verification test (left). Schematic depicting the testing setup and data flow (right) [23].**

During the test, a total of 23,000 raw data frames were captured, containing 1,233,661 events. Among these, only one processed frame exhibited an unexpected event with a low ADU amplitude, while all other events aligned precisely with expectations when compared to the raw data input. In subsequent tests, the identified error frame was processed again, and the unexpected event was no longer present. Given that the event's amplitude was below the event threshold but above the neighbor threshold, it was likely caused by a synchronization mismatch in the external SSRAM memory readout. Since this issue stemmed from a memory synchronization problem with components scheduled for replacement, the verification tests were deemed successful, and no further investigation was required.

### 6.3.3.3 ASIC INTERFACE VERIFICATION

The frame processor is responsible for programming and controlling the front-end ASICs of the WFI camera head. This requires generating SPI signals to configure the Veritas ASICs, producing a signal sequence for the Switcher ASICs to activate and deactivate detector rows, and synchronizing this row switching with the Veritas ASICs readout. Since these front-end ASICs are highly sensitive and designed for high-performance operations, it was necessary to test the frame processor's sequencer interface to verify its ability to program the ASICs correctly before connecting it to a valuable detector prototype.

To conduct these tests, an FMC containing a Veritas and a Switcher ASIC was developed by MPE and integrated into the Frame Processor Mother Board at IAAT. This ASIC board featured output ports that allowed recording and visualizing the signals in parallel using a laboratory logic analyzer.

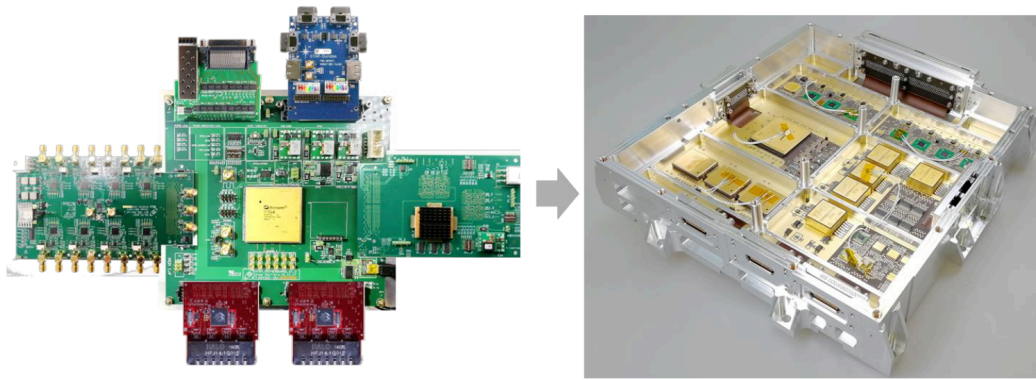


**Figure 88: The FP-MB with the ASICs FMC (top left). Test setup for the ASICs interface; a logic analyzer allows the visualization of the programming and sequencing signals. A waveform generator (black) emulates the analog signals coming from the CH.**

The setup successfully validated the frame processor’s ability to program the Veritas ASIC, confirming compliance with the digital interface specifications. Subsequently, the more complex analog interface of the Veritas ASIC was activated. Using a programmable laboratory waveform generator synchronized to the sequencer signals, the operation and sequencing of both the Veritas and Switcher ASICs were verified. The test also included monitoring self-diagnostic health signals, which provided insight into the ASICs' status. These results demonstrated the frame processor’s capability to program, sequence, and monitor the front-end electronics of the WFI camera head.

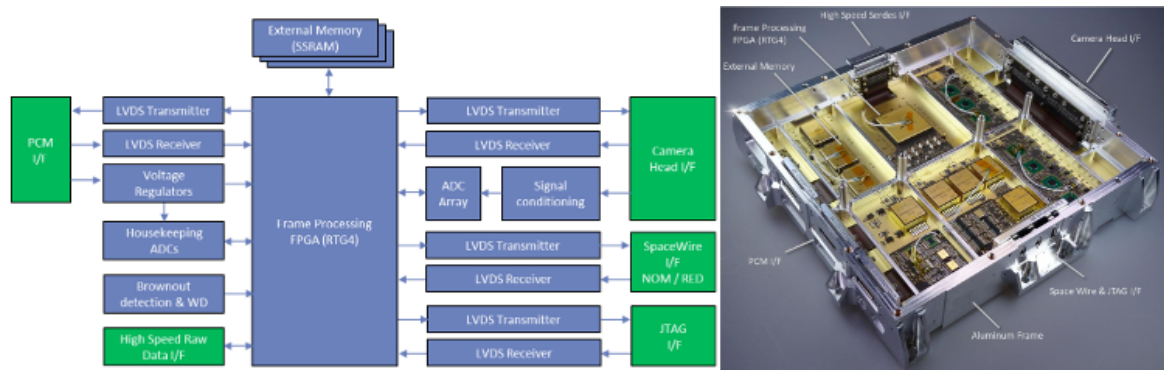
#### 6.4 PRE-EM HARDWARE INTEGRATION TESTS

Having completed extensive testing on the LDA-FP and FD-FP designs and successfully demonstrated the real-time processing capabilities of the frame processor operating on the RTG4 FPGA, along with validation of its interfaces and debugging functions, the Frame Processor Module hardware was ready to advance to a flight-like Preliminary Engineering Model (PRE-EM). This development integrated all the features of the FP-MB into a single rigid-flex PCB enclosed in a prototype aluminum frame, designed to facilitate environmental testing under conditions representative of an actual mission.



**Figure 89: Step from the FP-MB (left) to the PRE-EM (right) [40].**

The Preliminary Engineering Model (PRE-EM) was equipped with interfaces for all required subsystems, including the Payload Control Module, Camera Head System, and Instrument Control System. Additionally, it incorporated debugging and programming interfaces such as the High-Speed Serial Interface and JTAG, along with several internal auxiliary components essential for frame processor operation. These included lookup table-hosting SSRAM memory, differential transceivers, analog-to-digital converters, and voltage regulators supplying power to the FPGA and its peripherals. Figure 90 presents a schematic of the Preliminary Engineering Model architecture, where external interfaces are marked in green and internal modules in blue.



**Figure 90: Schematic block diagram showing the architecture of the PRE-EM with the RTG4 as a central element (left). Picture of the FPM PRE-EM showing its interfaces (right) [15].**

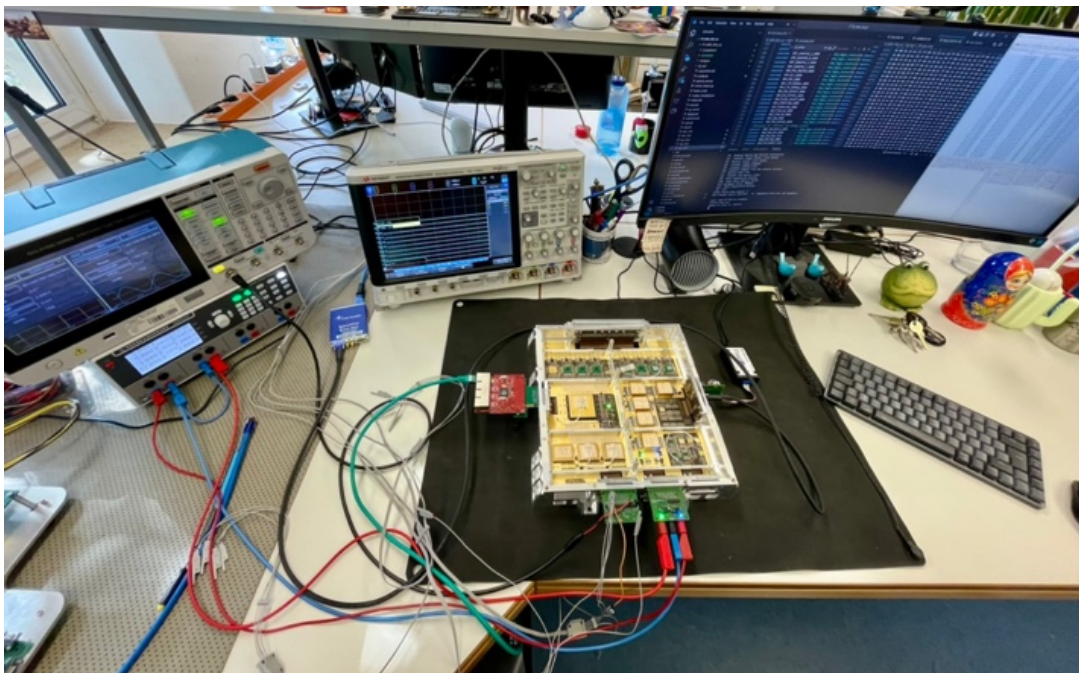
A particularly noteworthy feature of the JTAG programming interface is that, although primarily designed for ground-based programming and debugging, it is expected to be accessible to the ICS, enabling in-flight reprogramming capabilities. While modifying an FPGA's configuration in-flight was initially considered risky, the



RTG4 has demonstrated that such operations can be executed reliably without significant concerns. This capability allows for in-flight adjustments to the frame processor's design in response to evolving mission requirements.

#### 6.4.1 PRE-EM FIRMWARE INTEGRATION TEST

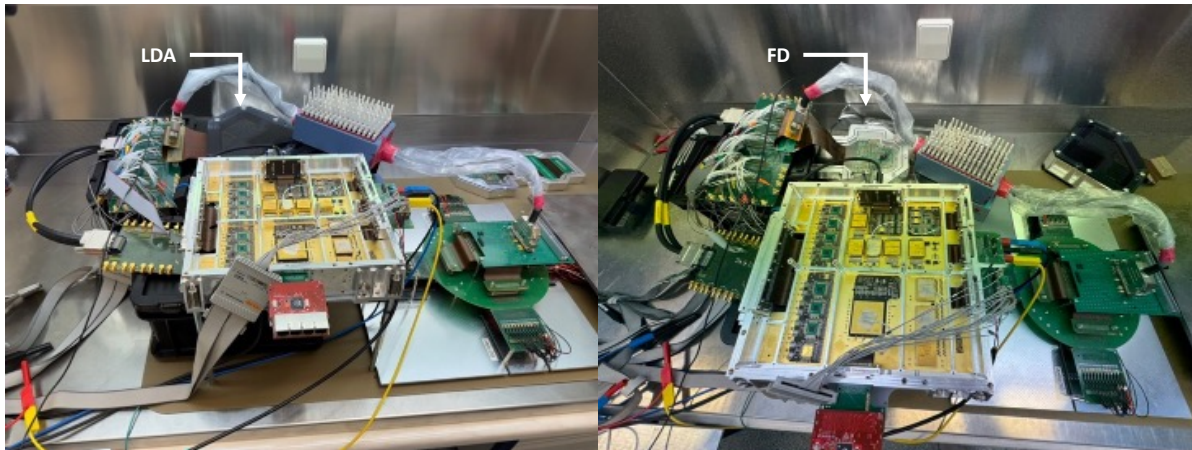
Integration of the FP firmware into the FPM PRE-EM hardware necessitated small modifications to the frame processor firmware design mainly encompassing pinout modifications and changes to the clocking network. To ensure that these changes were carried out in a safe and successful manner, a small integration campaign was carried out at IAAT taking advantage of the frame processors development equipment where the PRE-EM was successfully programmed. During this test, the PRE-EM was loaded with the full LDA-FP including the HSSI and ASICs interfaces which were tested using the internal diagnostic memory subsystem and confirming the correct data output both through the SpW interface and the HSSI. Figure 91 shows the PRE-EM hardware being tested at IAAT where data is successfully being retrieved and captured through the HSSI interface.



**Figure 91: Test setup for the first firmware integration of the PRE-EM conducted at IAAT. The PRE-EM hardware (center) is connected to the laboratory power supply and a waveform generator generating the clocking signals driving the hardware. A logic analyzer was used to check the CHS signals while the testing computer enabled data capture through the SpW brick and gigabit-ethernet port for the HSSI.**

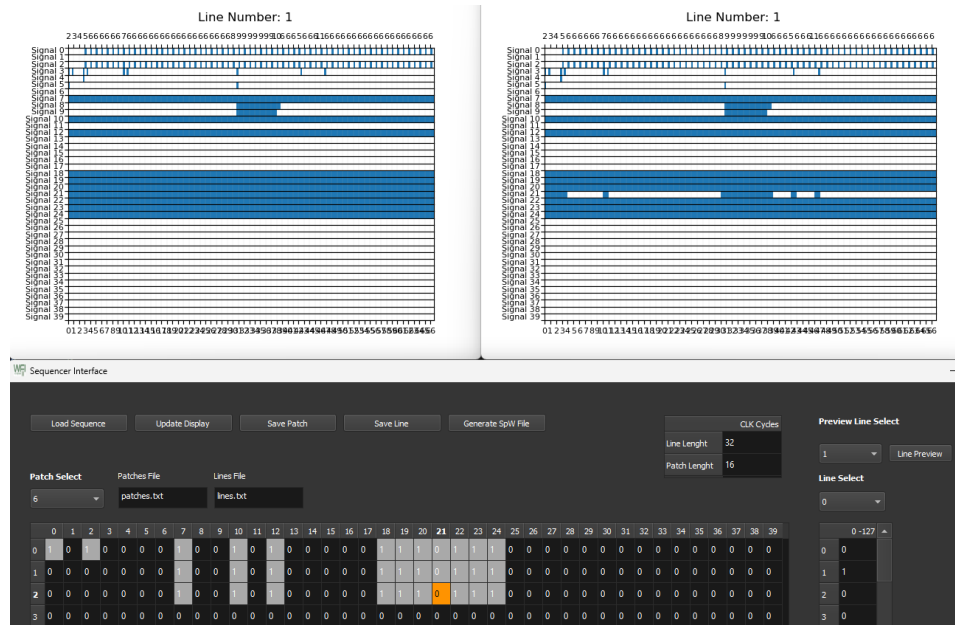
#### 6.4.2 CAMERA HEAD PRE-EM INTEGRATION TEST

During the second integration campaign, conducted at the MPE clean room facilities in Garching, the focus was on finalizing tests of the camera head interfaces and verifying the newly developed memory-based sequencer interface. By utilizing an automated method to generate sequencer programming files from an Excel-based system, the interface allowed for easy parametric modifications of detector readout conditions, including adjustments to integration and readout times.



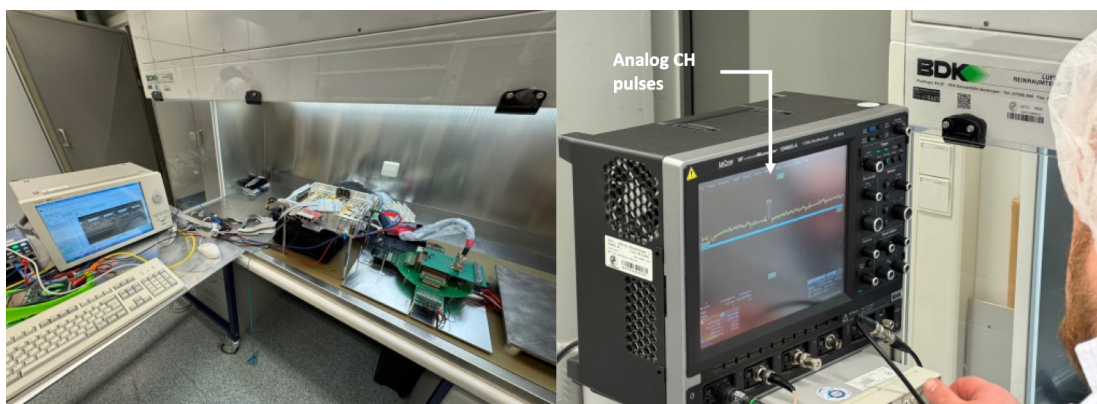
**Figure 92: LDA testing setup (left) and FD testing setup (right) inside the laminar flow chamber. The LDA and FD camera head prototypes can be appreciated on the top.**

For this test, the Preliminary Engineering Model was connected to the LDA and FD prototypes, enclosed within an aluminum frame and fitted with a screen protection to prevent contamination, Figure 92. A meticulous inspection of the sequencer signals was performed for both the LDA-FP and FD-FP designs using a laboratory logic analyzer. The programming signals were successfully generated via the sequencer interface software, enabling visual modifications of the Veritas and sequencer programming files with ease, see Figure 93.



**Figure 93: Window showing an example of a patch being modified. The original line is shown on the left, and the modified one is on the right. On the bottom, the patch visualizer shows the modified patch six.**

Upon completing the validation of the sequencer signals and confirming the correct programming of the Veritas ASICs, the analog chain of the camera head prototypes was activated. This resulted in the successful acquisition of analog pulses with amplitudes accurately modulated according to the acquisition parameters (see Figure 94). These results marked the successful integration of the LDA-FP and FD-FP designs with the PRE-EM and CH hardware, ensuring their readiness for further validation and environmental testing.



**Figure 94: Testing setup with the laboratory logic analyzer recording the camera head signals (left). An oscilloscope recorded the successful acquisition of analog signals pulses on the specified channels coming from the CH (right).**



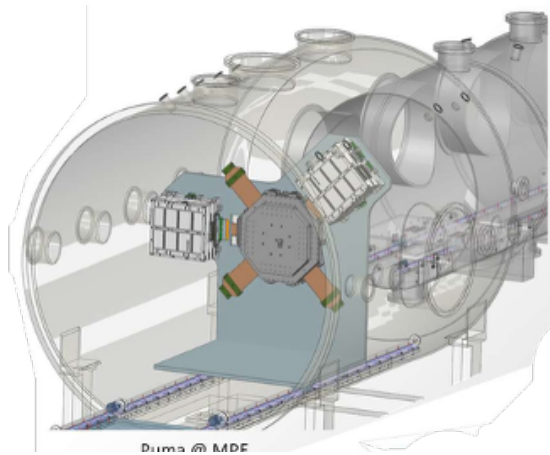
## 7 OUTLOOK AND CONCLUSION

Following the successful completion of verification activities for the Preliminary Engineering Model of the Frame Processor Module, the focus now shifts toward design validation. This phase will involve more comprehensive testing to confirm that the frame processor design performs under representative operational conditions.

In this chapter, a brief description of the next steps for the FP testing activities is given, together with an outlook of future activities towards the FPM Engineering Model and qualification activities, along with a brief conclusion on the major achievements of the FP development activities.

### 7.1 PRE-EM VALIDATION TESTS

To validate the operation of the DES PRE-EM under representative operational conditions, a testing campaign involving the full WFI acquisition chain, including the CHS and DES, will be conducted at the MPE PUMA facility. During this test, the entire acquisition chain will be evaluated while exposed to an X-ray source within the PUMA vacuum chamber. This setup includes a prototype model of a large detector quadrant and a fast detector, as well as two detector electronics modules, one for the LDA and one for the FD. The models will be installed inside the PUMA vacuum chamber as shown in Figure 95, and X-ray data will be collected using an EGSE system. This system, via the SpW and HSSI interfaces, will acquire both processed and raw data to enable full validation of the frame processor.



**Figure 95: Mounting of the camera head and detector electronics subsystems inside the PUMA facility vacuum chamber [26].**

To assess spectral performance under representative conditions, the FPM and camera head will be powered by the PCM through the GIM, with the EGSE supplying

power to the GIM. These validation tests will confirm the proper functioning of the WFI instrument's signal acquisition chain, marking a significant milestone on the path toward mission adoption. Additionally, during this campaign, the integration of a buffering module, which enables large separations between the CHS and DES, will be tested. This will provide a detailed insight into the WFI instrument's performance under different payload module configurations, offering flexibility within the tight constraints of the sophisticated spacecraft design.

## 7.2 NEW TECHNOLOGIES

While the frame processor has demonstrated its capability to process in real time the data from the state-of-the-art WFI instrument, its robust design has significantly reduced the criticality and risks associated with this subsystem, proving ahead of schedule its readiness in many successful ways. Nevertheless, it must be acknowledged that the planned launch date of the Athena observatory is still more than a decade away. Failing to build upon the solid baseline already developed, one that allows for improvements and further advancements, would represent a missed opportunity that a flagship instrument cannot afford.

With this in mind, exploring new technologies that could enhance the performance and capabilities of the frame processor is essential. Within this scope, two primary research areas come into focus:

New space-grade FPGAs have emerged in recent years offering increased resources and improved technological performance. Among these, two standout candidates are the Microchip RT PolarFire FPGA [47], which offers higher speed, lower power consumption, and more logic resources compared to the RTG4 FPGA. These advantages come at the cost of some reliability features, such as hardware-based triple modular redundancy, which could be mitigated using radiation-hardening techniques. The second candidate is the NanoXplore NG-Ultra 300 FPGA [48], a European-developed device based on robust 28nm STM-FDSOI silicon technology. It also offers improved performance and power efficiency, nearly doubling the logic resources of the RTG4, while facing the same redundancy limitations as the PolarFire. A major advantage of both technologies is their significantly larger internal memory, which could eliminate the need for external memories to store the LUTs used by the PCUs and EFUs. This would simplify the printed circuit board design, significantly reduce power consumption and thermal dissipation, and enhance the LTG processing speed in the LDA-FP. Furthermore, the SSRAMs used in the PRE-EM, now obsolete, would no longer need to be replaced, sparing substantial hardware development effort.

A second clear area for research lies in substituting certain frame processor units with neural network-based solutions. Manufacturers like Microchip now offer neural network acceleration cores with proven capabilities in FPGA devices. These solutions are considerably more efficient than traditional hardware such as GPUs, since FPGAs can optimize their resources specifically for the required tasks. Employing machine learning algorithms in the frame processor could offer improved noise subtraction, particularly in the early stages of the processing pipeline. This approach might even allow for the replacement of the CTU, PCU, and CMU with a single neural network acceleration engine. Although the results of these studies are still at an early stage and open to further development and discussion, this is an opportunity that should not be overlooked.

### 7.3 CONCLUSION

The development of the frame processor for the Athena Wide Field Imager (WFI) represents a significant achievement in the field of space-based X-ray instrumentation. This work has demonstrated the feasibility and robustness of a highly parallelized, real-time data processing architecture capable of meeting the demanding requirements of the next-generation Athena observatory. The frame processor was conceived and realized as a modular, technology-independent solution, leveraging the flexibility and performance of radiation-hardened FPGAs to address the unprecedented data rates generated by the WFI's state-of-the-art DEPFET sensors.

A central outcome of this project is the successful implementation of a demonstrator model that processes and filters raw detector data in real time, reducing the data volume to within the strict telemetry constraints of modern spacecraft. The processor's architecture, based on a pipeline of specialized units, enables efficient event detection, noise suppression, and pattern recognition, ensuring that only scientifically valuable X-ray events are transmitted to the ground. The use of parallel pipelines and dedicated memory management strategies has proven essential for handling the high throughput required by both the large detector array and the fast detector subsystems.

Extensive verification and validation activities, including simulation, hardware-in-the-loop testing, and integration with prototype detector electronics, have confirmed the processor's ability to operate reliably under representative mission conditions. The design has achieved timing closure at the target operating frequency, and resource utilization studies have shown that the implementation fits comfortably within the constraints of the selected RTG4 FPGA, with ample margin for future enhancements. The processor's modularity and configurability have also facilitated rapid adaptation to

evolving instrument requirements and technology upgrades, positioning the WFI electronics for long-term maintainability and scalability.

The successful development of the frame processor not only addresses the immediate needs of the Athena WFI instrument but also establishes a solid foundation for future high-performance space-borne data processing systems. The methodologies and architectural principles established in this work can be extended to other scientific missions facing similar challenges of high data rates and limited downlink capacity. Looking ahead, the integration of emerging FPGA technologies and the exploration of machine learning-based processing units offer promising avenues for further enhancing the capabilities and efficiency of on-board data processing.

In summary, the results presented in this thesis show that the frame processor can meet the scientific and technical requirements of the Athena mission. The design, implementation, and validation achievements mark a significant step forward in realizing cutting-edge X-ray astronomy instrumentation and ensure that Athena will be equipped to provide a new perspective on the hot and energetic universe.

## APPENDIX A: FD INTERFACE CONTROLLER OPERATION

Data transfers to the FP module are carried via the IFC's SpW interface. Each SpW packet is generally composed of 7 bytes with the following structure, where the meaning of each byte is displayed in Table 6.

0xXX 0XXXXX 0XXXXX 0XXXXX EOP

| Bytes | Value  | Interface         |
|-------|--------|-------------------|
| 1     | 0xXX   | Interface address |
| 2-3   | 0XXXXX | Address           |
| 4-5   | 0XXXXX | Send/Receive      |
| 6-7   | 0XXXXX | CMD/Data          |

**Table 6: 7-byte command packet structure contents.**

The first byte of the package corresponds to the interface address. This character allows the IFC to direct the received information to its respective interface. Possible values are given in Table 7.

| Value | Interface              |
|-------|------------------------|
| 0x33  | Housekeeping Interface |
| 0x34  | Command Interface      |
| 0x35  | Science Interface      |
| 0x20  | Echo Interface         |

**Table 7: Interface address map.**

The second and third bytes correspond to the address of the register to be written. A detailed description of this field is given in the following sections.

The fourth and fifth bytes set the command direction. In the case of send, the information contained in the CMD-data bytes (6-7) is written in the address indicated by the address bytes (2-3). When the value corresponds to receive, the CMD-data bytes are

ignored and instead the interface will reply to the command interface with the value stored in the register indicated by the address field.

| Value  | Action       |
|--------|--------------|
| 0x4182 | Send data    |
| 0xC003 | Receive data |

**Table 8: Send/Receive possible values.**

The remaining bytes (6-7) contain the information to be written on the registers in case of the send data command. In receive data commands, these bytes are ignored.

## HOUSEKEEPING INTERFACE

The housekeeping interface controls data transfers to and from the HKU, described in the previous section. The HKU uses a RAM memory as an asynchronous interface for all the read and write operations, and therefore all transactions are made with the same memory range. Therefore, this interface's transactions are handled as single multiple-word registers.

| Register   | Address (range) | Access type | Default value |
|------------|-----------------|-------------|---------------|
| HKU_MEMORY | 2000h -203Fh    | R/W         | NA            |

**Table 9: Register list of the housekeeping interface.**

## COMMAND INTERFACE

The command interface contains most of the registers necessary to operate the FP. The registers can be divided into three sub-groups: the EPP registers, those used for the EPP operation; communications and debug registers, used to handle the FP communication interfaces; and sequencer registers, controlling the SEQ unit. A list of each sub-group's registers is given in this subsection together with a short register description.

| EPP Registers           |                 |             |               |
|-------------------------|-----------------|-------------|---------------|
| Register                | Address (range) | Access type | Default value |
| <b>EPP_UPPER_THRHLD</b> | 7000h           | R/W         | 36B0h         |
| <b>EPP_LOWER_THRHLD</b> | 7001h           | R/W         | 000Ch         |
| <b>EPP_PFU_MASK</b>     | 7002h           | R/W         | FFFFh         |
| <b>EPP_MODE</b>         | 7005h           | R           | NA            |
| <b>EPP_DIAG_RATIO</b>   | 7007h           | R/W         | 0800h         |
| <b>EPP_SELECT</b>       | 7009h           | R/W         | 0000h         |
| <b>EPP_PFU_SWITCH</b>   | 7020h           | R/W         | FFFFh         |
| <b>EPP_MLT</b>          | 7021h           | R/W         | 0000h         |
| <b>EPP_MHT</b>          | 7022h           | R/W         | FFFFh         |
| <b>EPP_LTP_SIGMA1</b>   | 7024h           | R/W         | 0005h         |
| <b>EPP_LTP_SIGMA2</b>   | 7025h           | R/W         | 0003h         |
| <b>EPP_FRAME_SIZE</b>   | 7034h           | R/W         | 0001h         |
| <b>EPP_CTU_R1</b>       | 7040h           | R/W         | 0000h         |
| <b>EPP_CTU_R2</b>       | 7042h           | R/W         | 0000h         |
| <b>EPP_CMD</b>          | 7100h           | R/W         | NA            |
| <b>EPP_PFU_BYPASS</b>   | 7200h           | R/W         | 0000h         |

**Table 10: Registers list of the command interface FD EPP registers.**

## **EPP\_UPPER\_THRHLD/ EPP\_LOWER\_THRHLD**

Event preprocessing pipeline upper threshold/ Event preprocessing pipeline lower threshold: The upper threshold register holds the threshold energy above which an event is considered a MIP. This value is used by the PCU unit to set the UT flag to 0 in case the pixel energy is above this energy threshold value. The lower threshold register holds the threshold energy below which an event is considered a MISFIT. This value is used by the PCU unit to set the LT flag to 0 in case the pixel energy is below this energy threshold value.

## **EPP\_PFU\_MASK**

This register contains a mask used by the PFU to decide which pixels are marked as valid. Each bit on the mask corresponds to a pixel flag. If the pixel flags and the mask flags are all equal, then the pixel is marked as valid. Internal register structure can be seen in the PFU sub-section.

## **EPP\_MODE**

Event preprocessing pipeline operation mode: This read-only register allows us to request the status of the main operation register of the FP.

## **EPP\_DIAG\_RATIO**

Event preprocessing pipeline diagnostic good-frame/bad-frame ratio: This register sets the ratio between good and bad frames to be entered to the pipeline during diagnostic mode operation. The XXXXh value stored in the register can be read as follows: XXXXh is the number of “bad-frames” entered to the pipeline per one “good-frame”.

### **EPP\_MAP\_SELECT**

Event preprocessing pipeline memory select: This register is used to point to the different memories inside the FP, each value corresponds to a different memory, as shown in the Table 11 below.

| Value  | Selected Memory             |
|--------|-----------------------------|
| 0x0000 | Offset LUT                  |
| 0x0001 | Event filtering threshold 1 |
| 0x0002 | Event filtering threshold 2 |
| 0x003C | Good-frame diagnostic RAM   |
| 0x003D | Bad-frame diagnostic RAM    |

**Table 11: Memory selection address map.**

Once the desired memory has been selected on register 7009h, its contents are accessible in addresses 8000h-FFFFh as seen in table 10:

| Memory                      | Address range |
|-----------------------------|---------------|
| Offset LUT                  | 8000h-8FFFh   |
| Event filtering threshold 1 | 8000h-8FFFh   |
| Event filtering threshold 2 | 8000h-8FFFh   |
| Good-frame diagnostic RAM   | 8000h-8FFFh   |
| Bad-frame diagnostic RAM    | 8000h-8FFFh   |

**Table 12: Address range available to each memory.**

### **EPP\_PFU\_SWITCH**



Event preprocessing pipeline pixel filtering unit switch: This register contains information on which stages of the pipeline are switched ON/OFF, depending on the respective bit on the switch; different stages will stop flagging pixels or, in the case of the raw data bit, the complete pipeline can be set OFF. Internal register is shown in the PFU sub-section.

#### **EPP\_MLT / EPP\_MHT**

Event preprocessing pipeline bad-median lower threshold / Event preprocessing pipeline bad-median higher threshold: The lower threshold register stores the threshold value XXXXh below which, if the median found by the CMU is lower, all pixels of the line are marked with BM flag. The higher threshold register stores the threshold value XXXXh above which, if the median found by the CMU is higher, all pixels of the line are marked with BM flag.

#### **EPP\_LTP\_SIGMA1 / EPP\_LTP\_SIGMA2**

Event preprocessing pipeline look-up-table programmer sigma 1 / Event preprocessing pipeline look-up-table programmer sigma 2: These registers store the factor by which the LTP multiplies the noise (variance) of each pixel to give the first and second threshold values used on the EFU. Therefore, 1T = XXXXh \* sigma and 2T = XXXXh \* sigma.

#### **EPP\_FRAME\_SIZE**

Event preprocessing pipeline frame size: This register selects between full-frame and window mode for the FD readout. Setting the register to 0x0001 will set the pipeline to read out the full detector 64x64 pixels, while 0x0000 will activate window mode, reading out the inner 32 rows of the FD sensor (32x64 pixels).

#### **EPP\_CTU\_R1 / EPP\_CTU\_R2**

Event preprocessing pipeline CTU register 1/ Event preprocessing pipeline CTU register 2: These register sets the first and second correction values for the CTU. The value set by the first 8 bits of this register is divided by 8192 and multiplied by the energy value of the previous pixel (n-1), the resultant is subtracted from the energy of pixel (n). The correction factor Cf1 and Cf1 in percentage can be calculated as (Figure 96):

$$Cf_1[\%] = \frac{R_1}{81.92} \quad Cf_2[\%] = \frac{R_2}{81.92}$$

**Figure 96: Relation of the correction factor coefficients and correction percentages**

#### **EPP\_CMD**

Event preprocessing pipeline command: This register commands the main FSM of the FP. Different FP operating modes can be accessed through this register according to Table 13.

| Value | Command                         |
|-------|---------------------------------|
| FF00h | Soft-reset command              |
| FFF0h | Standby mode                    |
| FF80h | Program mode                    |
| FF8Ah | Offset LUT Generator            |
| FF8Bh | Noise LUT Generator             |
| FFFAh | Diagnostic Offset LUT Generator |
| FFFBh | Diagnostic Noise LUT Generator  |
| FFF5h | Start LUT Generation            |
| FFFC  | Diagnostic mode                 |
| FFFFh | Work mode                       |

**Table 13: EPP command list.**

Command descriptions:

**Soft reset command:** sets the FP to reset state for a few seconds leaving it on the initial state. It is recommended to use this command between observations.

**Standby mode command:** the standby mode command, as its name describes, sets the FP into standby mode. Standby mode is an idle state for the FP where it waits for instructions. This mode is a transition state, for security reasons, **all commanding operations are only executed if the FP is in this state**, except for the *Start LUT Generation* command.

**Program mode command:** access to most of the FP registers is only allowed while the FP is in programming mode. This avoids address access conflicts during FP operation. By sending this command when in Standby mode the FP will enter the programming state.

**Offset LUT generator command:** sending this command while in standby mode will set the FP ready to start the offset LUT generation.

**Noise LUT generator command:** sending this command while in standby mode will set the FP ready to start the noise LUT generation.

**Diagnostic offset LUT generation command:** sending this command while in standby mode will set the FP ready to start the offset LUT generation. In this mode the external pixel input will not be used but the diagnostic internal memories will be used instead.

**Diagnostic noise LUT generation command:** sending this command while in standby mode will set the FP ready to start the noise LUT generation. In this mode the external pixel input will not be used but the diagnostic internal memories will be used instead.

**Start LUT generation command:** sending this command while the FP is on any of the LUT generator modes, will start the LUT generation procedure.

**Diagnostic mode command:** sending this command while the FP is on standby mode will start the FP pipeline preprocessing operation in diagnostic mode. When in diagnostic mode the input to the pipeline is set from the internal diagnostic memories instead of a external input.

**Work mode command:** Sending this command while the FP is on standby mode will start the normal FP pipeline preprocessing operation. Pixel input is set from the FPGA's pin inputs synchronized with the pixel and frame clocks.

## EPP\_PFU\_BYPASS

Event preprocessing pipeline pixel filtering unit bypass: This register is used to bypass user-selected flags on the PFU. If a flag is set to bypass, the flag will no longer be considered by the PFU to determine the validity of a pixel. Therefore, all pixels will be considered valid regardless of that specific flag value. The internal structure of the register is given in the PFU subsection.

| Communication and debug registers |                 |             |               |
|-----------------------------------|-----------------|-------------|---------------|
| Register                          | Address (range) | Access type | Default value |
| MSD_TAB_DELAY                     | 7300h           | R/W         | 0000h         |
| MSD_SPW_DIV                       | 7301h           | R/W         | 0000h         |
| MSD_SPW_PKG_SIZE                  | 7302h           | R/W         | 0000h         |
| MSD_ETH_PWR                       | 7400h           | R/W         | 0000h         |
| MSD_ETH_BURST_SIZE                | 74001h          | R/W         | 0008h         |

**Table 14: Register list of the command interface communication and debug registers.**

## MSD\_TAB\_DELAY

**Debug tab delay:** This register sets the system clock cycles delay of the output of the pixel clock and frame clock from their output to their input to the FP. Calibration of this register is dependent on the offset between the input pixel values and the trigger clocking signal. The default value is set to zero delay. Modifying this value impacts the overall pipeline timing; bigger values lead to bigger delays. Values can range from 0 to 251.

### **MSD\_SPW\_DIV**

**Debug SpaceWire clock divisor:** This register represents a scaling factor minus 1, is used to scale the transmit base clock into the transmission bit rate of the SpW interface. The system clock is divided by (unsigned (MSD\_SPW\_DIV) + 1). Changing this signal will immediately change the transmission rate. During link setup, the transmission rate is always 10 Mbit/s.

### **MSD\_SPW\_PKG\_SIZE**

**Debug SpW package size:** This register connects to the IFC science data control interface to set the number of events sent on each SpaceWire packet. By modifying this, the user can set the number of pixel events to be sent on each SpaceWire packet register (up to 256 events, 6 bytes each). The number of events per packet is equal to the register value plus one; e.g. default value 0x0000 is equal to one event per packet. Increasing this value can have a positive effect on the receiving SpW equipment by decreasing the overhead data, thus increasing the net FP data output capabilities. However, big values can have a negative effect when big data bursts are transmitted.

### **MSD\_ETH\_PWR**

**Debug Ethernet Power:** This register activates/deactivates the High-Speed Serial Interface (HSSI) used to acquire raw data coming directly from the FPM ADCs. The activation switch is rising edge enabled, meaning that activation depends on the transition of the LSB from '0' to '1'. Once the interface and FP have been activated, the interface will output the requested number of frames according to the MSD\_ETH\_BURST\_SIZE register and stop automatically afterwards.

### **MSD\_ETH\_BURST\_SIZE**

**Debug Ethernet Burst Size** This register determines the number of frames to be output by the FP-HSSI once the interface has been activated. The register accepts values between 0001h and FFFEh, where each value corresponds to a burst size. Additionally, when the register is set to FFFFh, the interface will output data continuously until the HSSI is deactivated via the MSD\_ETH\_PWR register.

| Sequencer registers     |                 |             |               |
|-------------------------|-----------------|-------------|---------------|
| Register                | Address (range) | Access type | Default value |
| SEQ_VER_SPI_MEM         | 3000h-3BFFh     | R/W         | NA            |
| SEQ_INIT_LEN            | 3C00h           | R/W         | 003Fh         |
| SEQ_INIT_PATCH_LEN      | 3C01h           | R/W         | 000Fh         |
| SEQ_LINE_LEN            | 3C02h           | R/W         | 00C7h         |
| SEQ_LINE_PATCH_LEN      | 3C03h           | R/W         | 0002h         |
| SEQ_VER_SPI_CMD         | 3C04h           | R/W         | NA            |
| SEQ_VER_SPI_CLK_PER     | 3C05h           | R/W         | 0004h         |
| SEQ_VER_ERROR           | 3C06h           | R/W         | NA            |
| SEQ_VER_ASIC_CHECK_SIZE | 3C10h-3C11h     | R/W         | 000Fh         |
| SEQ_VER_INIT_SIZE       | 3C18h           | R/W         | 003Fh         |
| SEQ_VER_DATA_RD         | 3C19h           | R           | NA            |
| SEQ_CMD                 | 3CF2h           | R/W         | NA            |
| SEQ_FRAME_SIZE          | 3CF4h           | R/W         | 0001h         |
| SEQ_LAST_LEN            | 3CF5h           | R/W         | 003Fh         |
| SEQ_FRAME_MEM           | 3D00h-3BFFh     | R/W         | NA            |
| SEQ_PATCH_MEM_A         | 4000h-47FFh     | R/W         | NA            |
| SEQ_PATCH_MEM_B         | 4800h-4FFFh     | R/W         | NA            |
| SEQ_LINE_MEM            | 5000h-6FFFh     | R/W         | NA            |

**Table 15: Register list of the command interface sequencer registers.**

## SEQ\_VER\_SPI\_MEMORY

Veritas SPI memory of the sequencer unit: These registers are used to write directly into the Veritas memory inside the sequencer unit of the FP. The memory is composed of 1024 words of 48 bits. To adapt to the 16-bit interface, each word is composed of three consecutive addresses on the command interface, using a total of 3072 addresses. The address range of this command covers from 3000h to 3BFF. The 1024 addresses of the Veritas memory are subdivided into two main sections: the ASIC cyclic check section and the memory initialization/programming section. The first section is subdivided again into 8 sections corresponding to each of the 8 Veritas ASICs that can be connected to the sequencer's SPI interface, which in the case of the FD only uses 2 ASICs. The contents of these memories are meant to store commands that are used by the Veritas controller to sequentially check the status of the Veritas memories during operation mode. The second section of the memory is used to store command words to be used for Veritas initialization or individual register programming when the sequencer is in programming mode. The internal structure of the Veritas memory is given in the table below:

| Memory section                | Address range |
|-------------------------------|---------------|
| Cyclic check ASIC A           | 3000h-30EFh   |
| Cyclic check ASIC B           | 30F0h-31DFh   |
| Initialization/program memory | 3780h-3BFFh   |

**Table 16: Internal structure of the SPI interface memory.**

### **SEQ\_INIT\_LEN**

Sequencer Initialization line length: This register sets the length in clock cycles of the initialization line of the sequence minus one. This is the very first stream of values output when the sequencer is placed in work mode. The user should give a value in the range of 0000h-07FFh.

### **SEQ\_INIT\_PATCH\_LEN**

Sequencer initialization line patch length: This register sets the size in clock cycles of the patches used to generate the initialization line of the sequencer. This is the first line output when the sequencer is set to work mode. Its range can be set from 0000h to 000Fh. The counter of the patch size starts at zero; therefore, a value of 0000h implies 1 clock cycle.

### **SEQ\_LINE\_LEN**

Sequencer standard line length: This register sets the length in clock cycles of a standard line sequence minus one. The user should give a value in the range of 0000h-07FFh.

### **SEQ\_LINE\_PATCH\_LEN**

Sequencer initialization line patch length: This register sets the size in clock cycles of the patches used to generate the standard line sequences. Its range can be set from 0000h to 000Fh.

### **SEQ\_VER\_SPI\_CMD**

Sequencer veritas command: This register is used to command the Veritas controller to execute a command. The data field of this command contains the address of the Veritas memory word that the user wants to execute. The value of this command can range from address 3280h to 33FFh. However, address 33FFh will not execute word 3FFh but will execute consecutively words 280h to word XXXh. The address of the stopping word is programmed using the SEQ\_VER\_INIT\_SIZE command. This function is meant for initialization purposes to avoid the user from writing each word one by one.

### **SEQ\_VER\_SPI\_CLK\_PER**

Sequencer SPI clock period: This register allows the user to set the SPI clock period by programming the number of system clock cycles to take for each SPI clock. The register accepts a range from 0002h to 00FFh; the default is 0004h. If the system clock is 80MHz, the default SPI frequency is 20MHz.

### **SEQ\_VER\_ERROR**

Sequencer Veritas error register: This register is automatically updated by the Veritas control unit during operation to alert the user of any memory mismatch between the Veritas memory and the Veritas ASICs registers. The register is initialized to 0000h, and each of the first 2 bits corresponds to each of the 2 Veritas ASICs. If a mismatch is found in an ASIC, the corresponding bit is set to '1', indicating an error has occurred. The user can overwrite this register once the error has been cleared.

### **SEQ\_VER\_ASIC\_CHECK\_SIZE**

Sequencer Veritas ASIC check size: This register allows the user to set the size of the cyclic check Veritas memory region for each ASIC. In other words, it allows the user to define how many registers must be read/written for each ASIC during operation. The accepted range for the register data is 0000h to 004Fh. The correspondence of each address with the respective ASIC is given in Table 17.

| ASIC      | Address |
|-----------|---------|
| Veritas A | 3C10h   |
| Veritas B | 3C11h   |

**Table 17: Veritas ASIC check size register addresses.**

### **SEQ\_VER\_INIT\_SIZE**

Sequencer veritas initialization memory size: This register sets the size in words of the initialization block when the initialization command is sent to the Veritas controller. It accepts a data range from 0000h to 017Fh.

### **SEQ\_VER\_DATA\_RD**

Sequencer veritas SPI data read: This register allows the user to read back the data received during Veritas SPI operations. The incoming data is stored in the register and remains available until new command data is received.

### **SEQ\_CMD**

Sequencer command: This register sets the main command register for the internal sequencer. Its modification can set the internal sequencer to different operating modes. The truth table for this register is shown below. Transitions to work are only

allowed when the sequencer is in standby mode. Therefore, if the sequencer is in program mode, the user should first send the command to standby mode and then follow with a work mode command.

| Value | Operating mode |
|-------|----------------|
| FFF0h | Standby        |
| FF80h | Program        |
| FFFFh | Work           |

**Table 18: List of sequencer commands.**

### **SEQ\_FRAME\_SIZE**

Sequencer frame size register: This register sets the size of the sequence driving the FEE in terms of the amount of addresses needed by the frame memory. It should be used in conjunction with the EPP\_FRAME\_SIZE register and the sequencer software tools to set the appropriate length for the proper window mode. The accepted range of values goes from 0 to 511 lines, in other words, 0000h-01FFh.

### **SEQ\_LAST\_LEN**

Sequencer last line length: This register sets the length in clock cycles of the last line of the sequence minus one. This is the very last stream of values output when the sequencer reaches the end of a frame. Given that the last line can be shorter, this register allows for an additional degree of flexibility. The user should give a value in the range of 0000h-07FFh.

### **SEQ\_FRAME\_MEM**

Sequencer frame memory: The sequencer frame memory is a range of registers which point to the different line sequences stored in the sequencer line memory. The sequencer reads on each frame the frame memory addresses in consecutive order until it reaches the last address (configured with the SEQ\_FRAME\_SIZE register), each address points at which line is to be output. Memory values can be accessed in the range of addresses 3D00h-3BFFh.

### **SEQ\_PATCH\_MEM\_A / SEQ\_PATCH\_MEM\_B**

Sequencer patch memory map memory page A/ Sequencer patch memory map memory page B: The sequencer memory map memory is a range of registers which store patch sequences for the FP sequencer operation. Each basic patch sequence contains up to 16 sequence-clock-cycles which are read-out on consecutive system clock cycles. The order in which the patch sequences are output on each line is set by the line memory contents. Given that the sequencer has a width of 40 bits, the complete patch



memory is subdivided into 2 pages 'A' and 'B', page 'A' covers bits 0 to 31 of the sequence bits, while section 'B' covers bits 32 to 39. Given that the input structure is only 16 bits wide, each word on page 'A' corresponds to two consecutive addresses in the commanding interface. Page 'B' mirrors the access structure, but every second address is empty. Memory values on page 'A' cover the address range 4000h-47FFh. While memory values on page 'B' cover the address range 4800h-4FFFh.

### **SEQ\_LINE\_MEM**

Sequencer line memory: The sequencer line memory is a range of registers which point to the different patch sequences stored in the sequencer patch memory. The sequencer reads on each frame the line memory map address to generate each line-sequence (64 pixel-clocks). The order in which the line-sequence memory is read on each frame is given by the sequencer line output FSM. Memory values can be accessed in the range of addresses 5000h-6FFFh.

## APPENDIX B: LD INTERFACE CONTROLLER OPERATION

Data transfers to the FP module are carried via the IFC's SpW interface. Each SpW packet is generally composed of 7 bytes with the following structure, where the meaning of each byte is displayed in Table 19

0xXX 0XXXXX 0XXXXX 0XXXXX EOP

| Bytes | Value  | Interface         |
|-------|--------|-------------------|
| 1     | 0xXX   | Interface address |
| 2-3   | 0XXXXX | Address           |
| 4-5   | 0XXXXX | Send/Receive      |
| 6-7   | 0XXXXX | CMD/Data          |

**Table 19: 7-byte command packet structure contents.**

The first byte of the package corresponds to the interface address. This character allows the IFC to direct the received information to its respective interface. Possible values are given in Table 20.

| Value | Interface              |
|-------|------------------------|
| 0x33  | Housekeeping Interface |
| 0x34  | Command Interface      |
| 0x35  | Science Interface      |
| 0x20  | Echo Interface         |

**Table 20: Interface address map.**

The second and third bytes correspond to the address of the register to be written. A detailed description of this field is given in the following sections.

The fourth and fifth bytes set the command direction. In the case of send, the information contained in the CMD-data bytes (6-7) is written in the address indicated by the address bytes (2-3). When the value corresponds to receive, the CMD-data bytes are

ignored and instead the interface will reply to the command interface with the value stored in the register indicated by the address field.

| Value  | Action       |
|--------|--------------|
| 0x4182 | Send data    |
| 0xC003 | Receive data |

**Table 21: Send/Receive possible values.**

The remaining bytes (6-7) contain the information to be written on the registers in case of the send data command. In receive data commands, these bytes are ignored.

## HOUSEKEEPING INTERFACE

The housekeeping interface controls data transfers to and from the HKU, described in the previous section. The HKU uses a RAM memory as an asynchronous interface for all the read and write operations, and therefore all transactions are made with the same memory range. Therefore, this interface's transactions are handled as single multiple-word registers.

| Register   | Address (range) | Access type | Default value |
|------------|-----------------|-------------|---------------|
| HKU_MEMORY | 2000h -203Fh    | R/W         | NA            |

**Table 22: Register list of the housekeeping interface.**

## COMMAND INTERFACE

The command interface contains most of the registers necessary to operate the FP. The registers can be divided into three sub-groups: the EPP registers, those used for the EPP operation; communications and debug registers, used to handle the FP communication interfaces; and sequencer registers, controlling the SEQ unit. A list of each sub-group's registers is given in this subsection together with a short register description.

### EPP Registers

| Register                   | Address (range) | Access type | Default value |
|----------------------------|-----------------|-------------|---------------|
| <b>EPP_UPPER_THRHLD</b>    | 7000h           | R/W         | 36B0h         |
| <b>EPP_LOWER_THRHLD</b>    | 7001h           | R/W         | 000Ch         |
| <b>EPP_PFU_MASK</b>        | 7002h           | R/W         | FFFFh         |
| <b>EPP_MODE</b>            | 7005h           | R           | NA            |
| <b>EPP_DIAG_RATIO</b>      | 7007h           | R/W         | 0800h         |
| <b>EPP_SELECT</b>          | 7009h           | R/W         | 0000h         |
| <b>EPP_PFU_SWITCH</b>      | 7020h           | R/W         | FFFFh         |
| <b>EPP_MLT</b>             | 7021h           | R/W         | 0000h         |
| <b>EPP_MHT</b>             | 7022h           | R/W         | FFFFh         |
| <b>EPP_LTP_SIGMA1</b>      | 7024h           | R/W         | 0005h         |
| <b>EPP_LTP_SIGMA2</b>      | 7025h           | R/W         | 0003h         |
| <b>EPP_PCU_OFFSET_ADDR</b> | 7026h           | R/W         | 0000h         |
| <b>EPP_EFU_OFFSET_ADDR</b> | 7027h           | R/W         | 0208h         |
| <b>EPP_SLICE_SELECT</b>    | 7030h           | R/W         | 00FFh         |
| <b>EPP_STARTLINE</b>       | 7032h           | R/W         | 0000h         |
| <b>EPP_STOPPLINE</b>       | 7034h           | R/W         | 01FFh         |
| <b>EPP_CMD</b>             | 7100h           | R/W         | NA            |
| <b>EPP_PFU_BYPASS</b>      | 7200h           | R/W         | 0000h         |

**Table 23: Registers list of the command interface LD EPP registers.**

### **EPP\_UPPER\_THRHLD/ EPP\_LOWER\_THRHLD**

Event preprocessing pipeline upper threshold/ Event preprocessing pipeline lower threshold: The upper threshold register holds the threshold energy above which an event is considered a MIP. This value is used by the PCU unit to set the UT flag to 0 in case the pixel energy is above this energy threshold value. The lower threshold register holds the threshold energy below which an event is considered a MISFIT. This value is used by the PCU unit to set the LT flag to 0 in case the pixel energy is below this energy threshold value.

### **EPP\_PFU\_MASK**

This register contains a mask used by the PFU to decide which pixels are marked as valid. Each bit on the mask corresponds to a pixel flag. If the pixel flags and the mask flags are all equal, then the pixel is marked as valid. Internal register structure can be seen in the PFU sub-section.

## EPP\_MODE

Event preprocessing pipeline operation mode: This read-only register allows us to request the status of the main operation register of the FP.

## EPP\_DIAG\_RATIO

Event preprocessing pipeline diagnostic good-frame/bad-frame ratio: This register sets the ratio between good and bad frames to be entered to the pipeline during diagnostic mode operation. The XXXXh value stored in the register can be read as follows: XXXXh is the number of “bad-frames” entered to the pipeline per one “good-frame”.

## EPP\_SELECT

Event preprocessing pipeline memory select. This register is used to point the different memories inside the FP, each value corresponds to a different memory, as shown on the table below.

| Value  | Selected Memory           |
|--------|---------------------------|
| 0x0020 | SSRAM FIFO 3              |
| 0x0021 | SSRAM FIFO 2              |
| 0x003C | Good-frame diagnostic RAM |
| 0X003D | Bad-frame diagnostic RAM  |

**Table 24: Memory selection address map.**

Once the desired memory has been selected on register 7009h, its contents are accessible in addresses 8000h-FFFFh, depending on the selected memory.

| Memory                    | Address range |
|---------------------------|---------------|
| SSRAM FIFO 3              | 8000h-8007h   |
| SSRAM FIFO 2              | 8000h-8007h   |
| Good-frame diagnostic RAM | 8000h-8FFFh   |
| Bad-frame diagnostic RAM  | 8000h-8FFFh   |

**Table 25: Address range available to each memory.**

SRAM FIFOs 2 & 3 serve as interfaces to the SSRAM memory controller. SSRAM FIFO 2 values are read-only, while FIFO 3 values are write-only. The eight 16-bit registers of SSRAM FIFO 3 memory serve to communicate with the 121-bit interface of the SSRAM memory controller. Write operations take place only when address 8007h is written. At write command execution values from 8000h-8007h are written in parallel to the FIFO 3 interface. Register correspondence is shown in the following table:

| Address   | 8007h   |     |          | 8006h  | 8005h | 8004h | 8003h | 8002h | 8001     | 8000h    |
|-----------|---------|-----|----------|--------|-------|-------|-------|-------|----------|----------|
| Bit range | 127-112 |     |          | 111-96 | 95-80 | 79-64 | 63-48 | 47-32 | 31-16    | 15-0     |
| Sub-range | NA      | 8-5 | 4-0      | 15-0   | 15-0  | 15-0  | 15-0  | 15-0  | 15-0     | 15-0     |
| FIFO 3    | NA      | CMD | Sub-Addr |        | Th2_1 | TH2_0 | TH1_1 | TH1_0 | Offset_1 | Offset_0 |

**Table 26: SRAM FIFO 3 registers correspondence to SSRAM interface.**

Write operation takes place when the address register switches to address 8007h, therefore two consecutive write operations to address 8007h will write to FIFO 3 only the first time. Is expected that the user will update at least one of the other registers 8000h-8006h before the next write operation.

### **EPP\_PFU\_SWITCH**

Event preprocessing pipeline pixel filtering unit switch: This register contains information on which stages of the pipeline are switched ON/OFF, depending on the respective bit on the switch; different stages will stop flagging pixels or, in the case of the raw data bit, the complete pipeline can be set OFF. Internal register is shown in the PFU sub-section.

### **EPP\_MLT / EPP\_MHT**

Event preprocessing pipeline bad-median lower threshold / Event preprocessing pipeline bad-median higher threshold: The lower threshold register stores the threshold value XXXXh below which, if the median found by the CMU is lower, all pixels of the line are marked with BM flag. The higher threshold register stores the threshold value XXXXh above which, if the median found by the CMU is higher, all pixels of the line are marked with BM flag.

### **EPP\_LTP\_SIGMA1 / EPP\_LTP\_SIGMA2**

Event preprocessing pipeline look-up-table programmer sigma 1 / Event preprocessing pipeline look-up-table programmer sigma 2: These registers store the factor by which the LTP multiplies the noise (variance) of each pixel to give the first and

second threshold values used on the EFU. Therefore,  $1T = \text{XXXXh} * \text{sigma}$  and  $2T = \text{XXXXh} * \text{sigma}$ .

### EPP\_PCU\_OFFSET\_ADDR

Event preprocessing pipeline pixel correction unit offset address. This register considers the base address offset to which the EFU threshold values should be written in the SSRAM. This register is calibrated to the input of the pipeline and therefore should not be modified unless an external delay to the pipeline input exists.

### EPP\_EFU\_OFFSET\_ADDR

Event preprocessing pipeline event filtering unit offset address. This register considers the base address offset to which the PCU values should be written in the SSRAM. This register is calibrated to the input of the pipeline and therefore should not be modified unless an external delay to the pipeline input exists.

### EPP\_SLICE\_SELECT

Event preprocessing pipeline slice selection. This register allows to select which slices of the pipeline are active during operation. Pixels coming from non-active slices are marked as invalid and therefore are not transmitted. The truth table for this register is shown below:

| BIT        | 15-8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------------|------|---|---|---|---|---|---|---|---|
| Slice 0 ON | X    | X | X | X | X | X | X | X | 1 |
| Slice 1 ON | X    | X | X | X | X | X | X | 1 | X |
| Slice 2 ON | X    | X | X | X | X | X | 1 | X | X |
| Slice 3 ON | X    | X | X | X | X | 1 | X | X | X |
| Slice 4 ON | X    | X | X | X | 1 | X | X | X | X |
| Slice 5 ON | X    | X | X | 1 | X | X | X | X | X |
| Slice 6 ON | X    | X | 1 | X | X | X | X | X | X |
| Slice 7 ON | X    | 1 | X | X | X | X | X | X | X |

**Table 27: Slice select register truth table.**

Note that each slice is activated only depending on its corresponding bit, this allows the flexibility to activate any combination of slices as needed.

### EPP\_STARLINE/EPP\_STOPLINE

Event preprocessing pipeline starting and stopping line. These register sets the starting and stopping line for the FP operation; the default value is to start on line 0 while for the stop line is 511. By using these registers together the behavior of the FP can be modified for window mode operation. The starting and stopping lines can be set to any value according to the desired window.

## EPP\_CMD

Event preprocessing pipeline command: This register commands the main FSM of the FP. Different FP operating modes can be accessed through this register according to Table 28.

| Value | Command                         |
|-------|---------------------------------|
| FF00h | Soft-reset command              |
| FFF0h | Standby mode                    |
| FF80h | Program mode                    |
| FF8Ah | Offset LUT Generator            |
| FF8Bh | Noise LUT Generator             |
| FFFAh | Diagnostic Offset LUT Generator |
| FFFBh | Diagnostic Noise LUT Generator  |
| FFF5h | Start LUT Generation            |
| FFFC  | Diagnostic mode                 |
| FFFFh | Work mode                       |

**Table 28: EPP command list.**

Command descriptions:

**Soft-reset command:** sets the FP to reset state for a few seconds leaving it on the initial state. It's recommended to use this command between observations. The soft-reset network is not connected to the SSRAM controller, this unit has its own reset command accessible through the FIFO 3 interface.

**Standby mode command:** the standby mode command, as its name describes, sets the FP into standby mode. Standby mode is an idle state for the FP where it waits for instructions. This mode is a transition state, for security reasons, all commanding operations are only executed if the FP is in this state, except for the *Start LUT Generation* command.



**Program mode command:** access to most of the FP registers is only allowed while the FP is in programming mode. This avoids address access conflicts during FP operation. By sending this command when in Standby mode the FP will enter the programming state.

**Offset LUT generator command:** sending this command while in standby mode will set the FP ready to start the offset LUT generation.

**Noise LUT generator command:** sending this command while in standby mode will set the FP ready to start the noise LUT generation.

**Diagnostic offset LUT generation command:** sending this command while in standby mode will set the FP ready to start the offset LUT generation. In this mode the external pixel input will not be used but the diagnostic internal memories will be used instead.

**Diagnostic noise LUT generation command:** sending this command while in standby mode will set the FP ready to start the noise LUT generation. In this mode the external pixel input will not be used but the diagnostic internal memories will be used instead.

**Start LUT generation command:** sending this command while the FP is on any of the LUT generator modes, will start the LUT generation procedure.

**Diagnostic mode command:** sending this command while the FP is on standby mode will start the FP pipeline preprocessing operation in diagnostic mode. When in diagnostic mode the input to the pipeline is set from the internal diagnostic memories instead of an external input.

**Work mode command:** sending this command while the FP is on standby mode will start the normal FP pipeline preprocessing operation. Pixel input is set from the FPGA's pin inputs synchronized with the pixel and frame clocks.

| Communication and debug registers |                 |             |               |
|-----------------------------------|-----------------|-------------|---------------|
| Register                          | Address (range) | Access type | Default value |
| MSD_TAB_DELAY                     | 7300h           | R/W         | 0000h         |
| MSD_SPW_DIV                       | 7301h           | R/W         | 0000h         |
| MSD_SPW_PKG_SIZE                  | 7302h           | R/W         | 0000h         |
| MSD_ETH_PWR                       | 7400h           | R/W         | 0000h         |
| MSD_ETH_BURST_SIZE                | 7401h           | R/W         | 0008h         |
| MSD_ETH_SELECT_SLICE              | 7402h           | R/W         | 0000h         |

**Table 29: FP debug registers**

## **MSD\_TAB\_DELAY**

Debug tab delay: This register sets the system clock cycles delay of the output of the pixel clock and frame clock from their output to their input to the FP. Calibration of this register is dependent on the offset between the input pixel values and the trigger clocking signal. The default value is set to zero delay. Modifying this value impacts the overall pipeline timing; bigger values lead to bigger delays. Values can range from 0 to 251.

## **MSD\_SPW\_DIV**

Debug SpaceWire clock divisor: This register represents a scaling factor minus 1, is used to scale the transmit base clock into the transmission bit rate of the SpW interface. The system clock is divided by (unsigned (MSD\_SPW\_DIV) + 1). Changing this signal will immediately change the transmission rate. During link setup, the transmission rate is always 10 Mbit/s.

## **MSD\_SPW\_PKG\_SIZE**

Debug SpW package size: This register connects to the IFC science data control interface to set the number of events sent on each SpaceWire packet. By modifying this, the user can set the number of pixel events to be sent on each SpaceWire packet register (up to 256 events, 6 bytes each). The number of events per packet is equal to the register value plus one; e.g. default value 0x0000 is equal to one event per packet. Increasing this value can have a positive effect on the receiving SpW equipment by decreasing the overhead data, thus increasing the net FP data output capabilities. However, big values can have a negative effect when big data bursts are transmitted.

## **MSD\_ETH\_PWR**

Debug Ethernet Power: This register activates/deactivates the High-Speed Serial Interface (HSSI) used to acquire raw data coming directly from the FPM ADCs. The activation switch is rising edge enabled, meaning that activation depends on the transition of the LSB from '0' to '1'. Once the interface and FP have been activated, the interface will output the requested number of frames according to the MSD\_ETH\_BURST\_SIZE register and stop automatically afterwards.

## **MSD\_ETH\_BURST\_SIZE**

Debug Ethernet Burst Size This register determines the number of frames to be output by the FP-HSSI once the interface has been activated. The register accepts values between 0001h and FFFEh, where each value corresponds to a burst size. Additionally, when the register is set to FFFFh, the interface will output data continuously until the HSSI is deactivated via the MSD\_ETH\_PWR register.

## MSD\_ETH\_SELECT\_SLICE

Debug HSSI slice select register. The HSSI can only transmit data coming from one of the eight 64x512 pixel slides of a quadrant of the LDA. To allow users to debug the different slices of a LDA quadrant the HSSI can be re-connected using the slice select register. The corresponding values for slice selection are one-hot encoded on the least significant byte of the register, Table 30 depicts the corresponding slice selection values, please note that other values are not allowed and will return a zero output.

| Value | Selected slice |
|-------|----------------|
| 0001h | Slice 1        |
| 0002h | Slice 2        |
| 0004h | Slice 3        |
| 0008h | Slice 4        |
| 0010h | Slice 5        |
| 0020h | Slice 6        |
| 0040h | Slice 7        |
| 0080h | Slice 8        |

**Table 30: Slice select register values decoding.**

| Sequencer registers     |                 |             |               |
|-------------------------|-----------------|-------------|---------------|
| Register                | Address (range) | Access type | Default value |
| SEQ_VER_SPI_MEM         | 3000h-3BFFh     | R/W         | NA            |
| SEQ_INIT_LEN            | 3C00h           | R/W         | 003Fh         |
| SEQ_INIT_PATCH_LEN      | 3C01h           | R/W         | 000Fh         |
| SEQ_LINE_LEN            | 3C02h           | R/W         | 00C7h         |
| SEQ_LINE_PATCH_LEN      | 3C03h           | R/W         | 0002h         |
| SEQ_VER_SPI_CMD         | 3C04h           | R/W         | NA            |
| SEQ_VER_SPI_CLK_PER     | 3C05h           | R/W         | 0004h         |
| SEQ_VER_ERROR           | 3C06h           | R/W         | NA            |
| SEQ_VER_ASIC_CHECK_SIZE | 3C10h-3C17h     | R/W         | 000Fh         |
| SEQ_VER_INIT_SIZE       | 3C18h           | R/W         | 003Fh         |
| SEQ_VER_DATA_RD         | 3C19h           | R           | NA            |
| SEQ_CMD                 | 3CF2h           | R/W         | NA            |
| SEQ_FRAME_SIZE          | 3CF4h           | R/W         | 0001h         |
| SEQ_LAST_LEN            | 3CF5h           | R/W         | 003Fh         |
| SEQ_FRAME_MEM           | 3D00h-3BFFh     | R/W         | NA            |
| SEQ_PATCH_MEM_A         | 4000h-47FFh     | R/W         | NA            |

|                        |             |     |    |
|------------------------|-------------|-----|----|
| <b>SEQ_PATCH_MEM_B</b> | 4800h-4FFFh | R/W | NA |
| <b>SEQ_LINE_MEM</b>    | 5000h-6FFFh | R/W | NA |

**Table 31: Register list of the command interface sequencer registers.**

## SEQ\_VER\_SPI\_MEMORY

Veritas SPI memory of the sequencer unit: These registers are used to write directly into the Veritas memory inside the sequencer unit of the FP. The memory is composed of 1024 words of 48 bits. To adapt to the 16-bit interface, each word is composed of three consecutive addresses on the command interface, using a total of 3072 addresses. The address range of this command covers from 3000h to 3BFF. The 1024 addresses of the Veritas memory are subdivided into two main sections: the ASIC cyclic check section and the memory initialization/programming section. The first section is subdivided again into 8 sections corresponding to each of the 8 Veritas ASICs that can be connected to the sequencer's SPI interface. The contents of these memories are meant to store commands that are used by the Veritas controller to sequentially check the status of the Veritas memories during operation mode. The second section of the memory is used to store command words to be used for Veritas initialization or individual register programming when the sequencer is in programming mode. The internal structure of the Veritas memory is given in the table below:

| Memory section                       | Address range |
|--------------------------------------|---------------|
| <b>Cyclic check ASIC A</b>           | 3000h-30EFh   |
| <b>Cyclic check ASIC B</b>           | 30F0h-31DFh   |
| <b>Cyclic check ASIC C</b>           | 31E0h-32CFh   |
| <b>Cyclic check ASIC D</b>           | 32D0h-33BFh   |
| <b>Cyclic check ASIC E</b>           | 33C0h-34AFh   |
| <b>Cyclic check ASIC F</b>           | 34B0h-359Fh   |
| <b>Cyclic check ASIC G</b>           | 35A0h-368Fh   |
| <b>Cyclic check ASIC H</b>           | 3690h-377Fh   |
| <b>Initialization/program memory</b> | 3780h-3BFFh   |

**Table 32: Mapping of command addresses to the Veritas memory of the FP.**

## SEQ\_INIT\_LEN

Sequencer Initialization line length: This register sets the length in clock cycles of the initialization line of the sequence minus one. This is the very first stream of values output when the sequencer is placed in work mode. The user should give a value in the range of 0000h-07FFh.

### **SEQ\_INIT\_PATCH\_LEN**

Sequencer initialization line patch length: This register sets the size in clock cycles of the patches used to generate the initialization line of the sequencer. This is the first line output when the sequencer is set to work mode. Its range can be set from 0000h to 000Fh. The counter of the patch size starts at zero; therefore, a value of 0000h implies 1 clock cycle.

### **SEQ\_LINE\_LEN**

Sequencer standard line length: This register sets the length in clock cycles of a standard line sequence minus one. The user should give a value in the range of 0000h-07FFh.

### **SEQ\_LINE\_PATCH\_LEN**

Sequencer initialization line patch length: This register sets the size in clock cycles of the patches used to generate the standard line sequences. Its range can be set from 0000h to 000Fh.

### **SEQ\_VER\_SPI\_CMD**

Sequencer veritas command: This register is used to command the Veritas controller to execute a command. The data field of this command contains the address of the Veritas memory word that the user wants to execute. The value of this command can range from address 3280h to 33FFh. However, address 33FFh will not execute word 3FFh but will execute consecutively words 280h to word XXXh. The address of the stopping word is programmed using the SEQ\_VER\_INIT\_SIZE command. This function is meant for initialization purposes to avoid the user from writing each word one by one.

### **SEQ\_VER\_SPI\_CLK\_PER**

Sequencer SPI clock period: This register allows the user to set the SPI clock period by programming the number of system clock cycles to take for each SPI clock. The register accepts a range from 0002h to 00FFh; the default is 0004h. If the system clock is 80MHz, the default SPI frequency is 20MHz.

### **SEQ\_VER\_ERROR**

Sequencer Veritas error register: This register is automatically updated by the Veritas control unit during operation to alert the user of any memory mismatch between

the Veritas memory and the Veritas ASICs registers. The register is initialized to 0000h, and each of the first 2 bits corresponds to each of the 2 Veritas ASICs. If a mismatch is found in an ASIC, the corresponding bit is set to '1', indicating an error has occurred. The user can overwrite this register once the error has been cleared.

### **SEQ\_VER\_ASIC\_CHECK\_SIZE**

Sequencer Veritas ASIC check size: This register allows the user to set the size of the cyclic check Veritas memory region for each ASIC. In other words, it allows the user to define how many registers must be read/written for each ASIC during operation. The accepted range for the register data is 0000h to 004Fh. The correspondence of each address with the respective ASIC is given in Table 33.

| ASIC      | Address |
|-----------|---------|
| Veritas A | 3C10h   |
| Veritas B | 3C11h   |
| Veritas C | 3C12h   |
| Veritas D | 3C13h   |
| Veritas E | 3C14h   |
| Veritas F | 3C15h   |
| Veritas G | 3C16h   |
| Veritas H | 3C17h   |

**Table 33: Veritas ASIC check size register addresses.**

### **SEQ\_VER\_INIT\_SIZE**

Sequencer veritas initialization memory size: This register sets the size in words of the initialization block when the initialization command is sent to the Veritas controller. It accepts a data range from 0000h to 017Fh.

### **SEQ\_VER\_DATA\_RD**

Sequencer veritas SPI data read: This register allows the user to read back the data received during Veritas SPI operations. The incoming data is stored in the register and remains available until new command data is received.

### **SEQ\_CMD**

Sequencer command: This register sets the main command register for the internal sequencer. Its modification can set the internal sequencer to different operating modes. The truth table for this register is shown below. Transitions to work are only allowed when the sequencer is in standby mode. Therefore, if the sequencer is in

program mode, the user should first send the command to standby mode and then follow with a work mode command.

| Value | Operating mode |
|-------|----------------|
| FFF0h | Standby        |
| FF80h | Program        |
| FFFFh | Work           |

**Table 34: List of sequencer commands.**

## **SEQ\_FRAME\_SIZE**

Sequencer frame size register: This register sets the size of the sequence driving the FEE in terms of the amount of addresses needed by the frame memory. It should be used in conjunction with the EPP\_FRAME\_SIZE register and the sequencer software tools to set the appropriate length for the proper window mode. The accepted range of values goes from 0 to 511 lines, in other words, 0000h-01FFh.

## **SEQ\_LAST\_LEN**

Sequencer last line length: This register sets the length in clock cycles of the last line of the sequence minus one. This is the very last stream of values output when the sequencer reaches the end of a frame. Given that the last line can be shorter, this register allows for an additional degree of flexibility. The user should give a value in the range of 0000h-07FFh.

## **SEQ\_FRAME\_MEM**

Sequencer frame memory: The sequencer frame memory is a range of registers which point to the different line sequences stored in the sequencer line memory. The sequencer reads on each frame the frame memory addresses in consecutive order until it reaches the last address (configured with the SEQ\_FRAME\_SIZE register), each address points at which line is to be output. Memory values can be accessed in the range of addresses 3D00h-3BFFh.

## **SEQ\_PATCH\_MEM\_A / SEQ\_PATCH\_MEM\_B**

Sequencer patch memory map memory page A/ Sequencer patch memory map memory page B: The sequencer memory map memory is a range of registers which store patch sequences for the FP sequencer operation. Each basic patch sequence contains up to 16 sequence-clock-cycles which are read-out on consecutive system clock cycles. The order in which the patch sequences are output on each line is set by the line memory contents. Given that the sequencer has a width of 40 bits, the complete patch memory is subdivided into 2 pages 'A' and 'B', page 'A' covers bits 0 to 31 of the

sequence bits, while section 'B' covers bits 32 to 39. Given that the input structure is only 16 bits wide, each word on page 'A' corresponds to two consecutive addresses in the commanding interface. Page 'B' mirrors the access structure, but every second address is empty. Memory values on page 'A' cover the address range 4000h-47FFh. While memory values on page 'B' cover the address range 4800h-4FFFh.

## **SEQ\_LINE\_MEM**

Sequencer line memory: The sequencer line memory is a range of registers which point to the different patch sequences stored in the sequencer patch memory. The sequencer reads on each frame the line memory map address to generate each line-sequence (64 pixel-clocks). The order in which the line-sequence memory is read on each frame is given by the sequencer line output FSM. Memory values can be accessed in the range of addresses 5000h-6FFFh.



## BIBLIOGRAPHY

- [1] A. Markowitz et al., "An RXTE Survey of Long-Term X-Ray Variability in Seyfert 1 Galaxies," *The Astrophysical Journal*, vol. 547, no. 2, pp. 684-692, 2001.
- [2] C. Bambi et al., *Handbook of X-ray and Gamma-ray Astrophysics*, Springer, 2022.
- [3] F. D. Seward and P. A. Charles, *Exploring the X-ray Universe*, Cambridge University Press, 2010.
- [4] K. Arnaud and R. Smith, *Handbook of X-ray Astronomy*, Cambridge University Press, 2011.
- [5] L. Angelini and J. Allen, "NASA HEASARC: Observatories," NASA, 24 September 2020. [Online]. Available: <https://heasarc.gsfc.nasa.gov/docs/einstein/heao2.html>. [Accessed 26 August 2025].
- [6] F. Jansen et al., "XMM-Newton observatory," *Astronomy and Astrophysics*, vol. 365, no. Special Issue 500/01, pp. L1-L6, 2001.
- [7] ESA, "XMM-Newton," ESA, [Online]. Available: <https://www.cosmos.esa.int/web/xmm-newton/home>. [Accessed 13 September 2025].
- [8] A. Brinkman et al., "The Reflection Grating Spectrometer on board XMM," in *Proceedings of the First XMM Workshop: Science with XMM*, Noordwijk, 1998.
- [9] M. J. L. Turner et al., "The European Photon Imaging Camera on XMM-Newton: The MOS Cameras," *Astronomy and Astrophysics*, vol. 365, p. L27, 2001.

- [10] L. Strüder et al., "The European Photon Imaging Camera on XMM-Newton: The pn-CCD Camera," *Astronomy and Astrophysics*, vol. 365, p. L18, 2001.
- [11] K. O. Mason et al., "The XMM-Newton optical/UV monitor telescope," *Astronomy and Astrophysics*, vol. 365, p. L36, 2001.
- [12] G. Branduardi-Raymont et al., "First observation of Jupiter by XMM-Newton," *Astronomy and Astrophysics*, vol. 424, p. 331, 2004.
- [13] A. Marino et al., "Constraints on the dense matter equation of state from young and cold isolated neutron stars," *Nature Astronomy*, vol. 8, pp. 1020-1030, 2024.
- [14] J. S. Sanders et al., "Measuring bulk flows of the intracluster medium in the Perseus and Coma galaxy clusters using XMM-Newton," *Astronomy and Astrophysics*, vol. 633, p. A42, 2020.
- [15] D. Barret et al., "ATHENA+" THE FIRST DEEP UNIVERSE X-RAY OBSERVATORY," *Société Française d'Astronomie et d'Astrophysique*, 2013.
- [16] K. Nandra et al., "The Hot and Energetic Universe: A White Paper presenting the science theme motivating the Athena+ mission", " 2013.
- [17] European Space Agency, "ESA Cosmos," ESA, [Online]. Available: <https://www.cosmos.esa.int/web/athena>. [Accessed 11 June 2026].
- [18] Athena Community Office, "Athena Community Office," IRAP/CNRS/UT3/CNES/Fab&Fab, 29 October 2019. [Online]. Available: <https://www.the-athena-x-ray-observatory.eu/es/node/100>. [Accessed 11 June 2026].
- [19] R. Willingale et al., "The Optical Design of the Athena+ Mirror," *Athena+ supporting paper*, 2014.

- [20] Athena Community Office, "Athena Community Office," 20 November 2018. [Online]. Available: <https://www.the-athena-x-ray-observatory.eu/es/node/103>. [Accessed 16 11 2025].
- [21] S. Bandler, "Athena Community Office," 01 February 2018. [Online]. Available: <https://www.the-athena-x-ray-observatory.eu/es/node/388>. [Accessed 16 November 2025].
- [22] D. Barret et al., "The Athena X-ray Integral Field Unit: a consolidated design for the system requirement review of the preliminary definition phase," *Experimental Astronomy*, vol. 55, no. 2, p. 373–426, 2023.
- [23] P. Peille et al., "The X-ray Integral Field Unit at the end of the Athena reformulation phase," *Experimental Astronomy*, vol. 59, no. 2, 2025.
- [24] European Space Agency, "NewAthena," ESA, [Online]. Available: <https://www.cosmos.esa.int/web/athena/about-athena>. [Accessed 19 November 2025].
- [25] Athena Community Office, "Athena Community Office," 02 May 2021. [Online]. Available: <https://www.the-athena-x-ray-observatory.eu/es/node/107>. [Accessed 19 November 2025].
- [26] N. Meidinger, J. Eder, T. Eraerds, N. Kirpal, D. Pietschner, M. Plattner, A. Rau and R. Strecker, "The Wide Field imager Instrument for Athena," *Instrumentation and Methods for Astrophysics*, 2017.
- [27] A. Lederhuber et al., "Detector Electronics Sub-System development for the NewAthena Wide Field Imager," Yokohama, Japan, 2024.
- [28] V. Antonelli et al., "The Wide Field Imager for the NewAthena mission: preliminary design and verification," in *SPIE Space Telescopes and Instrumentation*, Yokohama, Japan, 2024.
- [29] V. Antonelli, "Primary Structure Subsystem design and future planning," in *Athena WFI Consortium Meeting 17th*, Prague, 2024.

- [30] N. Meidinger et al., "Development Status of the Wide Field Imager Instrument for Athena," in *SPIE Space Telescopes and Instrumentation 2020*, 2022.
- [31] A. Mayr, "Camera Head Sub-System development," in *Athena WFI Consortium Meeting 17th*, Prague, 2024.
- [32] W. Treberspurg et al., "Studies of prototype DEPFET sensors of the Wide Field Imager of Athena," in *SPIE Optical Engineering + Applications 2017*, San diego, 2017.
- [33] M. Bonholzer et al., "First tests of large prototype DEPFET detectors for ATHENA's wide field imager," in *SPIE Space Telescopes and Instrumentation 2018*, Austin, USA, 2018.
- [34] J. Müller-Seidlitz et al., "Spectroscopic performance of flight-like DEPFET sensors for Athena's WFI," in *SPIE Space Telescopes and Instruments 2022*, Montreal, Canada, 2022.
- [35] S. Herrmann et al., "VERITAS 2.2 a low noise, source follower and drain current readout integrated circuit for the wide field imager on the Athena X-ray satellite," in *SPIE Astronomical Telescopes +Instrumentation 2018*, Austin, 2018.
- [36] W. TEAM, "Athena Comunity Website," WFI TEAM, [Online]. Available: <https://www.the-athena-x-ray-observatory.eu/es/node/423>. [Accessed 5 February 2025].
- [37] J. Müller-Seidlitz, "Athena Comunity Website," WFI TEAM, May 2022. [Online]. Available: <https://www.the-athena-x-ray-observatory.eu/en/node/113>. [Accessed 11 February 2025].
- [38] N. Meidinger et al., "The Wide Field Imager Instrument for Athena," in *SPIE: Space Telescopes and Instrumentation 2016*, 2016.

- [39] M. Plattner et al., "Technology development of Athena WFI frame processor electronics and verification of its real-time performance," in *SPIE: Space Telescopes and Instrumentation*, 2020.
- [40] J. Reiffers et al., "Hardware Development of Athena WFI Frame Processing Module," in *SPIE: Space Telescopes and Instrumentation*, Montreal, 2022.
- [41] Microchip Technology Inc., "RTG4 Fabric User Guide," 2023.
- [42] Frontgrade Gaisler AB, "GR740 Data Sheet and User's Manual," Frontgrade Gaisler, Gotenbog, 2024.
- [43] H. Schnetler, "NewAthena Wide Field Imager Project Instrument Control Sub System(ICS) development status and future plans," in *Athena WFI CM#17*, Prague, 2024.
- [44] Frontgrade Gaisler AB, "Frontgrade Gaisler," 2024. [Online]. Available: <https://www.gaisler.com/products/gr740#downloads>. [Accessed 19 02 2025].
- [45] M. Herrmann et al., "Mitigation of bandwidth limitation induced crosstalk on Athena's WFI," in *SPIE: Space Telescopes and Instrumentation 2022*, Montreal, 2022.
- [46] T. Schanz et. al., "A fast Event Preprocessor for the Simbol-X Low-Energy Detector," in *SPIE: Space Telescopes and Instrumentation* , 2008.
- [47] Microchip Technology Inc., "Microchip," Microchip, 2025. [Online]. Available: <https://www.microchip.com/en-us/products/fpgas-and-plds/radiation-tolerant-fpgas/rt-polarfire-fpgas>. [Accessed 26 06 2025].
- [48] NanoXplore, "NanoXplore," 2025. [Online]. Available: <https://nanoxplore.com/index.php/product/ultra300/>. [Accessed 26 06 2025].