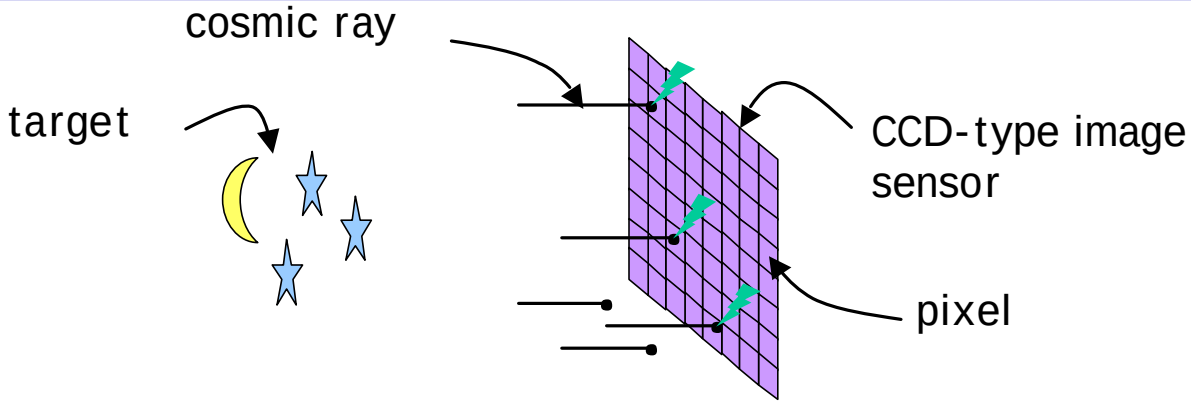
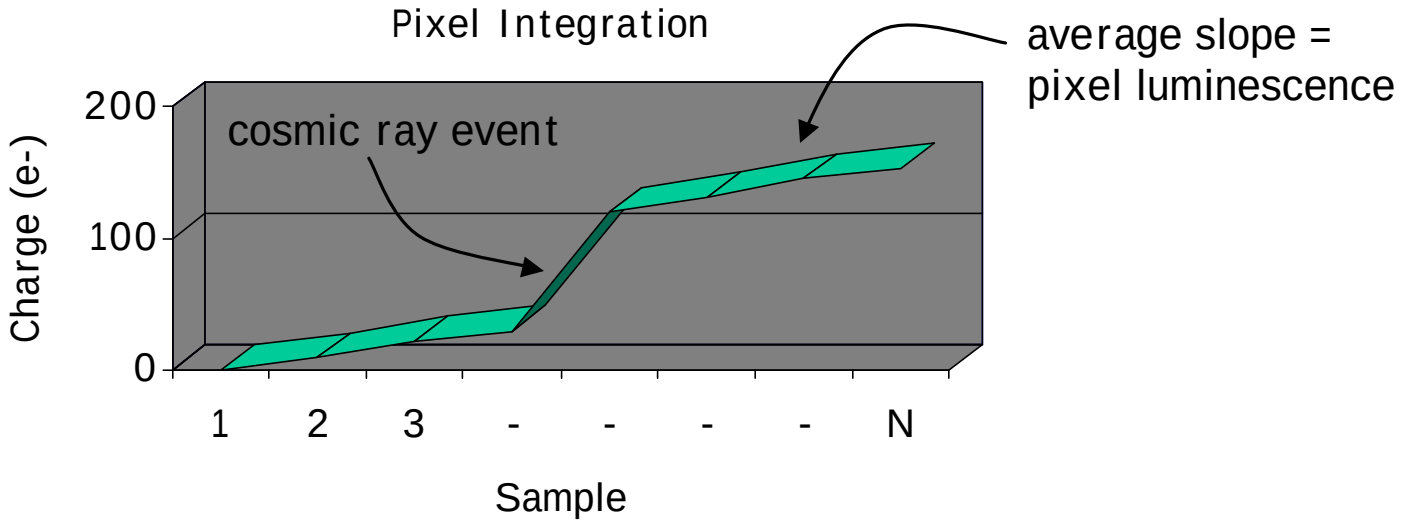
Next Generation Space Telescope Overview



- Some of the astronomical targets have extremely low luminescence
  - The image sensor is non-destructively sampled multiple times during an observation
  - The target image is hard to separate from background noise
  - The image sensor is susceptible to cosmic ray events
    - Such events inject charge into one or more pixels and appear as noise
    - These events can be filtered out using several known algorithms
- Filtering process is computationally intensive
  - Filtering must be done in-flight because downlink bandwidth is too low
  - Due to sample rate and frame size, each pixel must be processed in 2.5 us
  - The ability to change the filtering algorithm by uploading a new algorithm in-flight is required, precluding a hardware optimized implementation

Cosmic Ray Rejection Algorithm

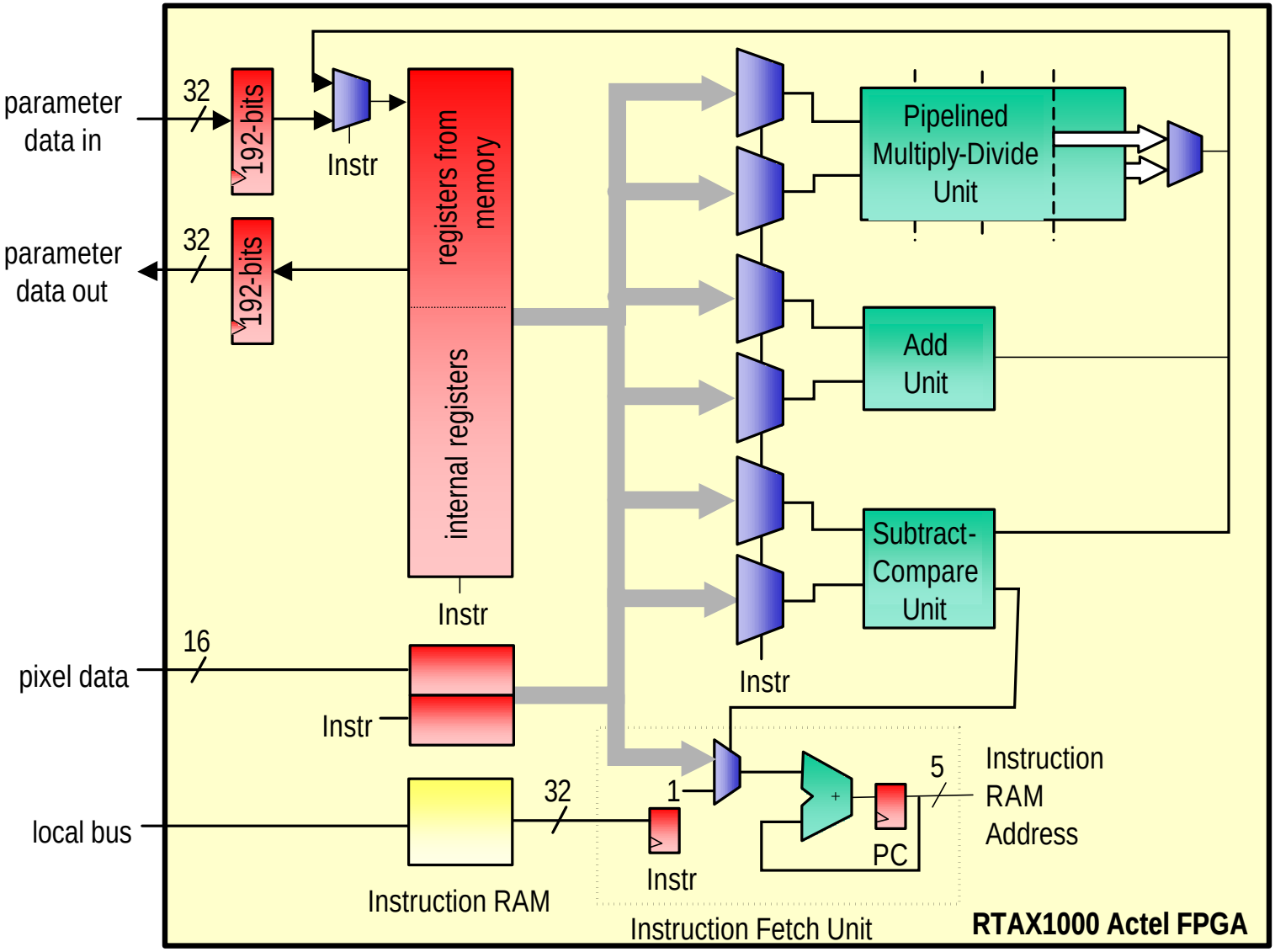


- The cosmic ray rejection algorithms studied by the NGST team operate on each pixel independently
- Several parameters are stored for each pixel in RAM
  - When a pixel is sampled, its parameters are fetched from RAM, the filtering algorithm is executed using and updating the parameter values, then the parameter values are stored back in RAM
- Even in its simplest form, cosmic ray rejection requires integer addition, subtraction, multiply, divide, and compare operations

Design Issues

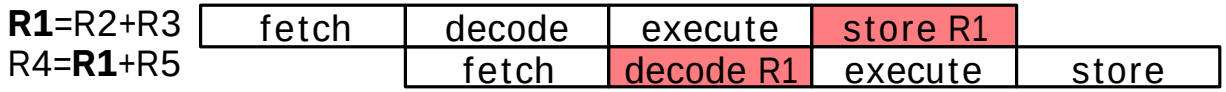
- Constraints
  - Radiation Environment - components must be radiation hardened or radiation tolerant
  - Board Area - the pixel processor for the image sensor is allocated 11 in^2
  - Power - the pixel processor for the image sensor is allocated 2.5W
  - Cost - NGST has budgeted for an FPGA design
- Component selection
  - An Actel RTAX1000 FPGA is the current baseline for the design
    - Internal RAM
    - 6,048 register cells
    - 12,096 combinatorial cells
    - 25 % of FPGA resources allocated to a programmable pixel processing unit
- Programmable Pixel Processing Unit
  - 32-bit arithmetic units operating in parallel
    - Add unit, Subtract/Compare Unit, Multiply/Divide Unit
  - 16 general purpose registers
  - 16.5MHz clock
  - NO HAZARD DETECTION LOGIC – MINIMAL CONTROL LOGIC

Programmable Pixel Processor Block Diagram

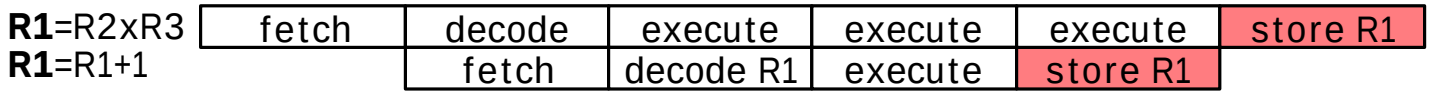


Pipeline Hazards

- Structural Hazard
  - Store Hazard
    - Multiple arithmetic units attempt to store output to same register
- Data Hazards
  - Read After Write
    - Instruction N+1 reads register before instruction N has a chance to write to it

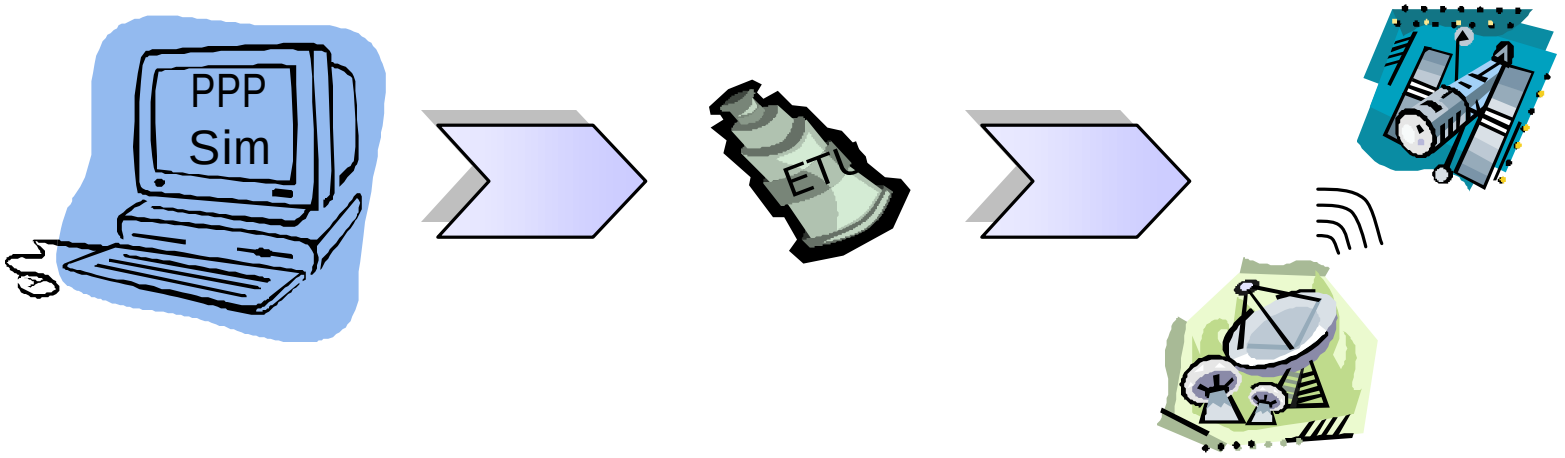


  - Write After Write
    - Multiply instruction N writes results after add instruction N+1 does



- Control Hazards
  - Branch taken
    - Instruction N+1 is executed although branch instruction N was taken
- Detecting and preventing pipeline hazards consumes a considerable amount of FPGA resources, especially when arithmetic units operate in parallel
- In some applications, this responsibility can be shifted from hardware to software

New Algorithm Deployment



- New algorithms developed using a PPP software simulator
  - Program is developed using the PPP machine language
  - Verify that no pipelining hazards exist in program
- Algorithm is tested on the ground using the engineering test unit, ETU (identical to flight hardware)
  - Verify proper behavior of the program on the actual hardware
- Upload the program for the new algorithm to the telescope
- This deployment process allows the hardware to remain as simple as possible, allowing it use minimal FPGA resources, draw minimal power, and provide high performance