Appendix 0: Main Body Support

References and Further Investigation

- J. Moore, M. McLean, “FPGA Single Chip Cryptographic Solution”, Proceedings of the 2006 SDR Technical Conference and Product Exhibition http://www.sdrforum.org/pages/sdr06/sdr06_papers/1.3/1.3-04.pdf
  - More on SCC: http://www.xilinx.com/products/silicon_solutions/market_specific_devices/aero_def/capabilities/crypto.htm
- C. Carmicael, J. George, G. Miller, G. Swift, G. Allen, CW Tseng, “Static Upset Characteristics of the 90nm Virtex-4QV FPGAs”, Proceedings of the 2008 NSREC Technical Conference
  - More on Virtex-4QV and mitigation http://www.xilinx.com/products/silicon_solutions/aero_def/4qv_promo.htm
- XAPP988 Correcting SEUs in Virtex-4 Platform FPGA Configuration Memory
- XAPP989 Correcting SEUs with a Self-Hosting Configuration Management Core
- XAPP1004 Single-Event Upset Mitigation Design Flow for Xilinx FPGA PowerPC Systems
- XAPP962 : Single-Event Upset Mitigation for Xilinx FPGA Block Memories
- XAPP1051 : Single Event Upset Mitigation Design Flow for Xilinx FPGA MicroBlaze Systems

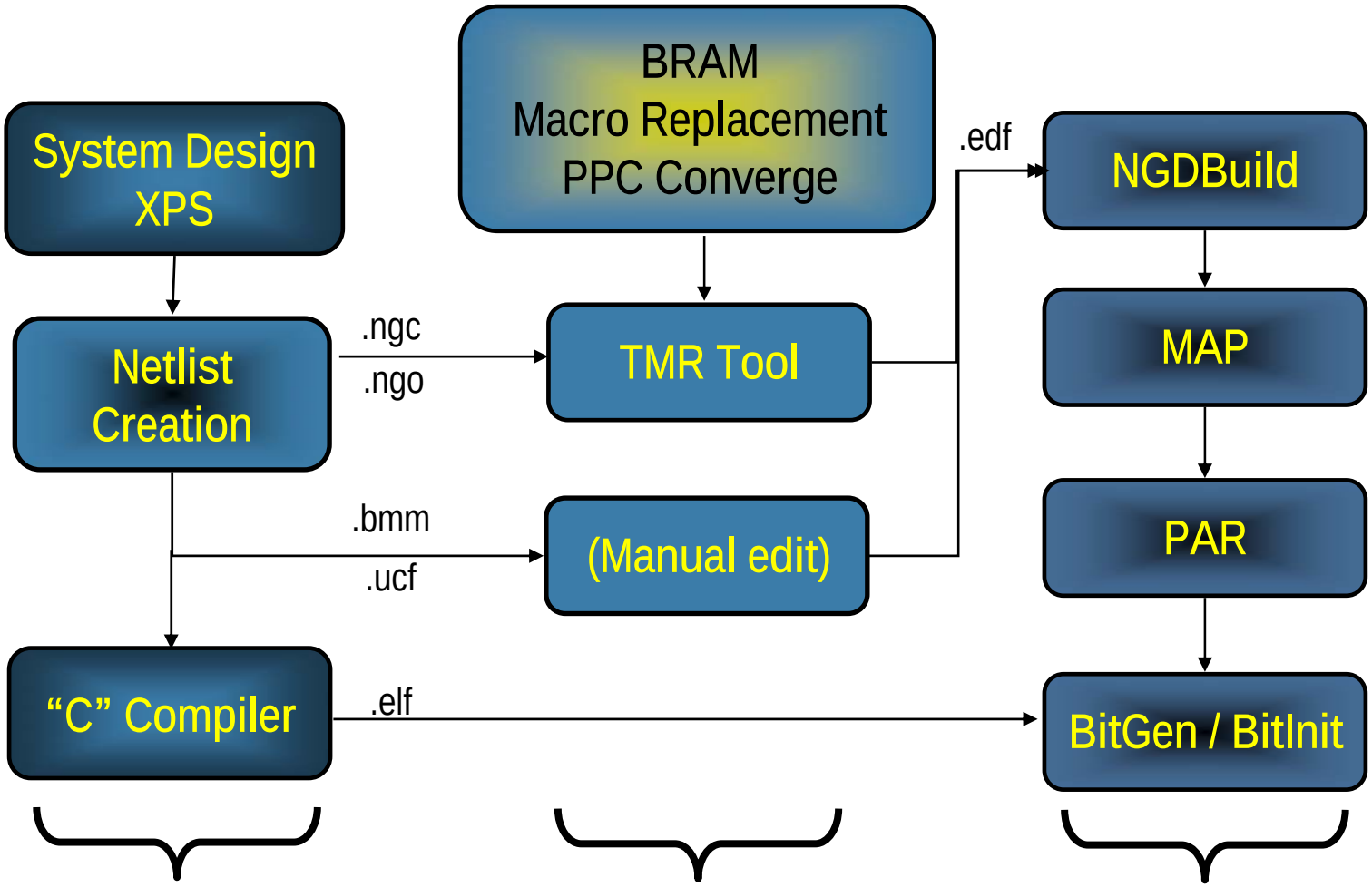
MAPLD Applications Section - *Secure SDR in Space Abstract*

Software Defined Radio (SDR) has now become a common practice. What was the subject of debate and dreams a decade ago has now become reality in as small a form factor as hand-held radios for soldiers and in micro-satellites. Many of these communications transmit sensitive information for government, military or financial applications. Thus, they require the data to be encrypted. Of course, SDR in space is an extra challenge because it requires high-performance computing in a radiation-intensive environment. In this session, we discuss the architecture of a card which can provide secure SDR in space. Then, we look into a software flow for how this application can be implemented with modern high-performance, radiation-tolerant and reprogrammable FPGAs. This flow combines a unique methodology for secure communications which has multiple independent levels of security and isolation of the channels along with intellectual property made redundant by using a software tool that automates the process of triple modular redundancy. This discussion will consider the large variety of waveforms and cryptographic algorithms which can be performed by an FPGA, even looking at how one might change waveforms through partial reconfiguration with the satellite while in service. Finally, we will discuss some good design practices for intellectual property and the overall board subsystem in space.

Software Flows: Tool Elements

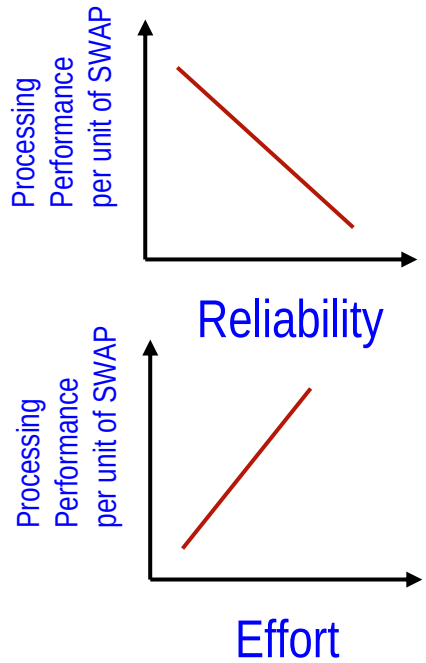
- Basic Hardware Flow
  - Block-level generation
  - Design / Code Entry
  - Simulation
  - Synthesis / physical synthesis
  - Simulation
  - Place and route
  - Timing closure/back annotation
  - Bitstream Generation
- Other useful tools
  - DSP System & Simulation
    - Accelchip (DSP synthesis)
    - Simulink / System Generator
  - MPU SW Development
    - Platform Studio
    - Embedded Development Kit
  - PlanAhead (Floorplanner)
  - Chipscope (Debug / Probe)

XPS / TMR Tool Design Flow



Mitigation Considerations

- Methods are application & orbit dependant
  - SWAP constraints
  - Processing performance
  - Reliability requirements
  - Design schedule
  - Type of data and peripherals
  - Latency constraints
- Weigh factors before implementing an approach
- Designs often use multiple mitigation methods



V4-RCC Board Supports Wide Range of SEE Mitigation Approaches

Courtesy of SEAKR Engineering

Virtex-4QV Orbital Rates

Orbit	LEO	LEO	POLAR	CONST.	GEO
Altitude (km)	400	800	833	1,200	36,000
Inclination	51.6°	22.0°	98.7°	65.0°	0°

Functional Interrupts (All Virtex-4)

Upsets/Device-Day
Typical Solar Conditions

POR	2.83E-06	2.73E-05	1.85E-05	7.36E-05	1.21E-05
SMAP+FAR	2.25E-06	2.45E-05	1.69E-05	6.71E-05	9.46E-06
GSIG	1.57E-06	2.41E-05	1.57E-05	6.47E-05	4.87E-06

Device-Years Between Event

All SEFIs (Combined)	**412**	**36**	**53**	**13.3**	**103**

Configuration Memory Cells

Upsets/Device-Day
Typical Solar Conditions

XQR4VSX55	0.76	7.43	5.12	20.0	4.20
XQR4VFX60	0.80	7.79	5.36	20.9	4.40
XQR4VFX140	-	-	-	-	-
XQR4VLX200	2.15	21.0	14.5	56.5	11.9

Notes:

1) Unmitigated Device Performance
2) Reference: Gary Swift et al. – NSREC Poster Presentation; *Static Upset Characteristics of the 90nm Virtex-4QV FPGAs*

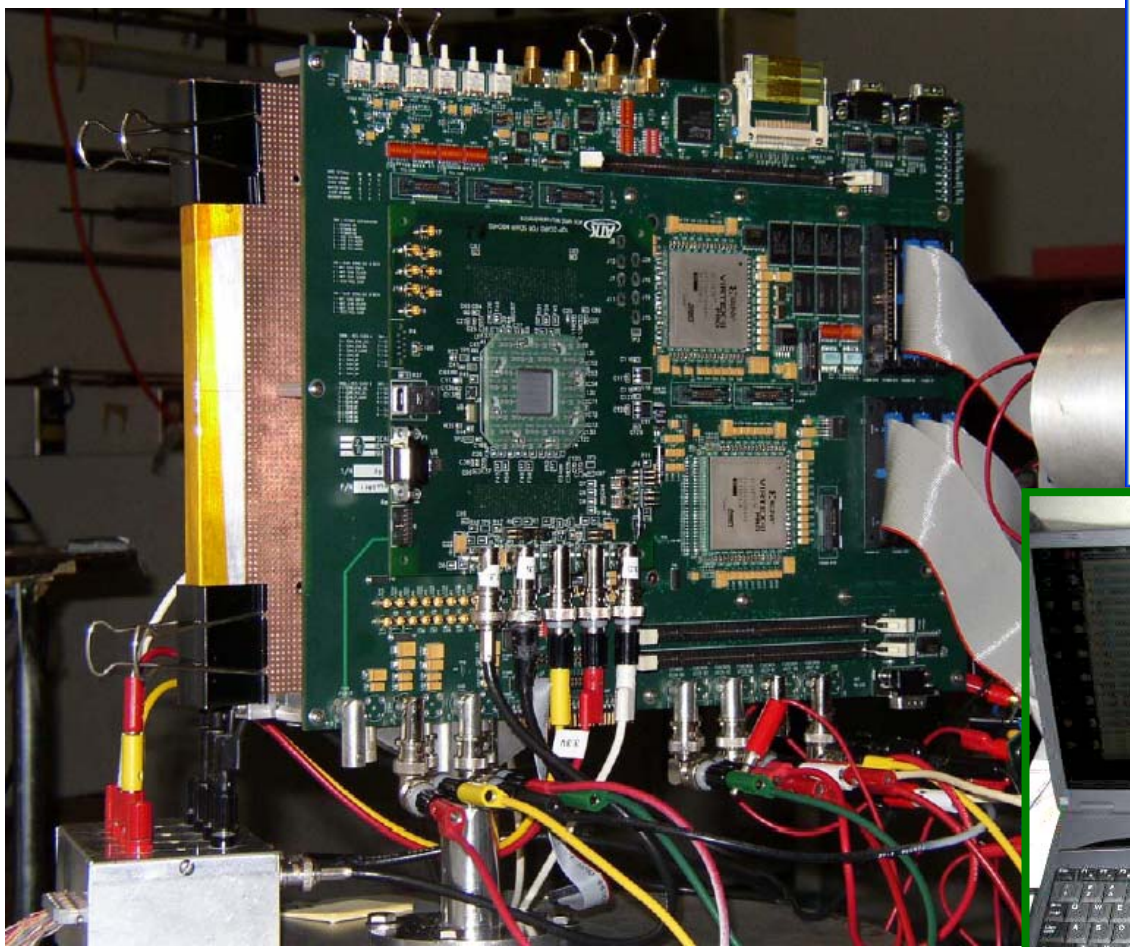
XILINX®

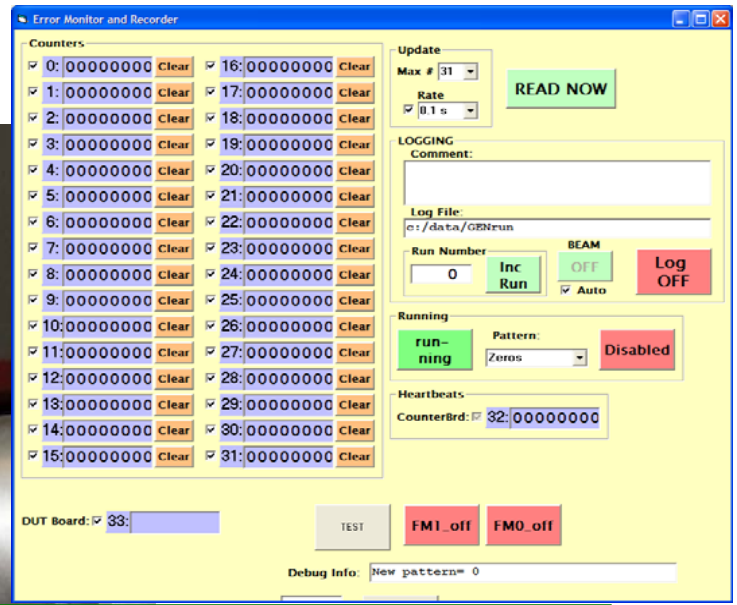
SEE Mitigation Effectiveness

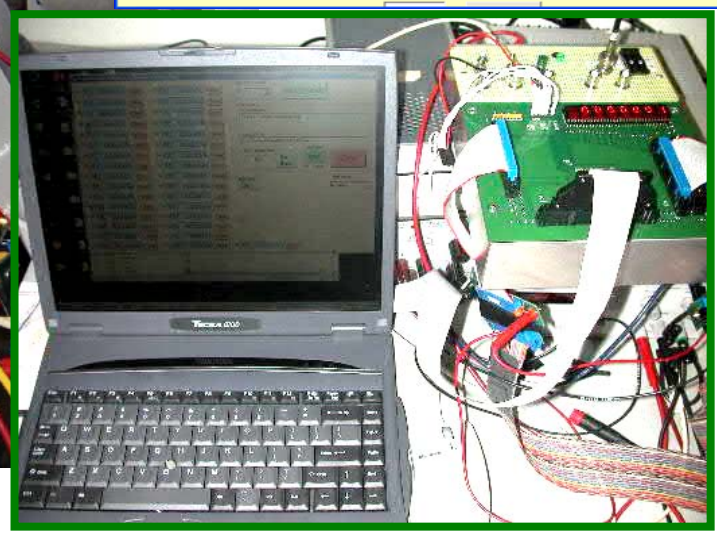
Implementing triple modular redundancy allows a design to operate correctly in spite of any single upset or transient in either the configuration memory or user resources. Constant monitoring and scrubbing of the configuration fixes upsets as they happen. Together they can lower the system error rate due to multiple upsets to *well below the SEFI rate of once per century in geosynchronous orbit*. Adding SEFI detection allows a quick reconfiguration, thus maximizing system availability. For example, if detecting a SEFI, reconfiguring, and resetting can be done in a leisurely ten seconds, then availability is better than eight nines (or ten seconds outage in over 3 billion cycles of correct operation in GEO).

Reference: Gary Swift et al. – NSREC Poster Presentation
Static Upset Characteristics of the 90nm Virtex-4QV FPGAs

Fault Injection Test Setup







XILINX®

IP Considerations

- Minimum deliverables with Intellectual Property
  - Core, testbench, Documentation, Development boards
- Better IP will have
  - Design and verification to requirements, understanding of ‘designer reuse’ and/or heritage
- Space Considerations
  - Fail safe state machines
  - 100% code and functional coverage
  - Document error messages/anomolies
  - 3rd-party review of code

Appendix 1: C to FPGA

- Following slides and slides in the main body on this subject are courtesy of ImpulseC

C to FPGA for SDR

Project Vision

- Develop key algorithm libraries for SDR
- Develop key platform integration libraries
- Port reference designs to top 5 COTS platforms

Expected Results

- Improve productivity
  - Easier programming of SDR applications
  - More reusable, easily understood, C-based libraries and building blocks
  - Easier interoperability with other tools through C input and VHDL/Verilog output
- Increase portability via device independent design
- Preserve government investment in SDR building blocks

C to FPGA Current Use in Mil/Aero

- Air Force
- Army
- Battelle
- DOD
- DRA Associates
- General Dynamics
- Lawrence Livermore
- Lockheed Martin
- Los Alamos
- MNB Technologies
- Navy
- NASA
- Northrop Grumman
- NSA
- Oak Ridge National Laboratory
- Raytheon
- Sandia

This is a partial list

Target Gains for Offloading to FPGA

- Fir Filters — 2 – 10x
- Encryption — > 100x
- Up/Down Decode — > 10x

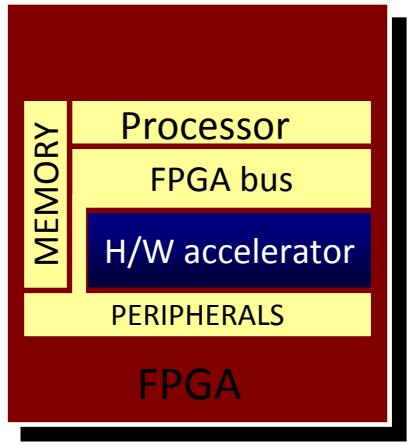
SDR System Elements: Partial List

The FPGA as an Accelerator

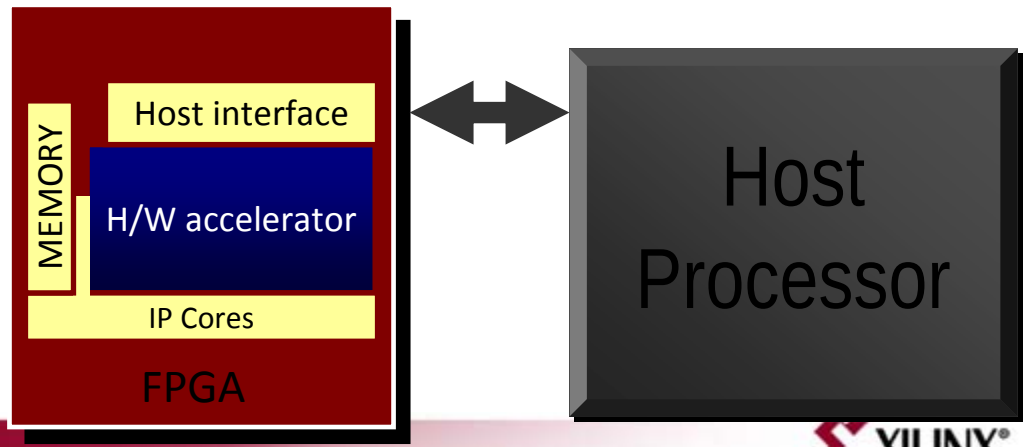
- Compile C code into a processor-attached accelerator
- Processor may be embedded within the FPGA, or external
  - PowerPC
  - MicroBlaze
  - Intel
  - AMD

Note: a processor is not required to use Impulse C

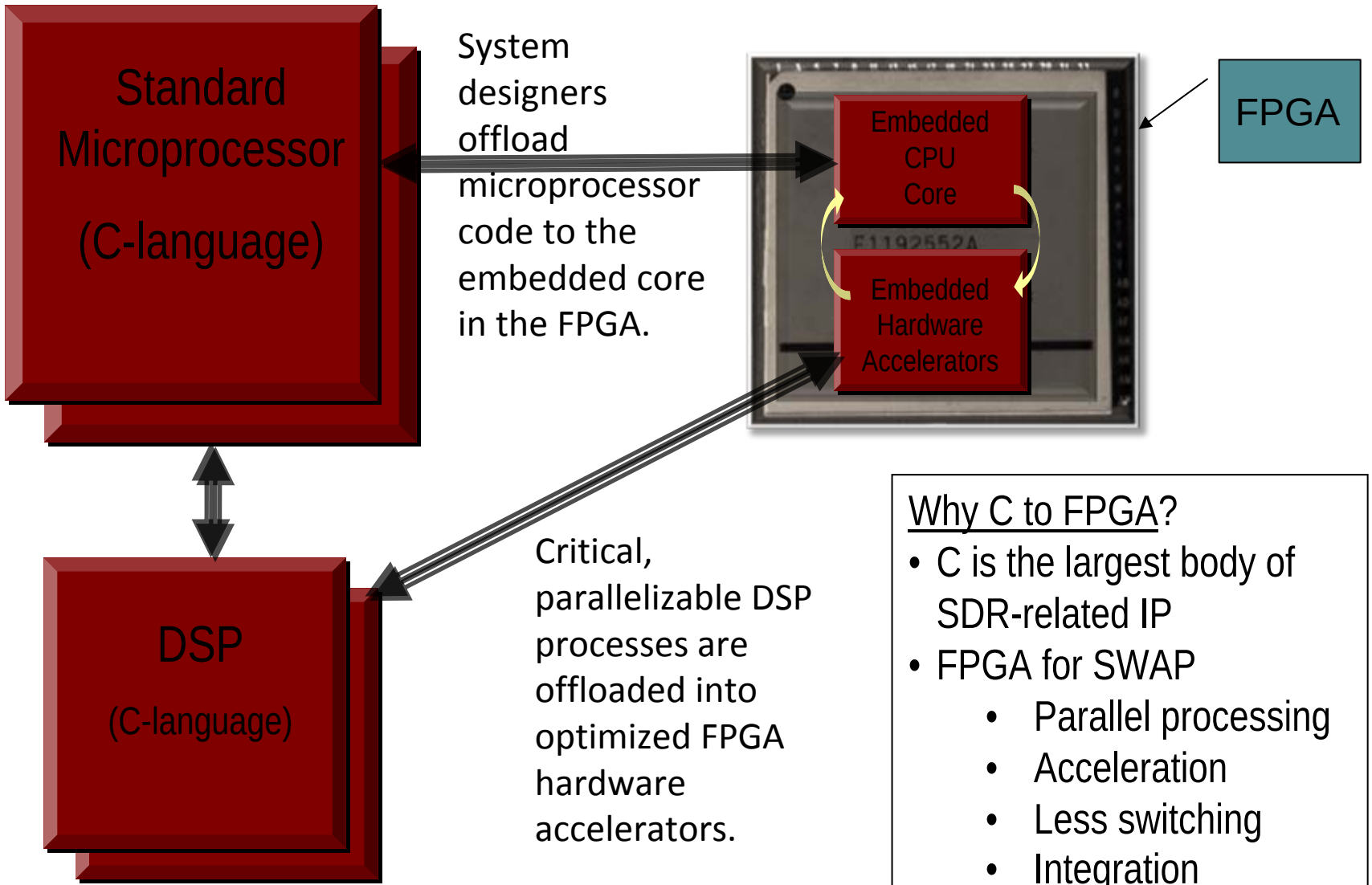
For embedded processing…



For co-processing…



FPGAs Offload Both Processor & Hardware Code



Why C to FPGA?

- C is the largest body of SDR-related IP
- FPGA for SWAP
  - Parallel processing
  - Acceleration
  - Less switching
  - Integration

Courtesy of ImpulseC

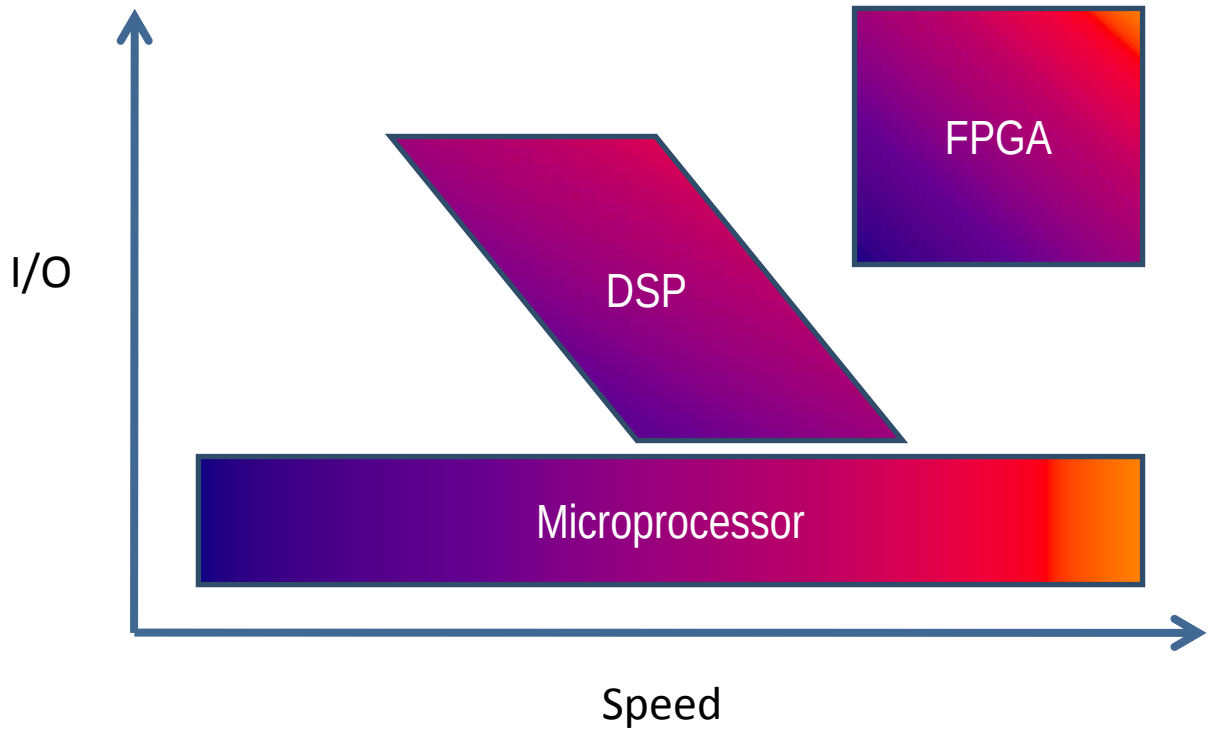
SDR with Processors Integrated

All can be programmed in ANSI C.

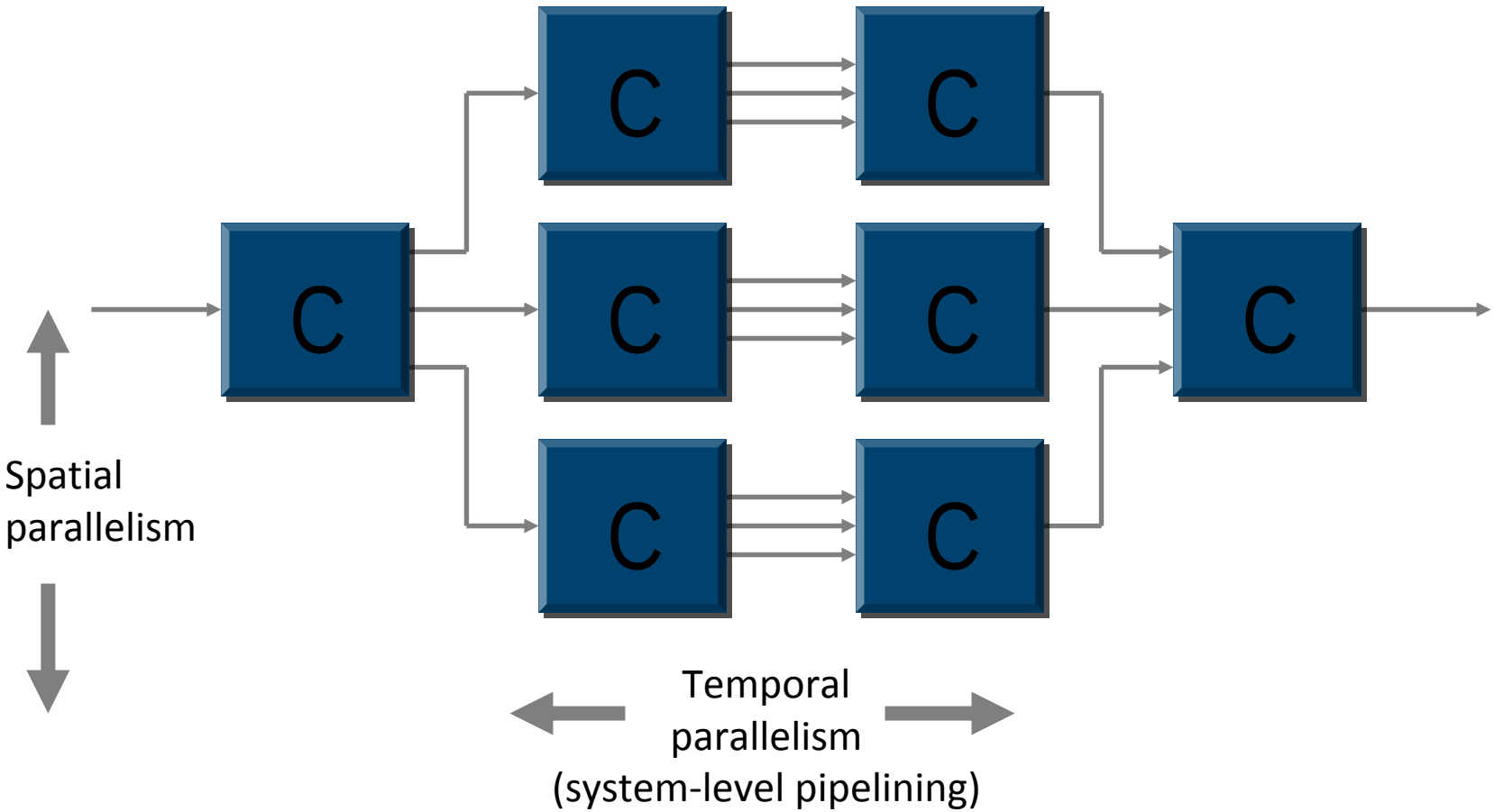
Device:	Microprocessor
Strength:	Single or dual channel processing with high memory access
SDR Uses:	Protocol management
	Low-frequency waveform processing
	Transmit and Receive management
Device:	DSP
Strength:	Single channel, higher performance signal manipulation
SDR Uses:	Mod/Demod
	FEC
	Low and medium speed I/O processing
Device:	FPGA
Strength:	High I/O, multi-process, with optimized logic elements
SDR Uses:	Incoming signal, data (e.g. encryption) or image processing
	Up/Down conversion
	Signal Filters

Each Device Has I/O & Speed Strength

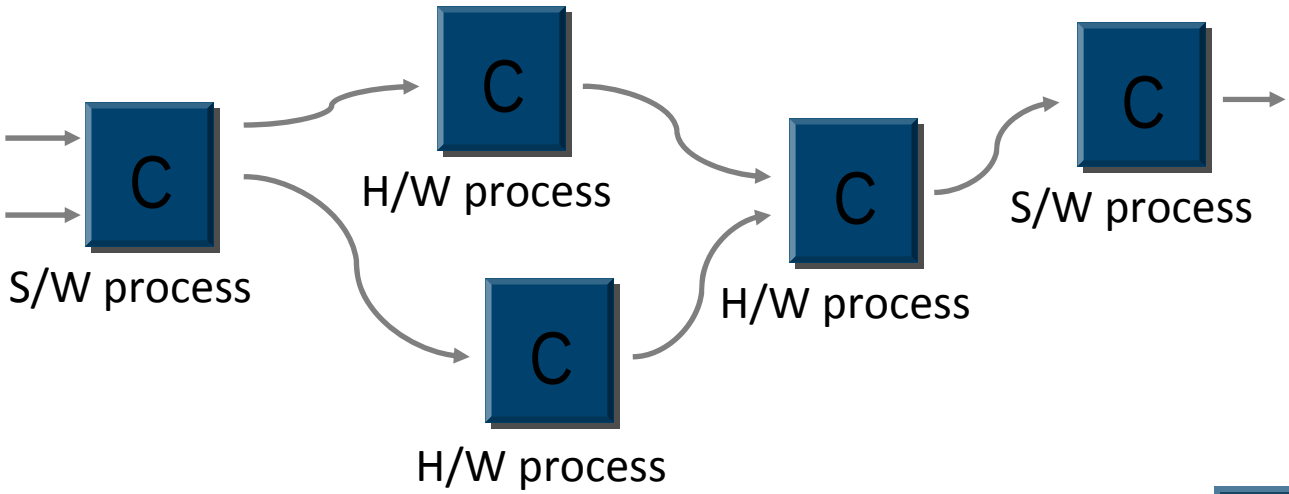
Power also increases with speed and # of processes



FPGAs Parallelize in Time or # of Processes to Achieve Performance at Lower Power

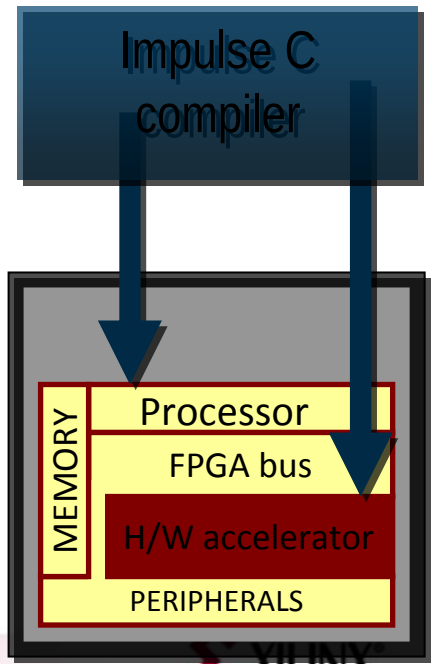


FPGA C Programming Model



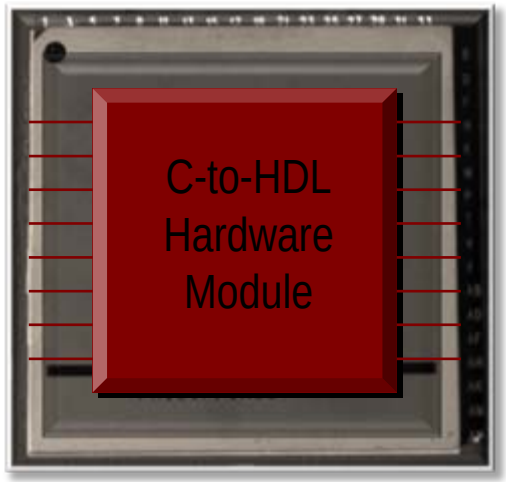
Communicating C-Language Processes

- Buffered communication channels to implement data streams
- Supports dataflow and message-based communications
- Supports parallelism at the application level and at the level of individual processes



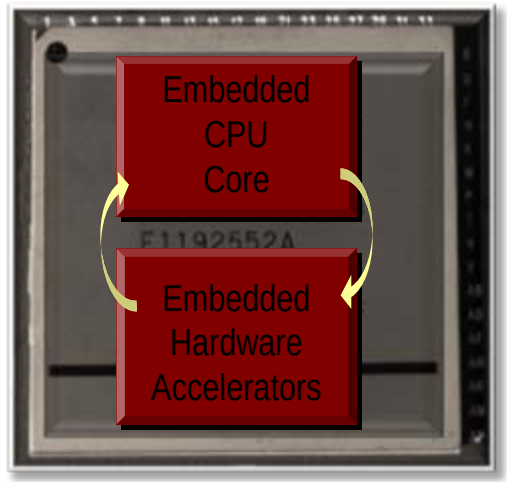
FPGA Co-Processing Partitioning Options

Option 1



Create a hardware module

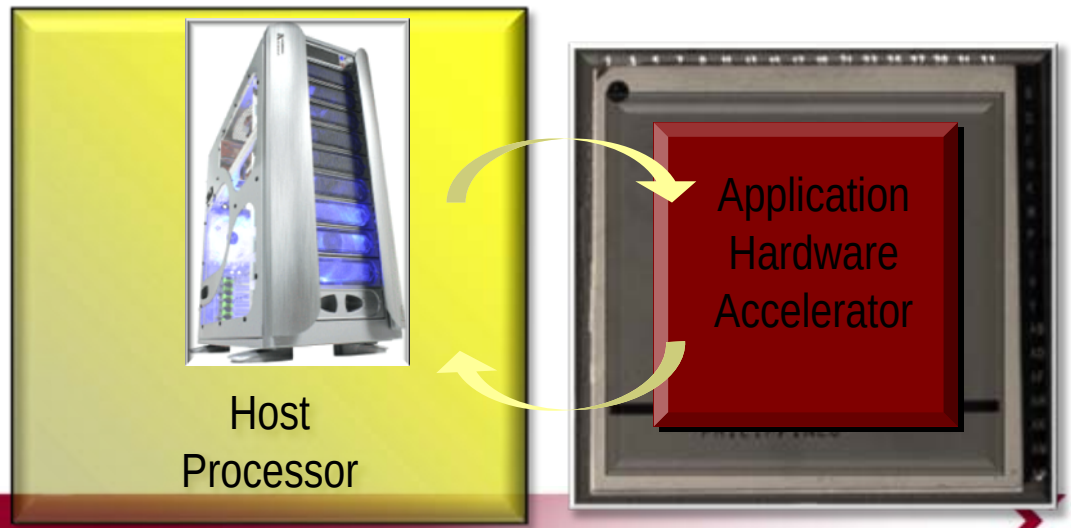
Option 2



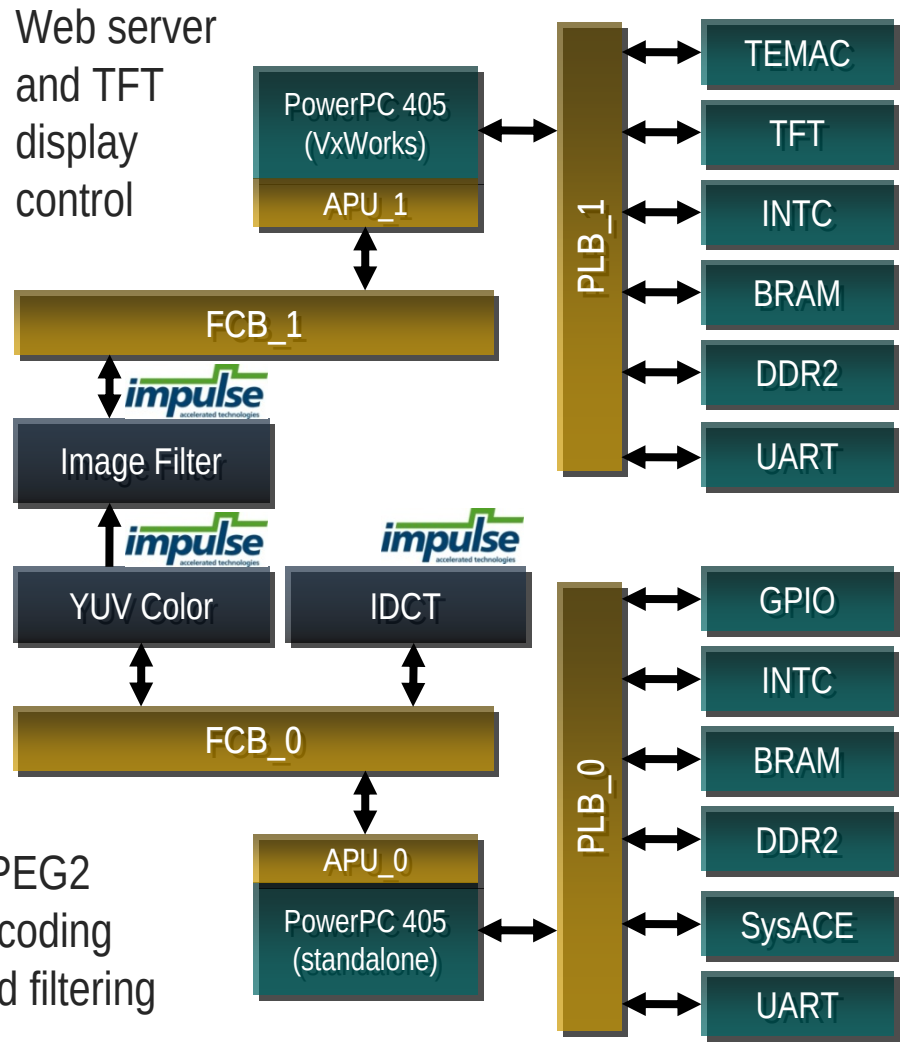
Accelerate an embedded CPU

Option 3

Accelerate an external/host CPU



Dual PowerPC Embedded Computing

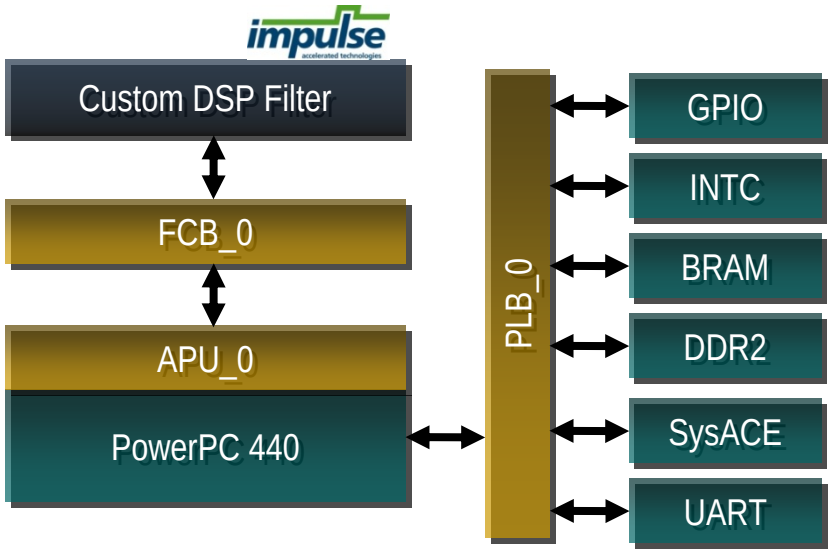


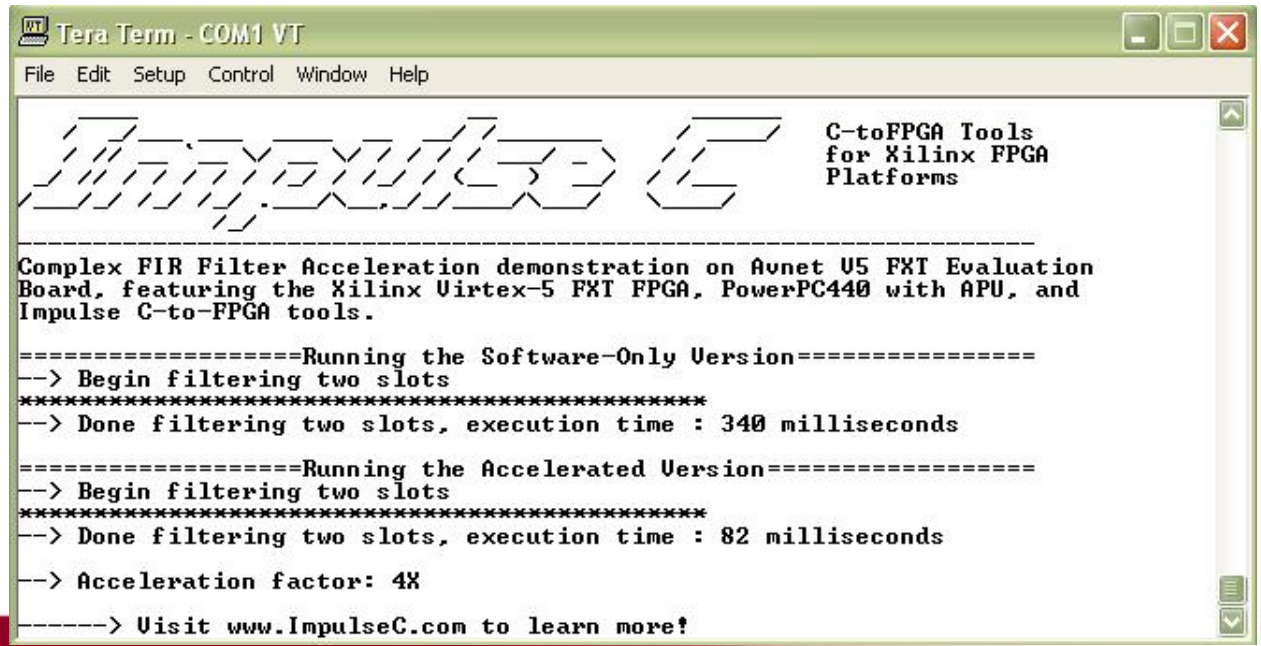
Dual PowerPC Acceleration

- Embedded web server, camera interface and accelerated filtering
- Two PowerPC processors, both with Impulse C acceleration
- Featuring the Xilinx ML410 development board and Virtex-4 FX60 FPGA

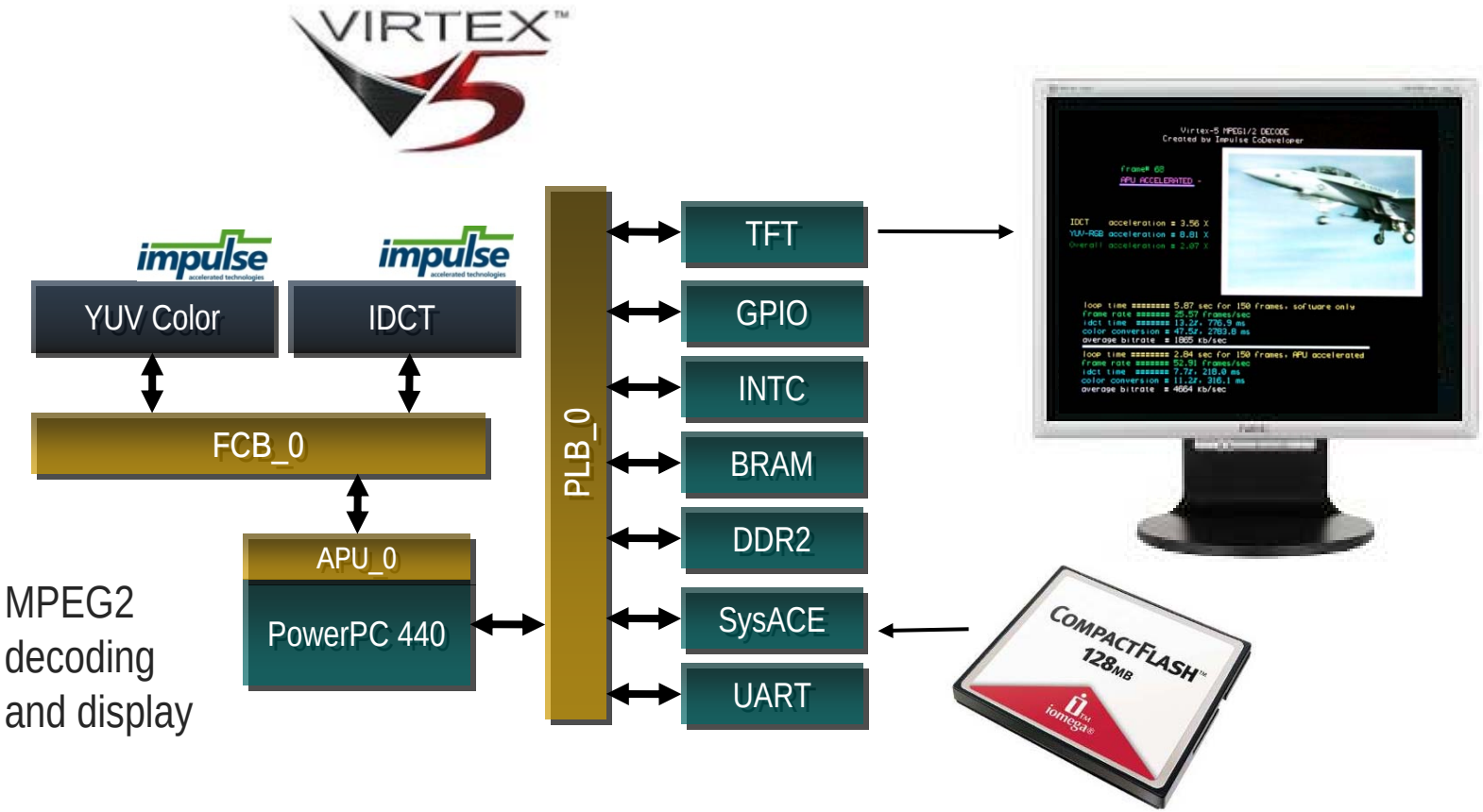
FPGA/DSP Co-Design Example MPEG2

DSP filter function accelerated Using an FPGA coprocessor





PowerPC Embedded Computing



XILINX

Accelerated Process Library Example Showing Simple Doubling
MPEG Process Accelerated 2 – 8X Over CPU

frame# 68
APU ACCELERATED -

IDCT acceleration = 3.56 X
YUV-RGB acceleration = 8.81 X
Overall acceleration = 2.07 X

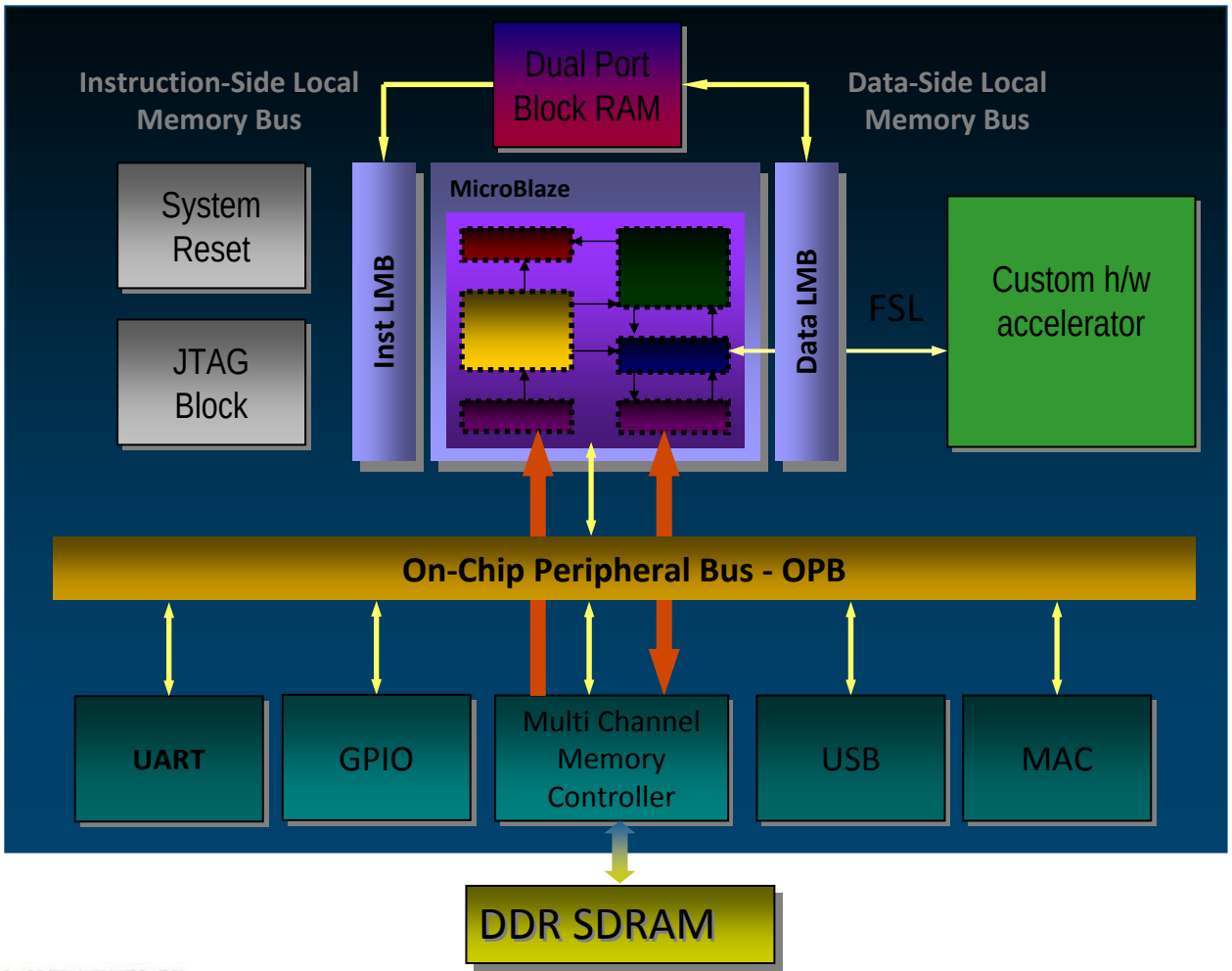
loop time ======== 5.87 sec for 150 frames, software only
frame rate ======= 25.57 frames/sec
idct time ======= 13.2%, 776.9 ms
color conversion = 47.5%, 2783.8 ms
average bitrate = 1865 Kb/sec

loop time ======== 2.84 sec for 150 frames, APU accelerated
frame rate ======= 52.91 frames/sec
idct time ======= 7.7%, 218.0 ms
color conversion = 11.2%, 316.1 ms
average bitrate = 4664 Kb/sec

Frame rate doubled by offloading critical computations to the FPGA

Example: FPGA Embedded System

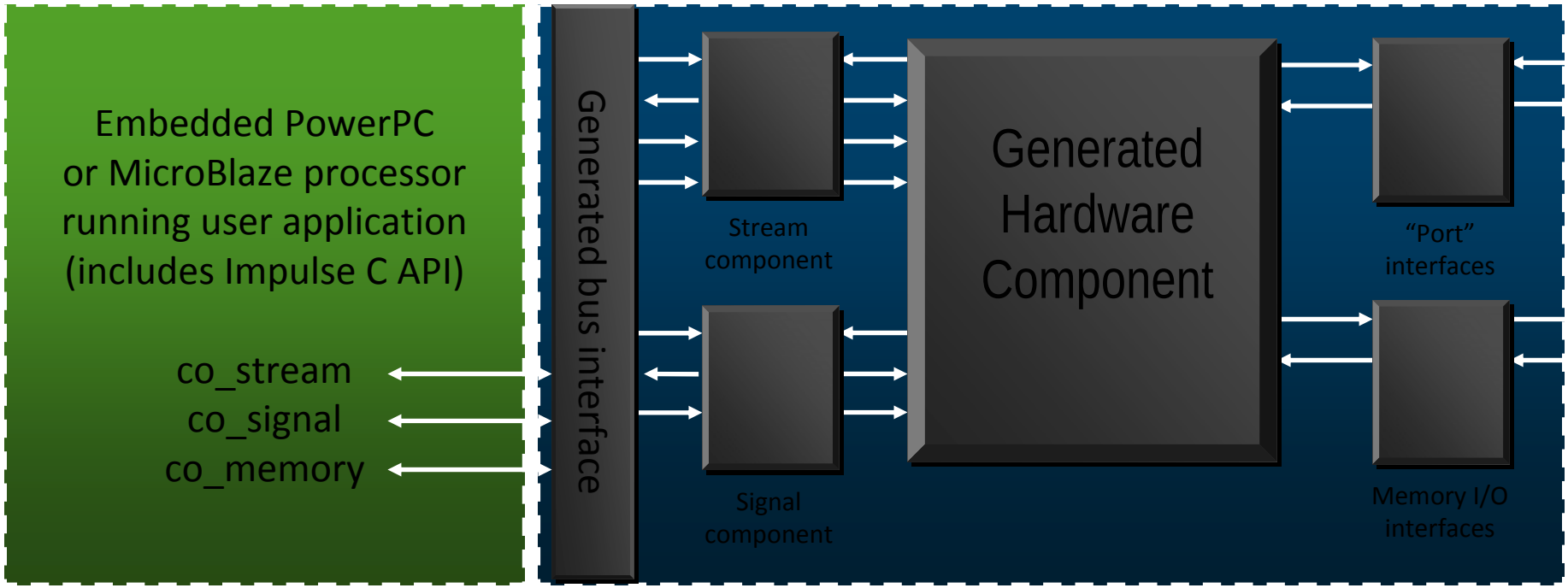
Using Xilinx Virtex-5 with MicroBlaze soft processor



impulse accelerated technologies

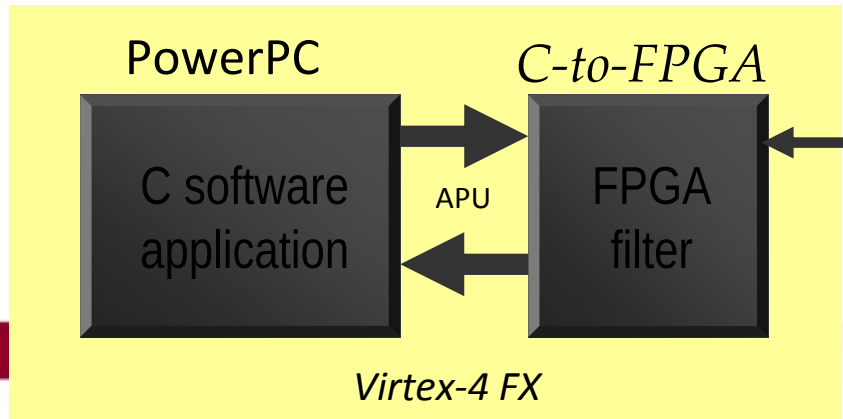
```
void img_proc(co_stream pixels_in, co_stream pi
   int nPixel;
   . . .
   do {
      co_stream_open(pixels_in, O_RDONLY, INT
      co_stream_open(pixels_out, O_WRONLY, IN
      while ( co_stream_read(pixels_in, &nPixel, si

         . . .
         // Do a filtering operation here using standa
         . . .

         co_stream_write(pixels_out, &nPixel, sizeo
      }
      co_stream_close(pixels_in);
      co_stream_close(pixels_out);
   } while(1);                                  // Run forever
}
```

Embedded Processor Acceleration



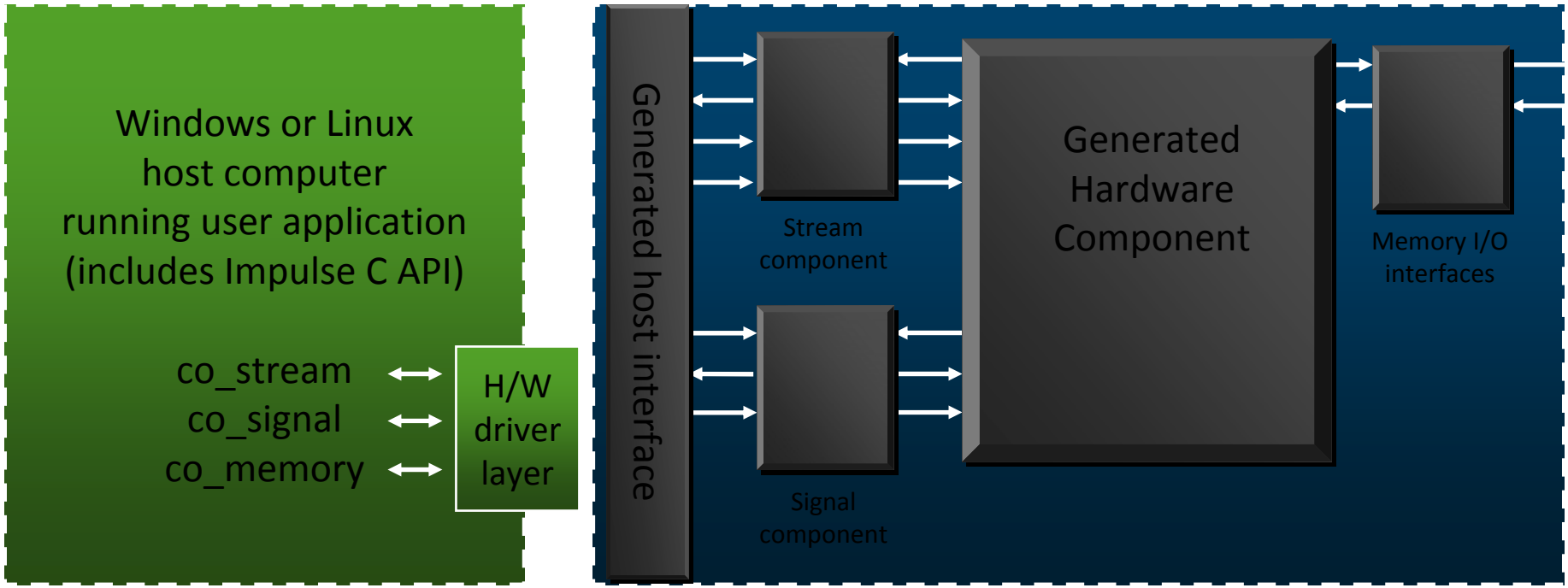
Generated hardware module/peripheral

Streaming filter attached to embedded processor

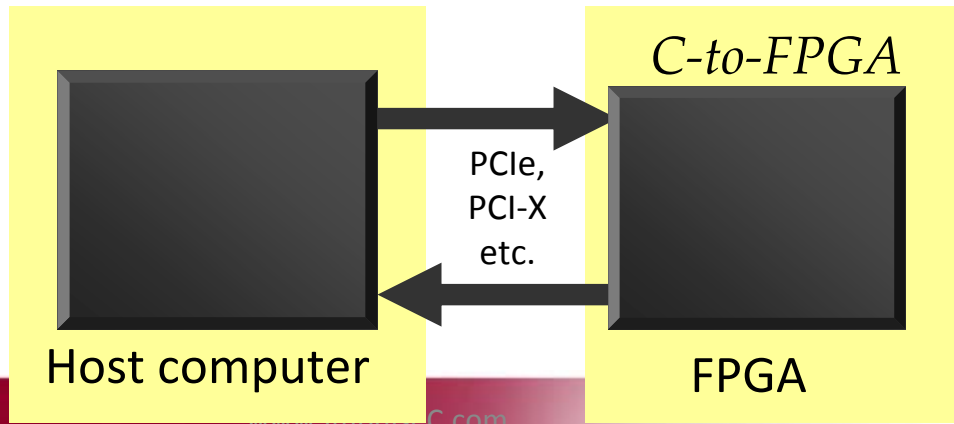


XILINX®

Host Processor Acceleration

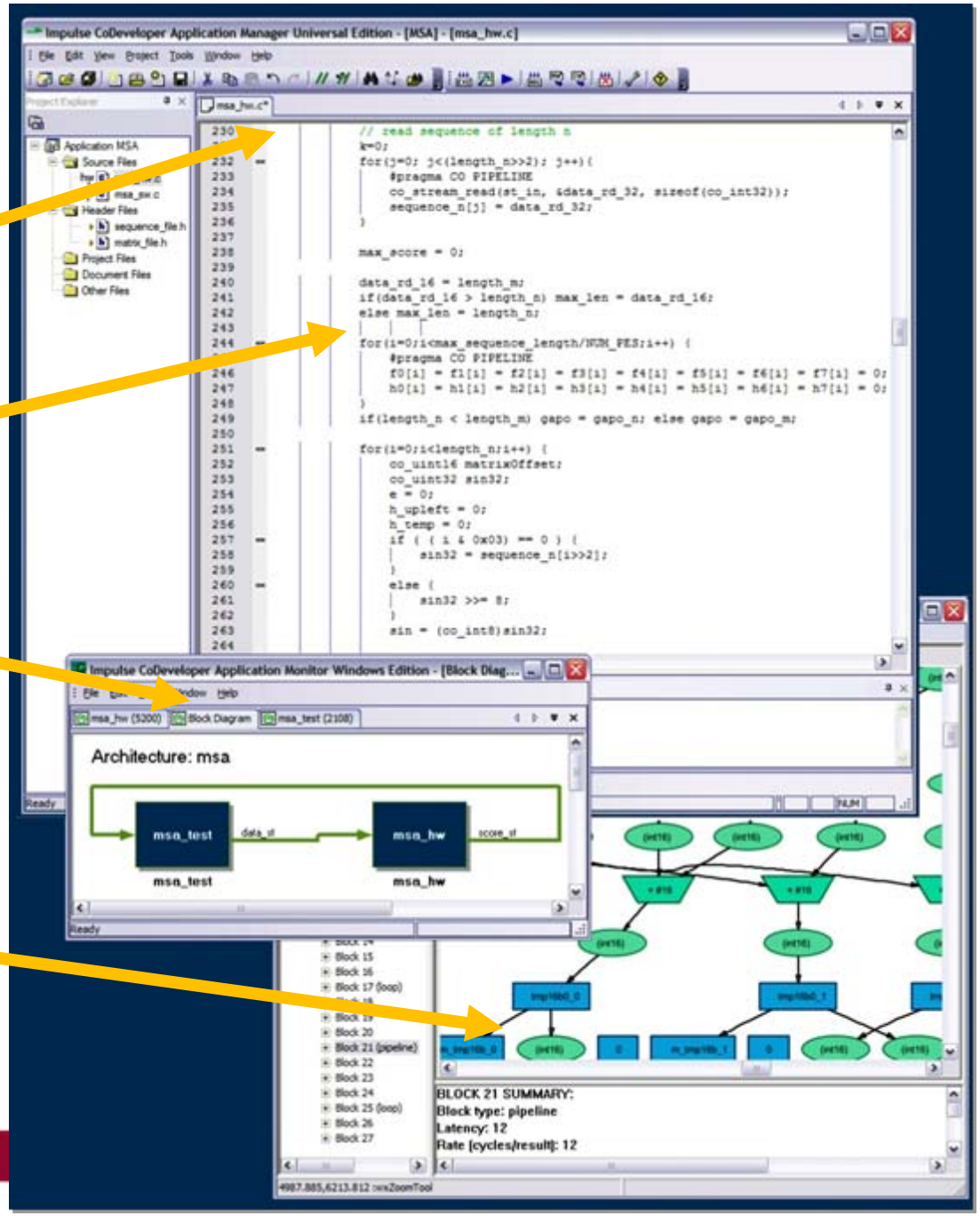


Host processor acceleration example



Impulse Flow

- Familiar IDE Interface
- Desktop simulation
  - Using standard C programming methods
- Dataflow analysis
  - Identify bottlenecks
- Graphical optimization analysis
  - Balance size and speed to meet performance goals



Impulse C Streaming Process

```
void img_proc(co_stream pixels_in, co_stream pixels_out) {
   int nPixel;
   . . .
   do {
      co_stream_open(pixels_in, O_RDONLY, INT_TYPE(32));
      co_stream_open(pixels_out, O_WRONLY, INT_TYPE(32));
      while ( co_stream_read(pixels_in, &nPixel, sizeof(int)) == 0 ) {

         . . .
         // Do a filtering operation here using standard C…
         . . .

         co_stream_write(pixels_out, &nPixel, sizeof(int));
      }
      co_stream_close(pixels_in);
      co_stream_close(pixels_out);
   } while(1);                    // Run forever
}
```

Impulse C streaming API functions compile automatically to generate processor bus Interfaces.

Impulse C API Functions

co_memory_create
co_memory_ptr
co_memory_readblock
co_memory_writeblock

co_process_config
co_process_create

co_register_create
co_register_get
co_register_put
co_register_read
co_register_write

co_semaphore_create
co_semaphore_release
co_semaphore_wait

co_signal_create
co_signal_post
co_signal_wait

co_stream_close
co_stream_create
co_stream_eos
co_stream_open
co_stream_read
co_stream_read_nb
co_stream_write
co_stream_write_nb

cosim_logwindow_create
cosim_logwindow_fwrite
cosim_logwindow_init
cosim_logwindow_write

Impulse C Shared Memory Process

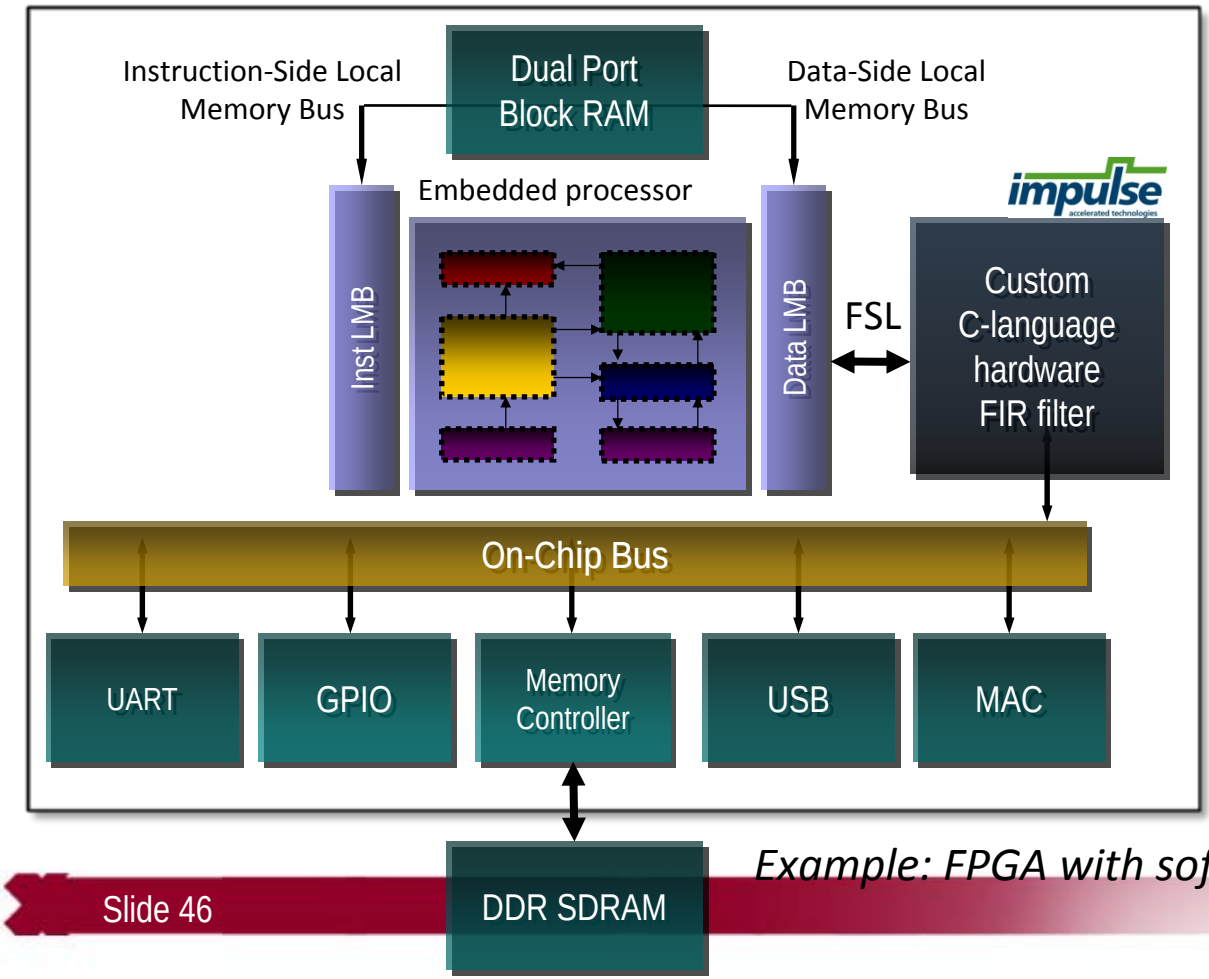
```
void img_proc(co_signal start, co_memory datamem, co_signal done) {
   double A[ARRAYSIZE];
   double B[ARRAYSIZE];
   int32 status;
   int32 offset = 0;
   . . .
   do {
      co_signal_wait(start, (int32*)&status);
      co_memory_readblock(datamem, offset, A, ARRAYSIZE * sizeof(double));

      . . .
      // Do some kind of computation here, perhaps calculating A[] into B[]
      . . .

      co_memory_writeblock(datamem, offset, B, ARRAYSIZE * sizeof(double));
      co_signal_post(done, 0);
   } while(1);
}
```

Device Independent Libraries w/ SDR

- Quicker Results
  - Tie into C, the biggest body of IP
  - Ensure that new libraries are portable
  - Open up the world of COTS FPGA/DSP/uP platforms to software engineers
- Reduced Risk
  - Preserve the value of the government's investment in IP
  - Make it much easier to jump to better hardware components as they emerge
  - C as input and VHDL or Verilog as output links to wide array of tools
- More Cooperation Between Groups
  - Chips away at hardware incompatibility by linking to a compelling, standard body of IP
  - Increases COTS hardware compatibility
  - Defer hardware dependencies to later in the process and onto the hardware manufacturers

Embedded Computing for SDR

Combine embedded processors with custom C-language accelerators



Example: FPGA with soft processor

Easily partition your application between the embedded processor and FPGA logic

Verify using standard C debugging tools

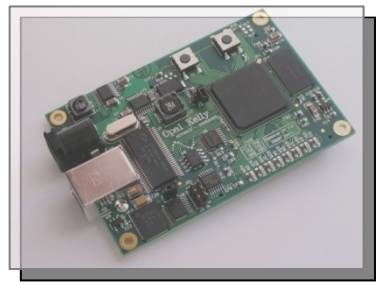
Optimize your application for high performance

XILINX®

FPGA-Based Platforms

How to match a platform to your project?

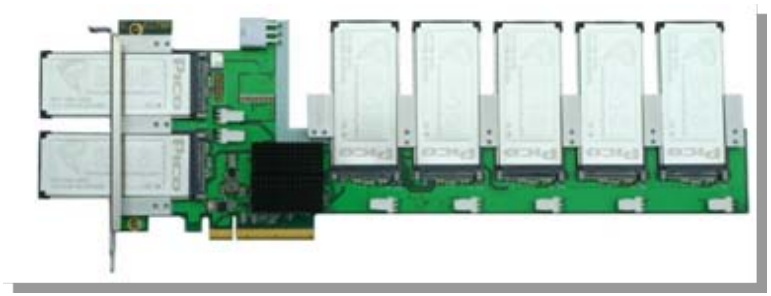
- TYPE OF FPGA
  - What resources are available for computation?
- HOST PROCESSOR
  - Embedded within the FPGA, discrete, or none at all?
- BUS ARCHITECTURE
  - How is the FPGA connected to the processor?
- OTHER PERIPHERALS
  - What devices will the FPGA application be communicating with?
- OPERATING SYSTEM
  - For the host processor?

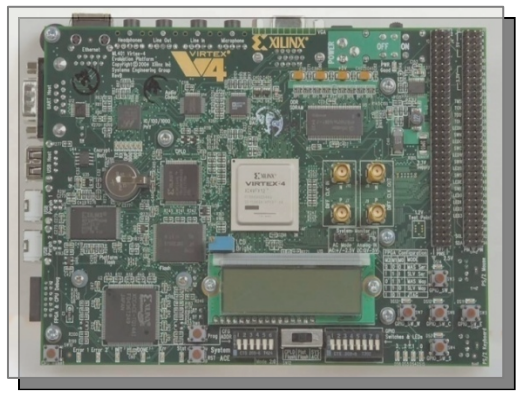


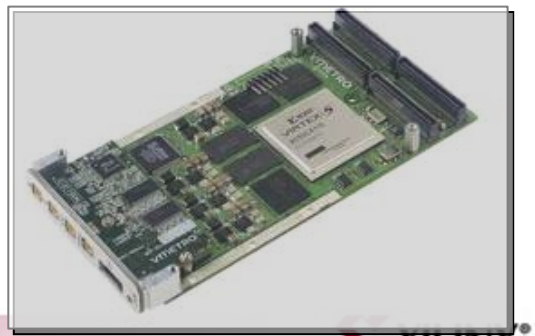
How many types of platforms are there?

- UNLIMITED!
  - For embedded computing
  - For desktop prototyping
  - For high performance computing
  - Customized for specific applications

Existing Development Boards







Appendix 2: In-flight SDR

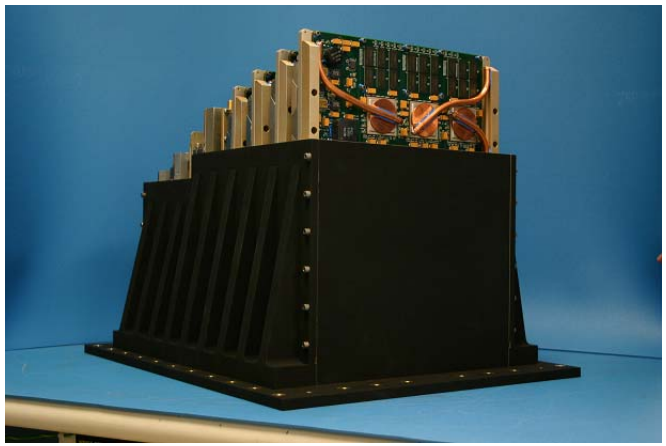
- Slides Courtesy of SEAKR Engineering
- Uses SRAM FPGAs

Flight Hardware Now

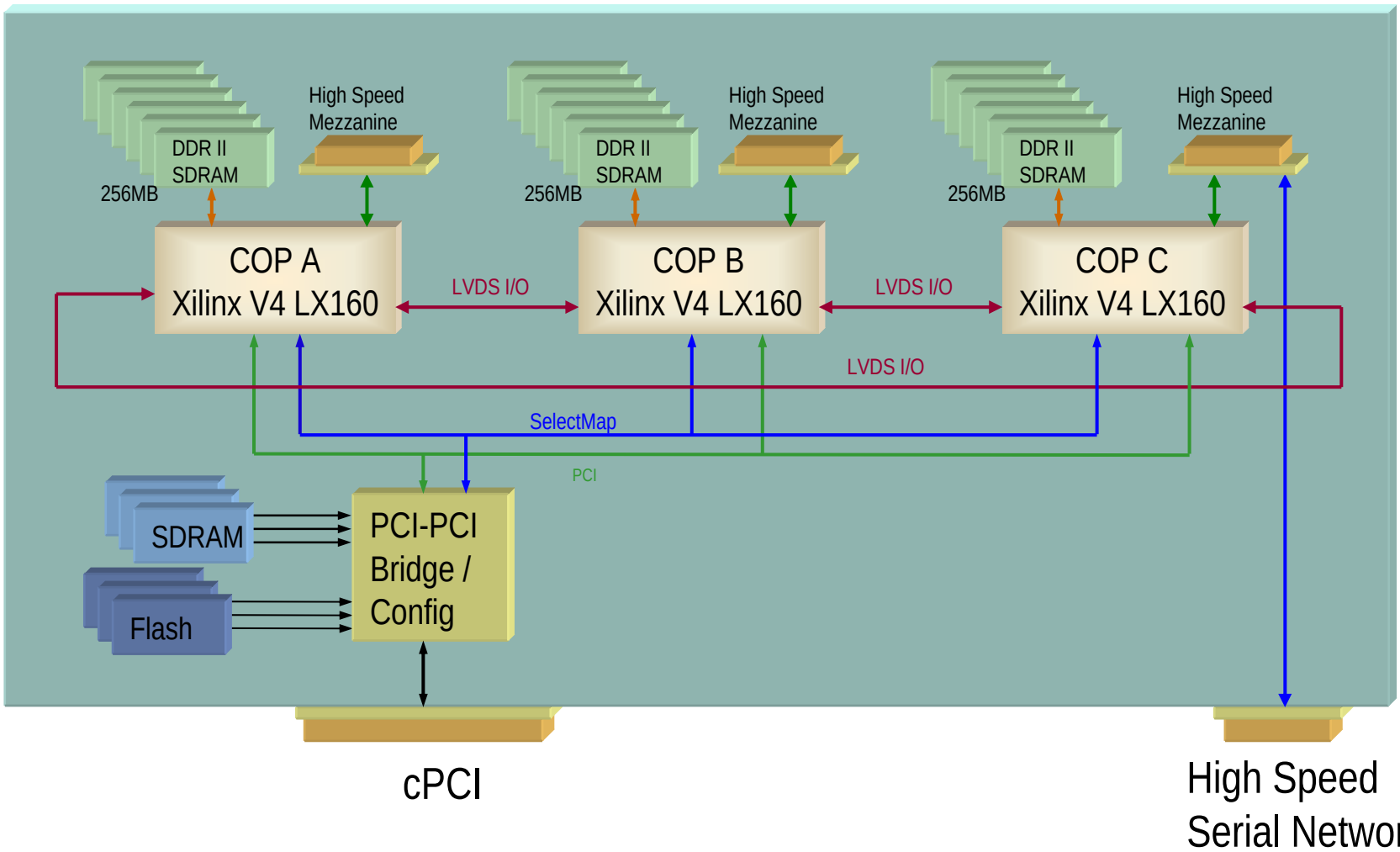
Virtex 4 ReConfigurable Computer for Space

3000MIP SBC for Space

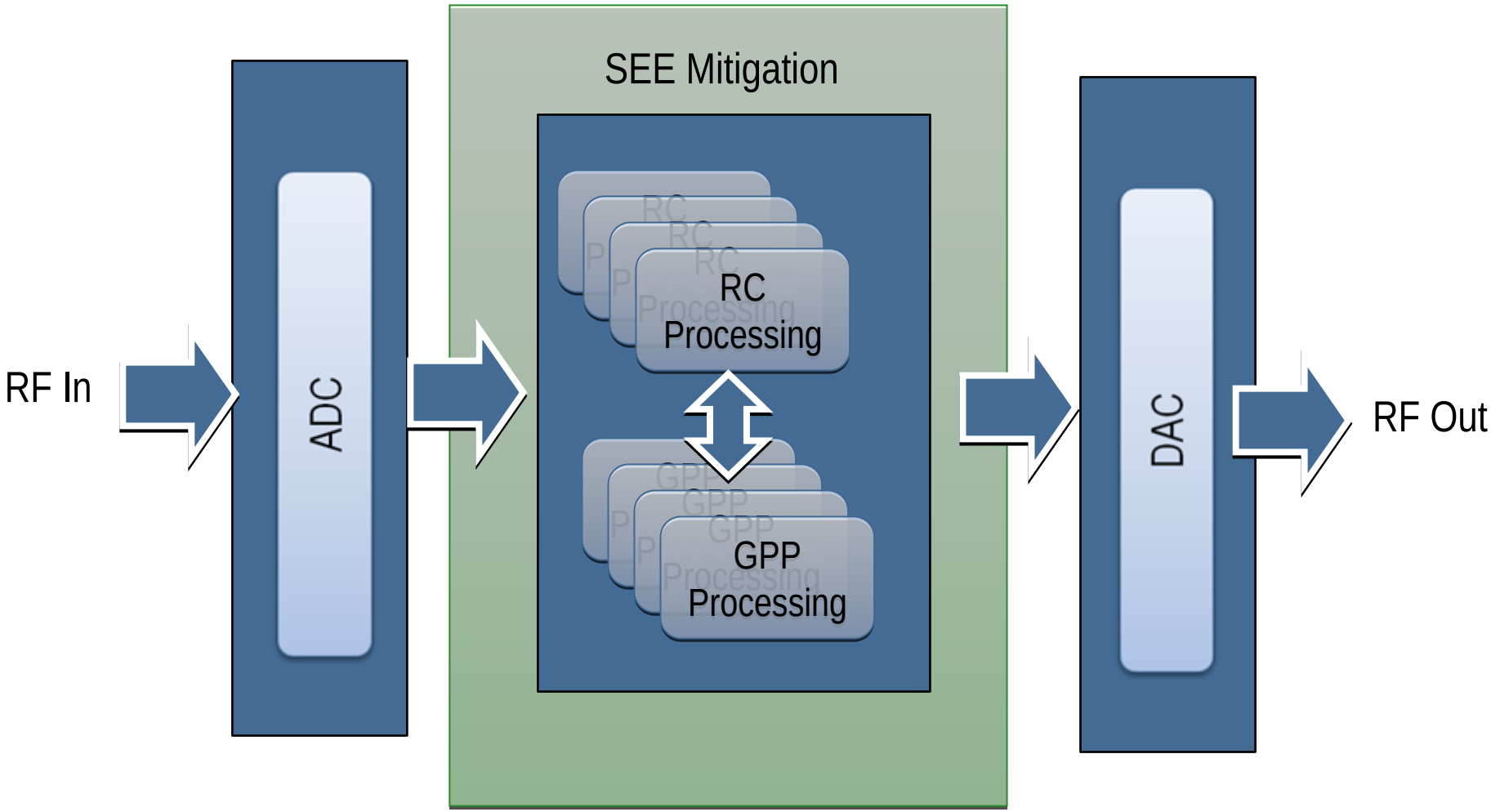
Flight Hardware Now



RA-RCC Block Diagram

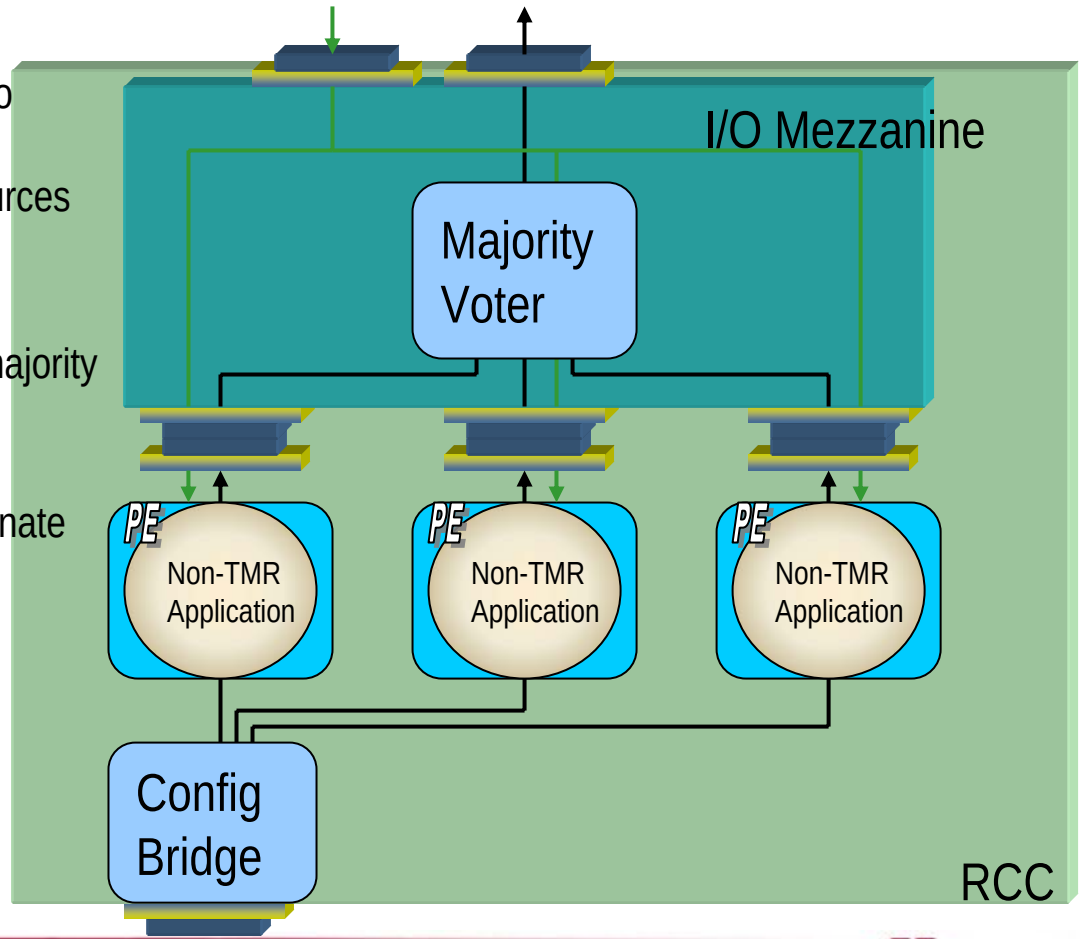


High Level Block Diagram



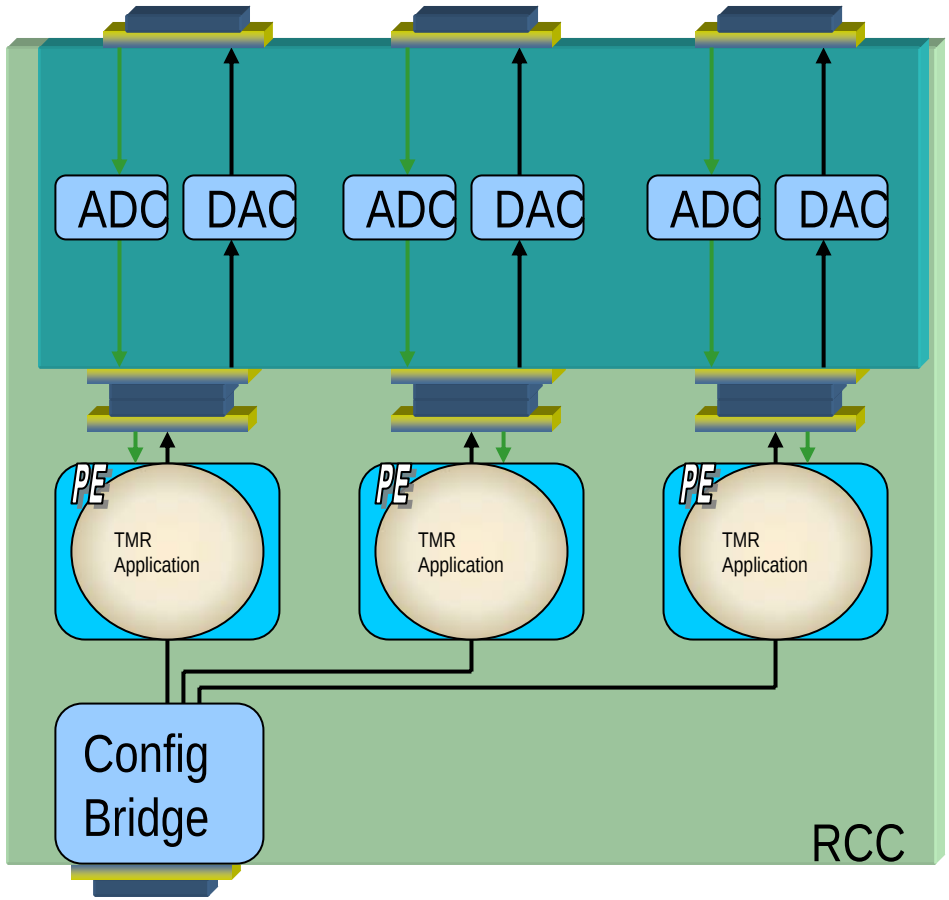
SEE Mitigation, External TMR

- Pros:
  - SEFI Immune
  - SEU Immune (or very low)
  - “Brute force” approach, easy to implement at the FPGA level
  - Protection for all internal resources (macos, dcms, I/O)
- Cons:
  - Requires external device for majority voting
  - Large SWAP increase
  - Requires design mods to eliminate persistence and provide resynchronization logic



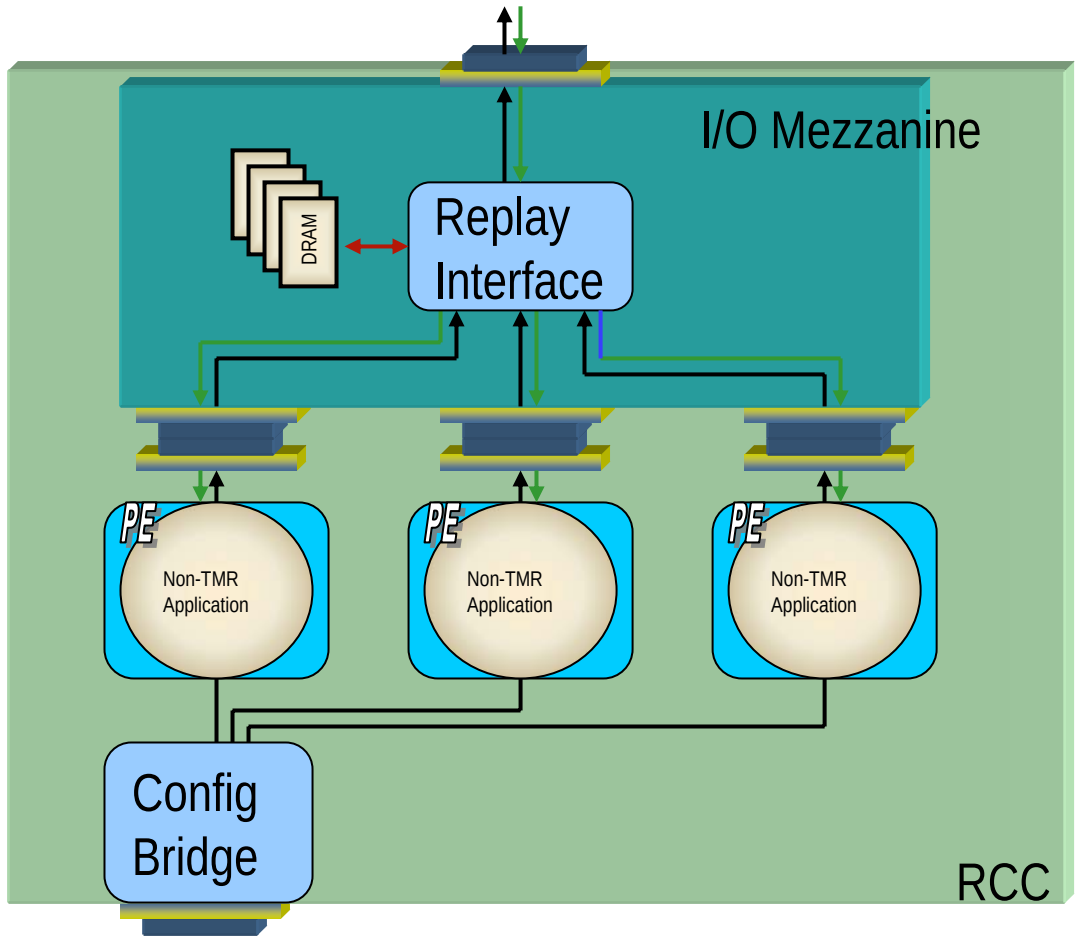
SEE Mitigation, Internal TMR Configuration

- Pros:
  - Fault tolerant design can be implemented in a single FPGA
    - Reduce system SWAP
- Cons:
  - Difficult to implement
  - Design grows 3.4 to 6x
  - Speed decrease 5% to 20%
  - Power increase > 3x
  - Difficult/impossible to TMR some embedded elements (PPC 405, SERDES)
  - Not immune to SEFI's
  - Has susceptible cross-section
  - MBUs may break internal TMR



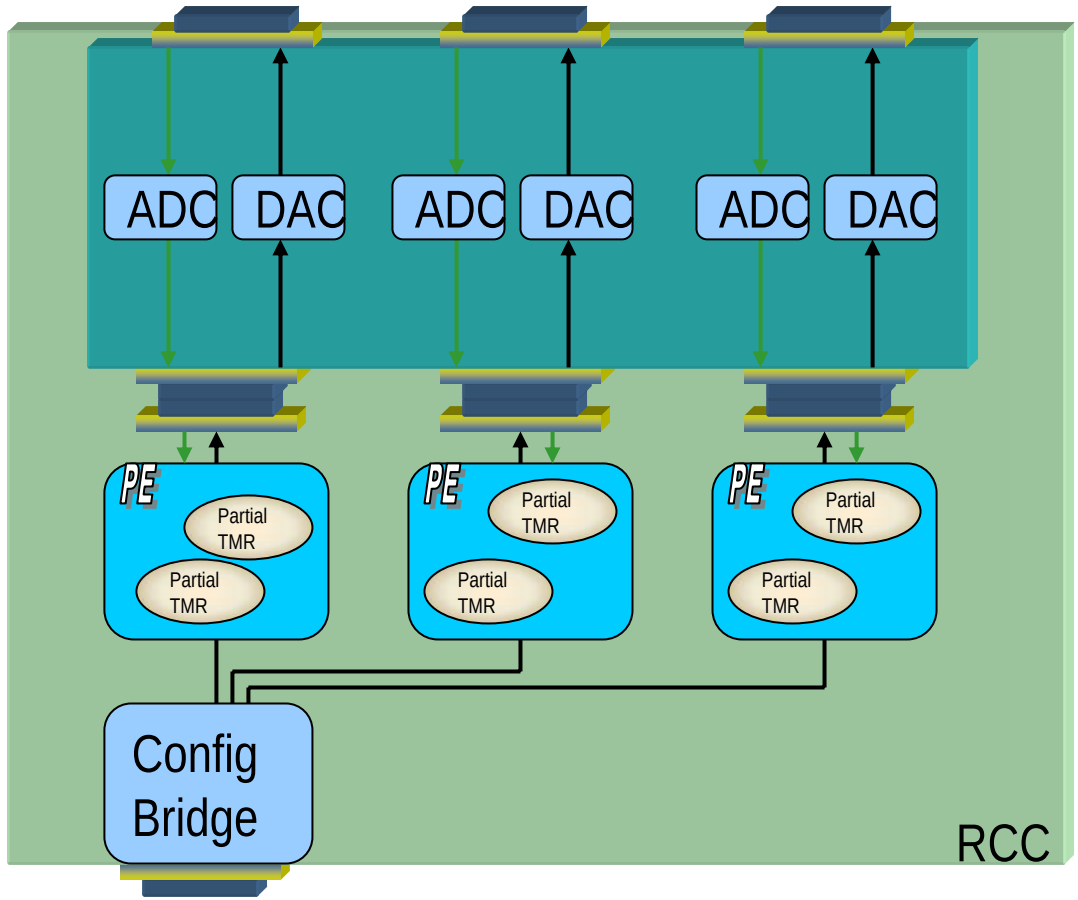
SEE Mitigation, Replay

- Little to no TMR
- Increased performance
- Very Small SEU cross-section
- SEFI immune
- Use commercial development tools
  - System gen, Celoxica, Simulink, commercial cores…
- Network interfaces to front panel or backplane
- Patent #7263631



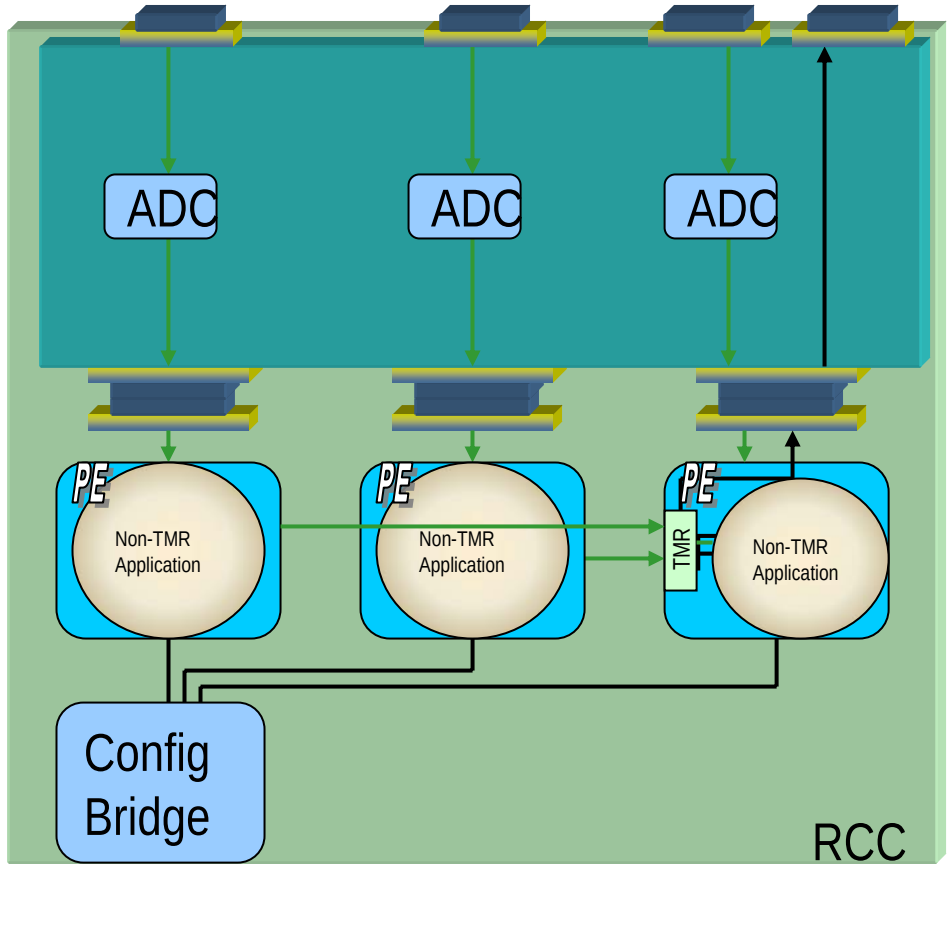
SEE Mitigation, Partial TMR

- Use selective TMR to remove persistence
- Data comes in through PCI or Mezzanine
- High processing performance
- Use commercial development tools
  - System gen, Celoxica, Simulink, commercial cores…
  - Designs require modification to remove persistence

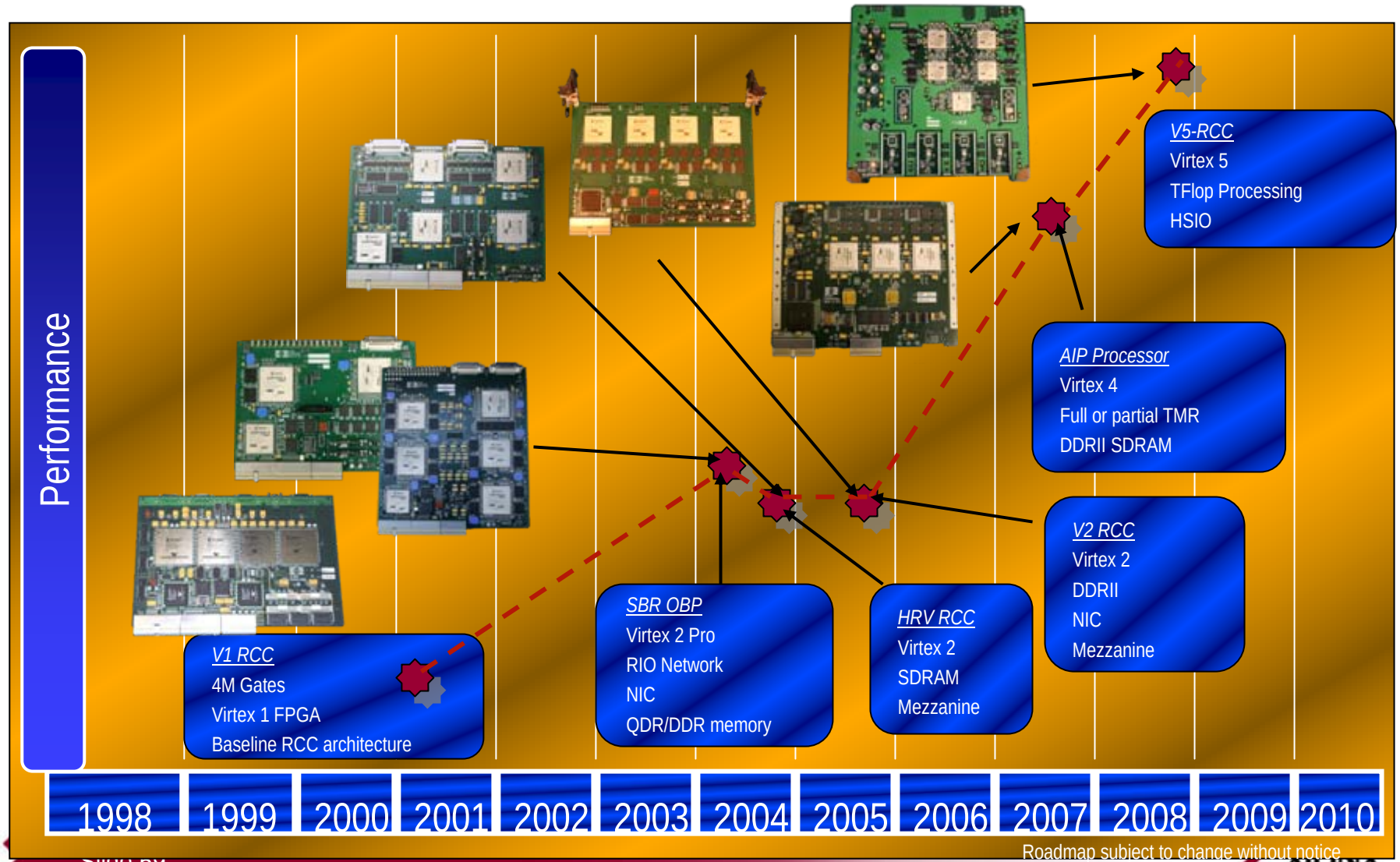


SEE Mitigation, Hybrid TMR Configuration

- Non-TMR applications in all three FPGAs
- Third FPGA has a TMR'd Voter and TMR output
- Very small SEU cross-section



Space RCC Roadmap



Roadmap subject to change without notice

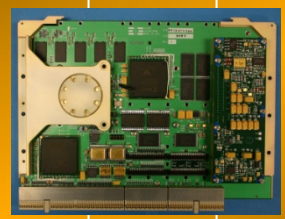
Space SBC Roadmap

Performance

MightySat II
TI Quad C40
250 MFLOP OBP
HSI Cloud Detection

SEAKR 603
PowerPC603e
100 MIPS
VME

SEAKR G4
PowerPC Gen4
800 MIPS
2 GFLOPS
cPCI



Athena Processor
PowerPC Processor
5.3 Gbyte/S memory bandwidth
3000 MIPS
ECC on internal cache

Next Generation
TFlop Processing
Multi GFLOPS/Watt
In development

1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010

Roadmap subject to change without notice