Virtex-4 FX Auxiliary Processor Unit Controller

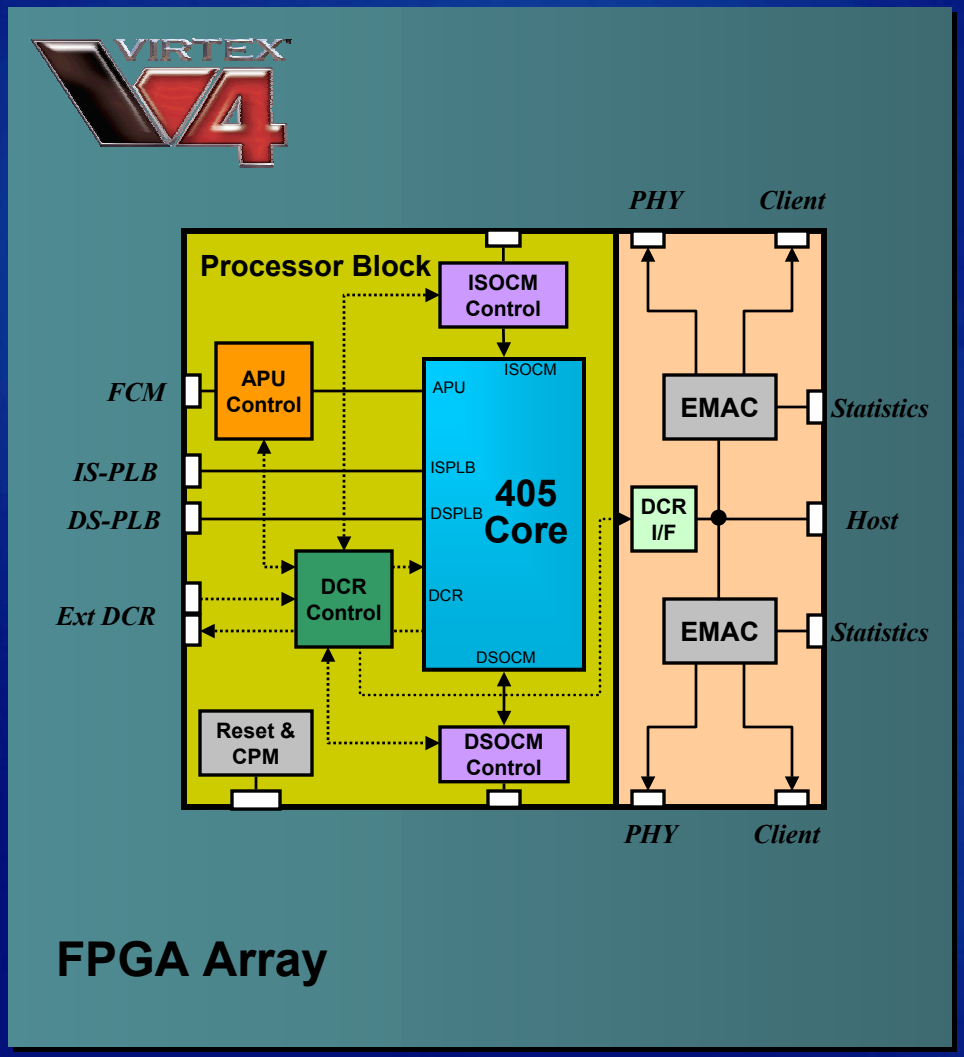
Ahmad R. Ansari

October 2004

Agenda

- Virtex-4 FX Processor Block
- Auxiliary Processing Overview
- APU controller Fundamentals
- Operations and execution classes
- ISA Extension mechanisms
- Application Examples

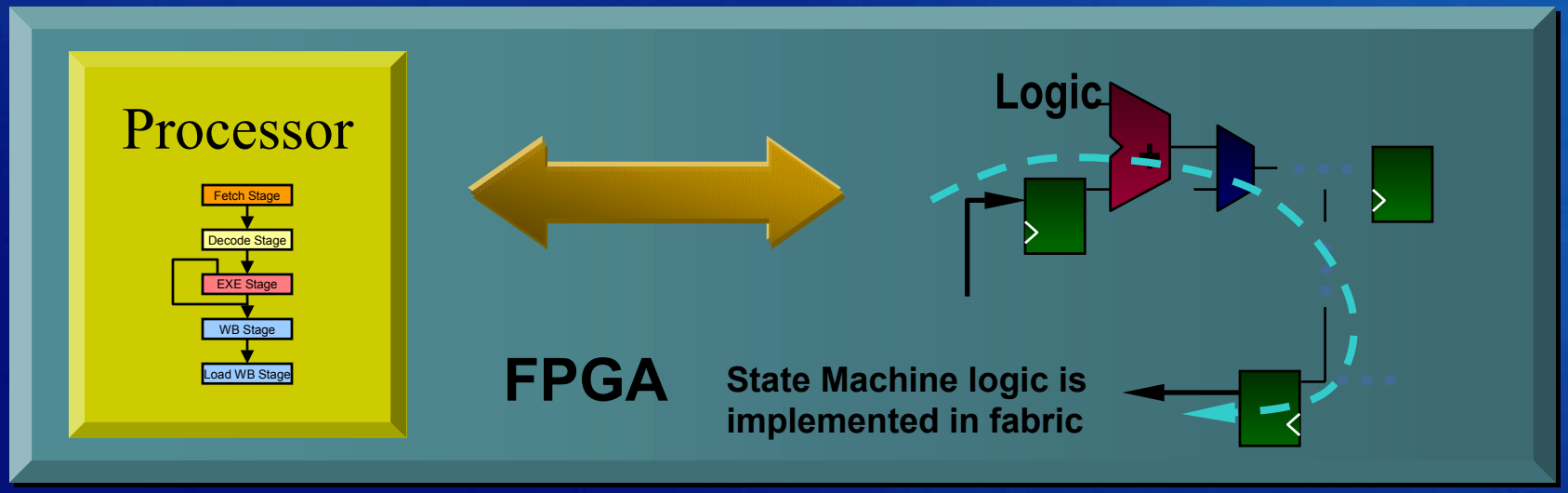
Virtex-4 Processor Block



- PowerPC 405 core
- Separate on-chip instruction and data memory controllers
- Separate high-performance Instruction and data PLB
- Device Control Register Interface
- Two EMAC 10/100/1000 Cores
- Auxiliary Processor Unit (APU) Controller
- Backward compatible with Virtex-II Pro processor

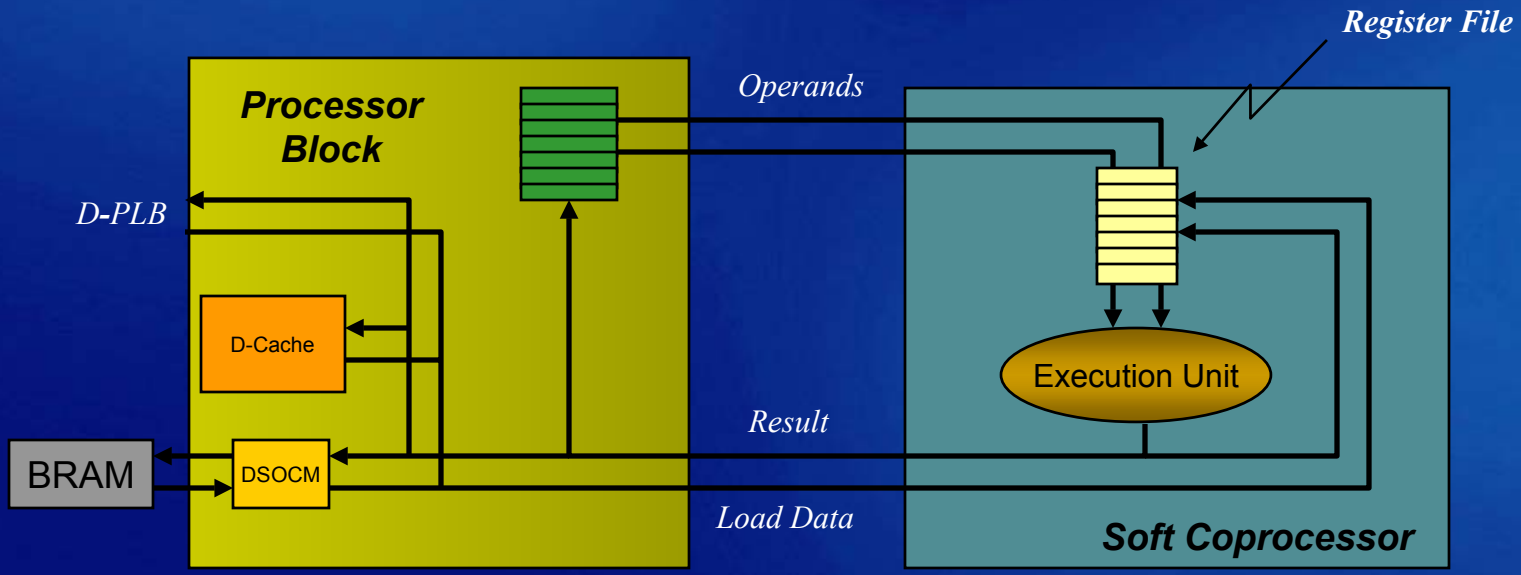
Need for an Auxiliary Processing Interface

- Algorithms are implemented in soft logic
- Applications run on Processor
- Efficient connection between applications running on CPU and algorithms implemented in fabric
  - Low latency
  - Ease of HW/SW integration



Coprocessor Model

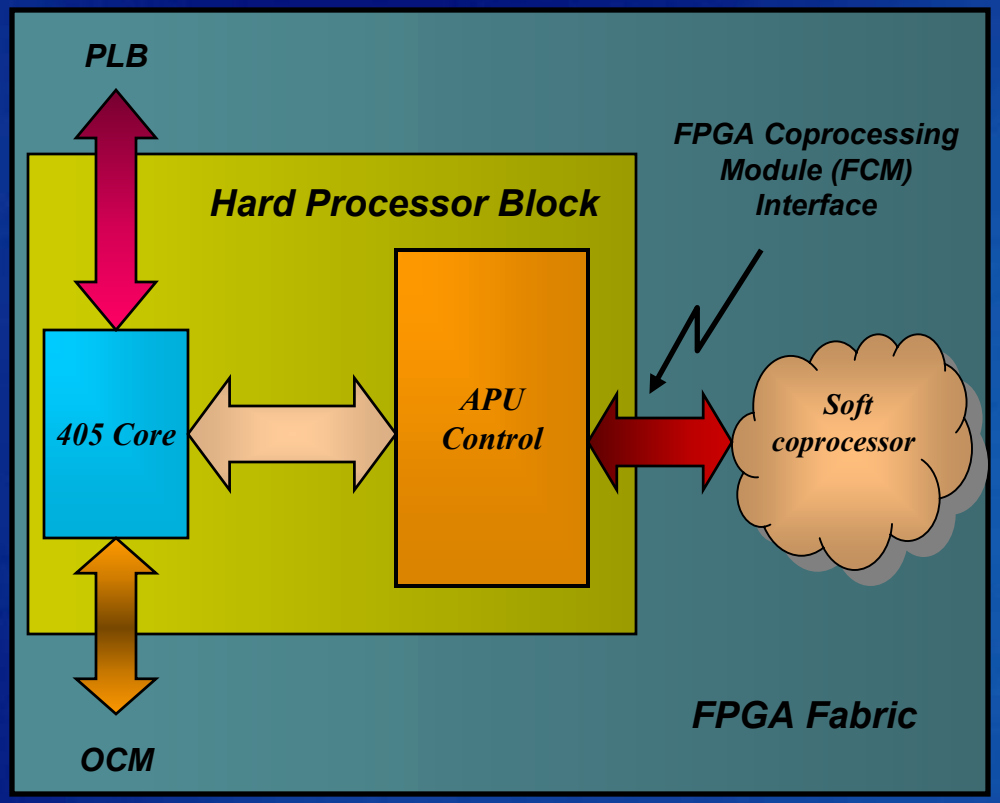
- Execution Units are implemented in the FPGA
- Operands and Results can be passed between the CPU and Coprocessor
- Coprocessor may contain local register file to improve performance
- Data can be communicated between processor storage and Coprocessor



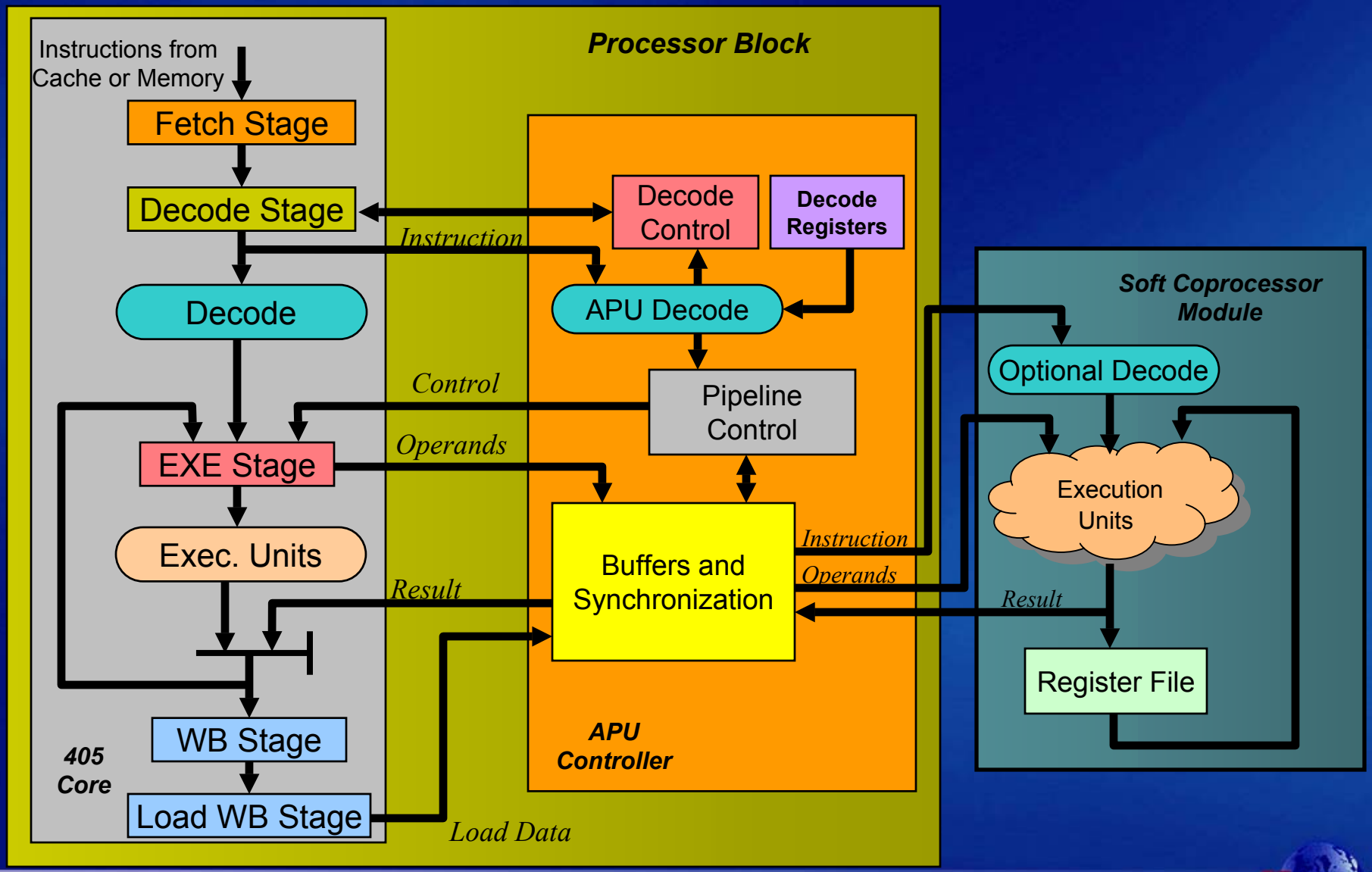
Example: Floating-Point Unit

APU Controller

- Allows direct connection of soft coprocessors in fabric to PPC405. The coprocessor can be:
  - PowerPC FPU
  - User-defined which can be programmed
    - Dynamically: Through PowerPC DCR Instructions
    - Statically: Through FPGA Bitstream
- The fabric processor and the CPU can run at different clock rates
- Opcodes are decoded either by the controller or by the fabric processor
- Autonomous or synchronized execution with the CPU
- High-bandwidth FPGA Coprocessing Module (FCM) interface between processor and fabric

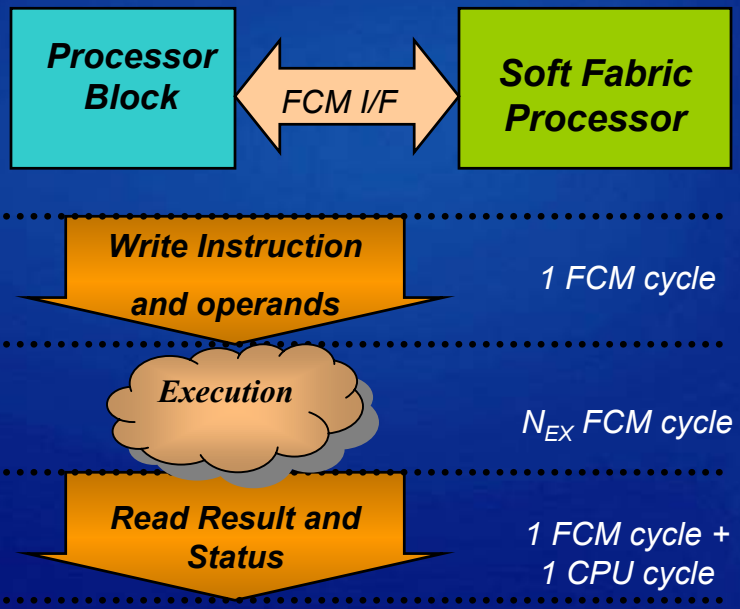


APU Controller Operation



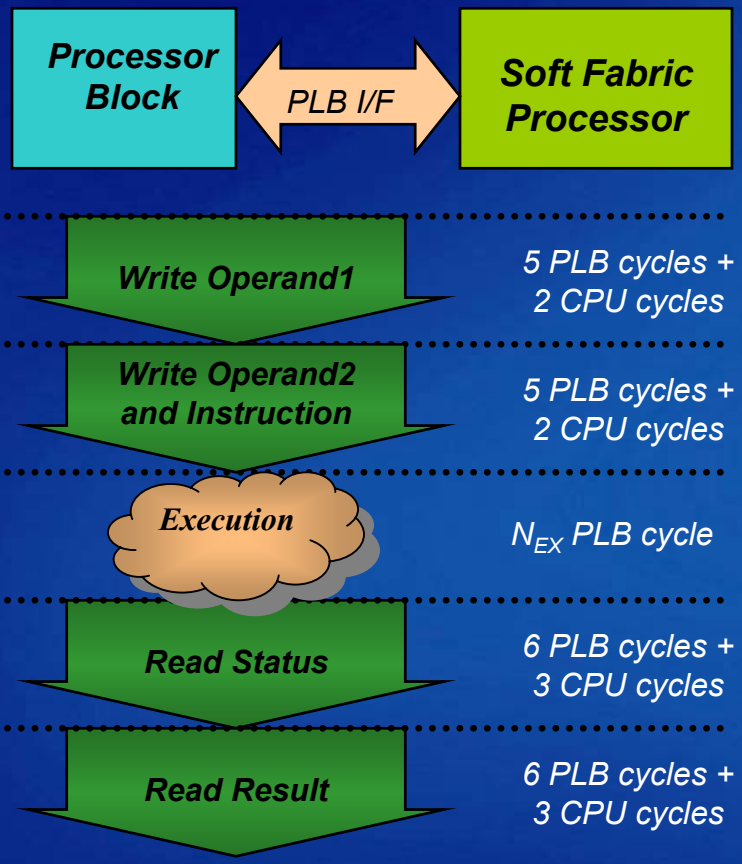
Comparison with a Current Alternative

Using FPGA Co-processing Module (FCM) Interface



(If CPU/FCM ratio = 2) Total = N_{EX} + 2.5 FCM cycles

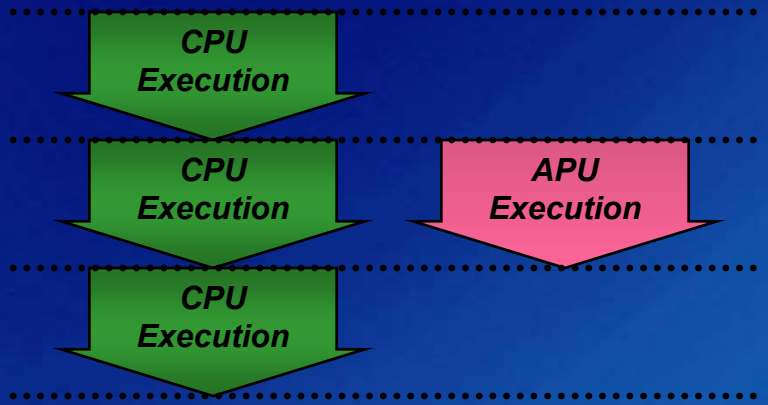
Using CoreConnect Processor Local Bus (PLB)



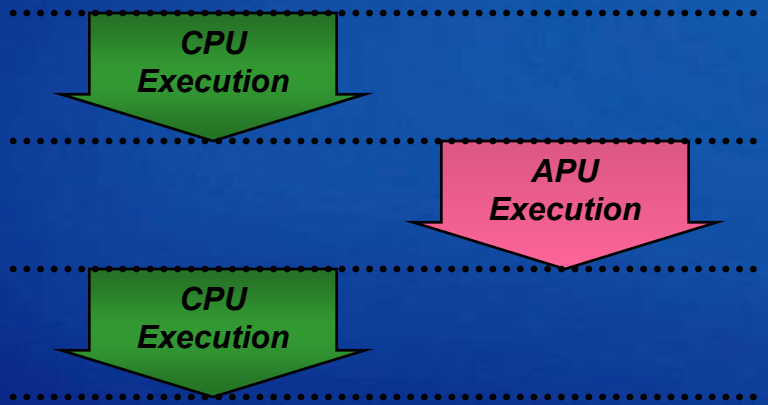
(If CPU/PLB ratio = 2) Total = N_{EX} + 27 PLB cycles

Instruction Execution Classes

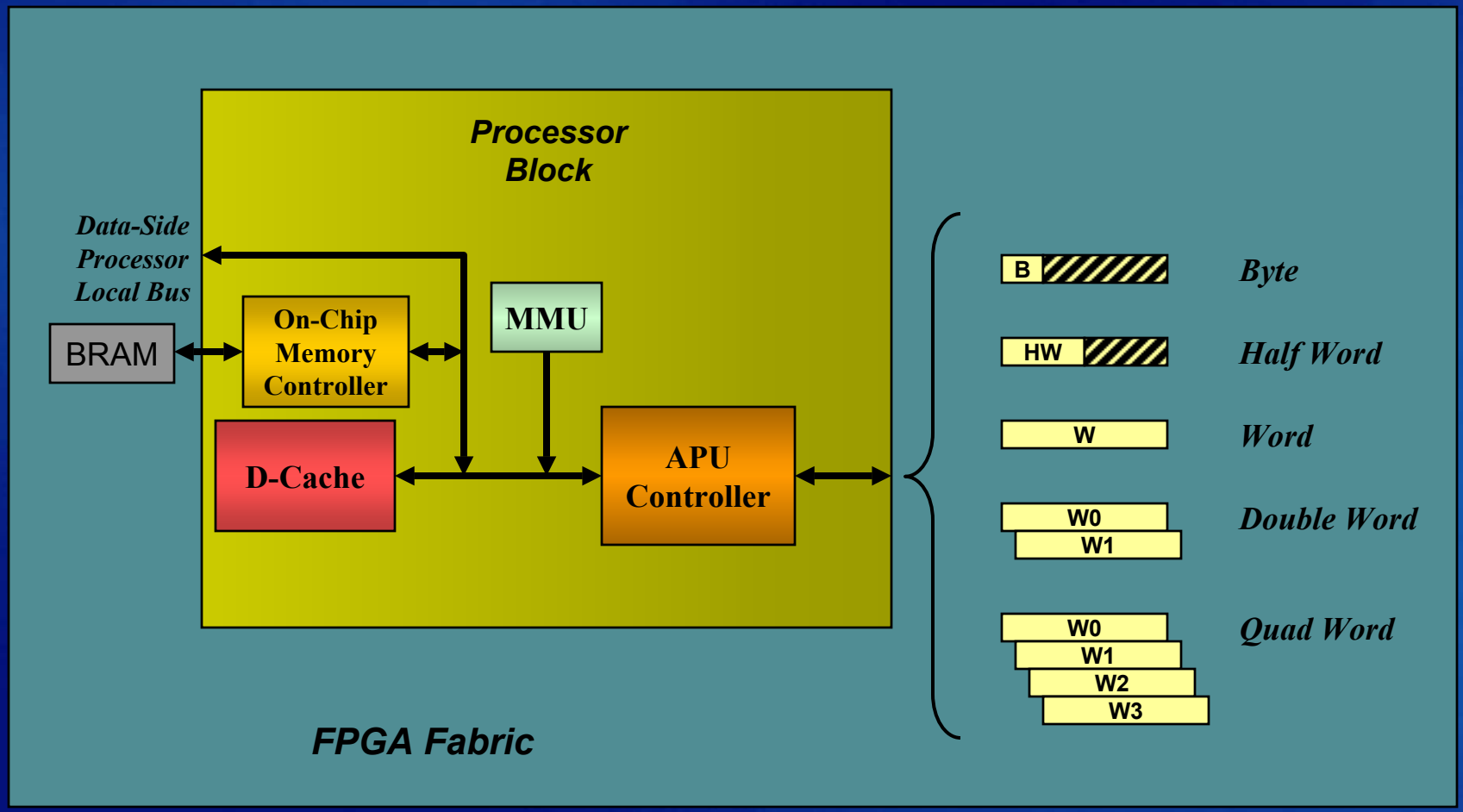
- **Autonomous Instructions**
  - These instructions do not stall issue of instructions in the CPU
    - Instructions that do not return a value to CPU GPR



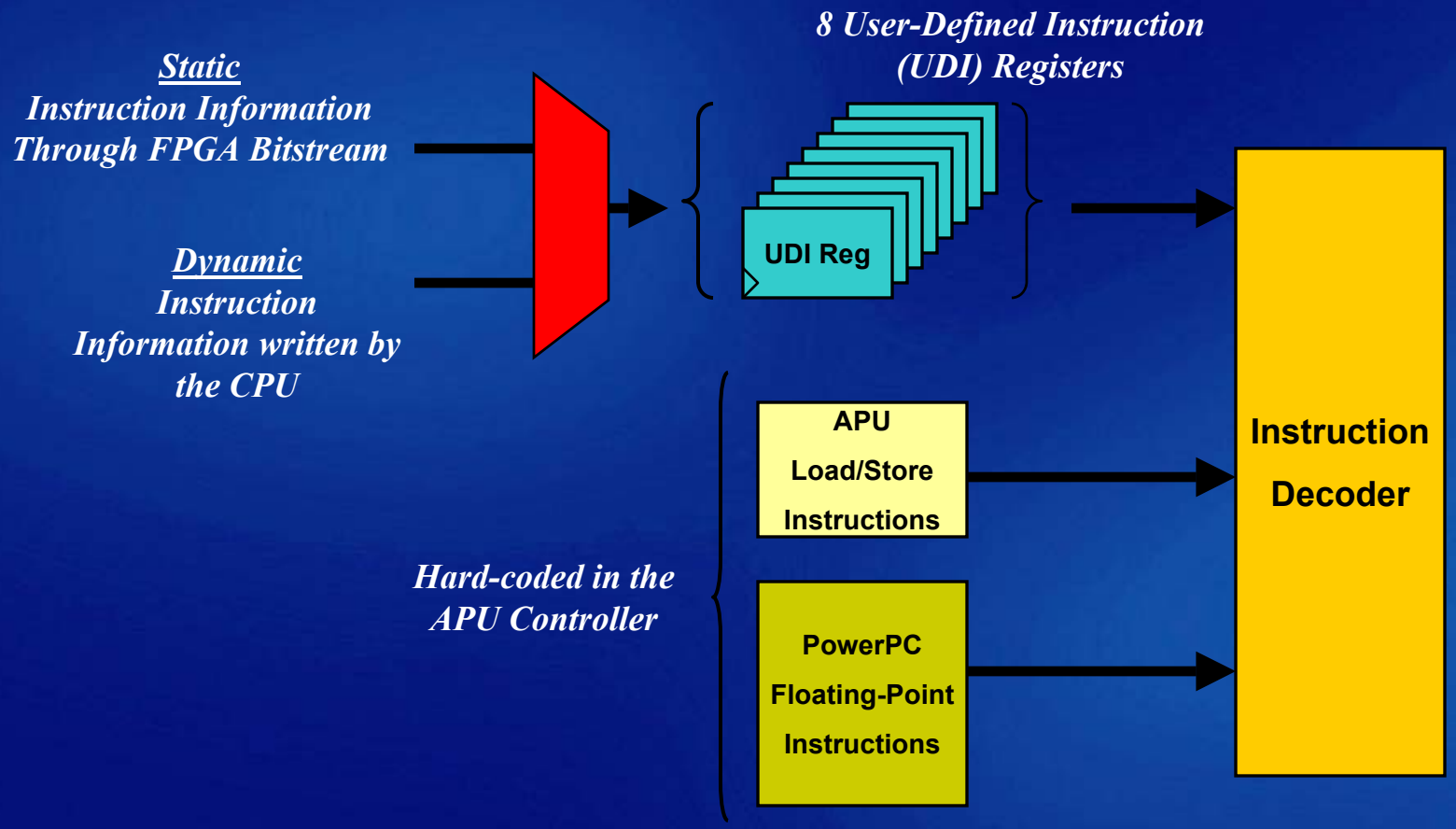
- **Non-autonomous Instructions**
  - These instructions stall issue of instructions in the CPU
    - Instructions that return a value to CPU GPR
      - Blocking
      - Non-Blocking



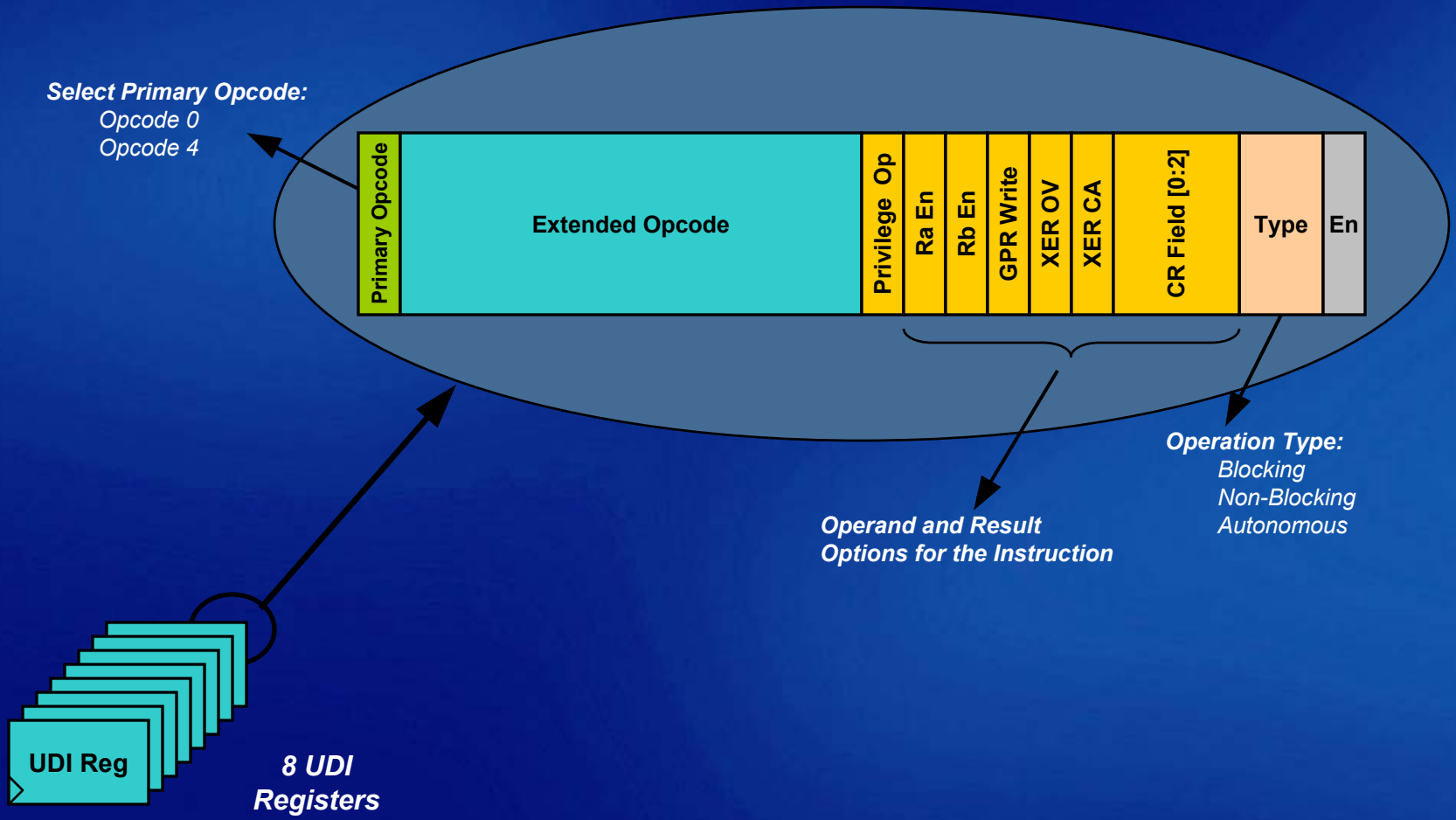
APU Load/Store Operations



APU Decode Interface



User-Defined Instruction (UDI) Registers



User-Defined Instruction Opcodes

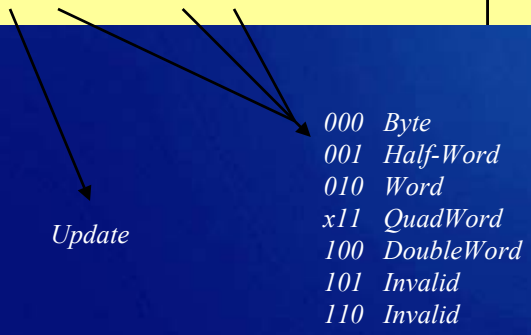
A) All opcode 0 except Ext.Op = 0

B) All opcode 4 except Ext.OP = xxx-xxx1-xx0x.
Following subset is reserved for Instructions that modify the condition register:

Primary	Extended	Explanation
00 0100	1xx xx00 0110	Any APU instruction that needs to set a CR field as part or all of the result of the instruction

C) APU Load/Store Operations:

Primary	Extended	Explanation
01 1111	U W_0 L/S W_1 W_2 0 0 1 1 1	APU Load/Store Instructions

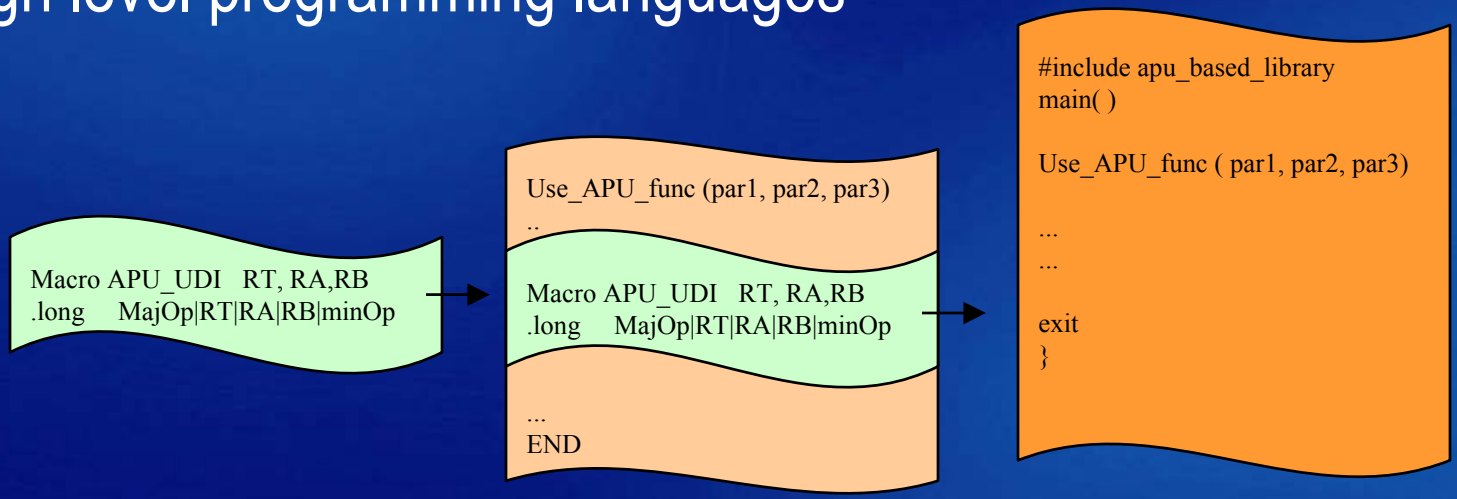


APU Controller Benefits

- Extends instruction set of the 405 Core
  - Several instructions can be replaced with one instruction
- Offloads compute-intensive operations
  - Floating point mathematics
  - Multimedia Processing: PowerPC VMX (AKA AltiVec)
  - Complex DSP cores such as FFT
- Direct and controllable coupling between CPU pipeline and fabric
  - Interface signals control the flow of an instruction in the CPU and fabric pipelines
  - Otherwise software constructs (semaphore, etc.) are needed
- Wide data paths into and out of the CPU
  - In 1 FPGA cycle, 32 bits of instruction and 64 bits of data is sent to fabric. Next cycle 32 bit of data and various status bits can be read in parallel
  - Byte, half-word, word, double-word and quad-word transfers between the cache or processor storage and the FPGA fabric. (APU load/store)

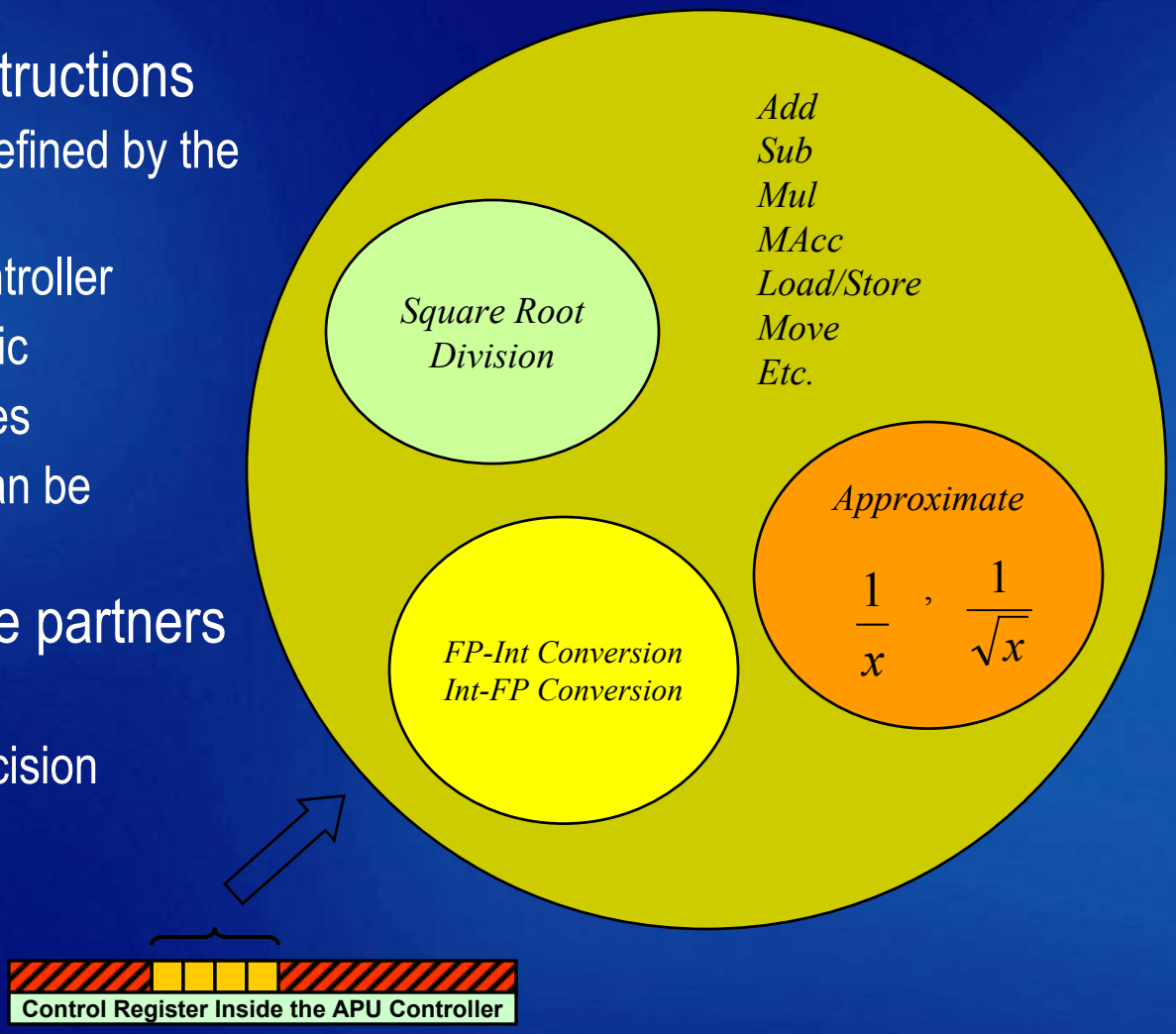
Usage of UDI Instructions in Software

- User-Defined Instructions (UDI) in APU controller can be used via SW macros
- Library functions calling the macros make the UDI available in high level programming languages



Virtex-4 Floating-Point Support

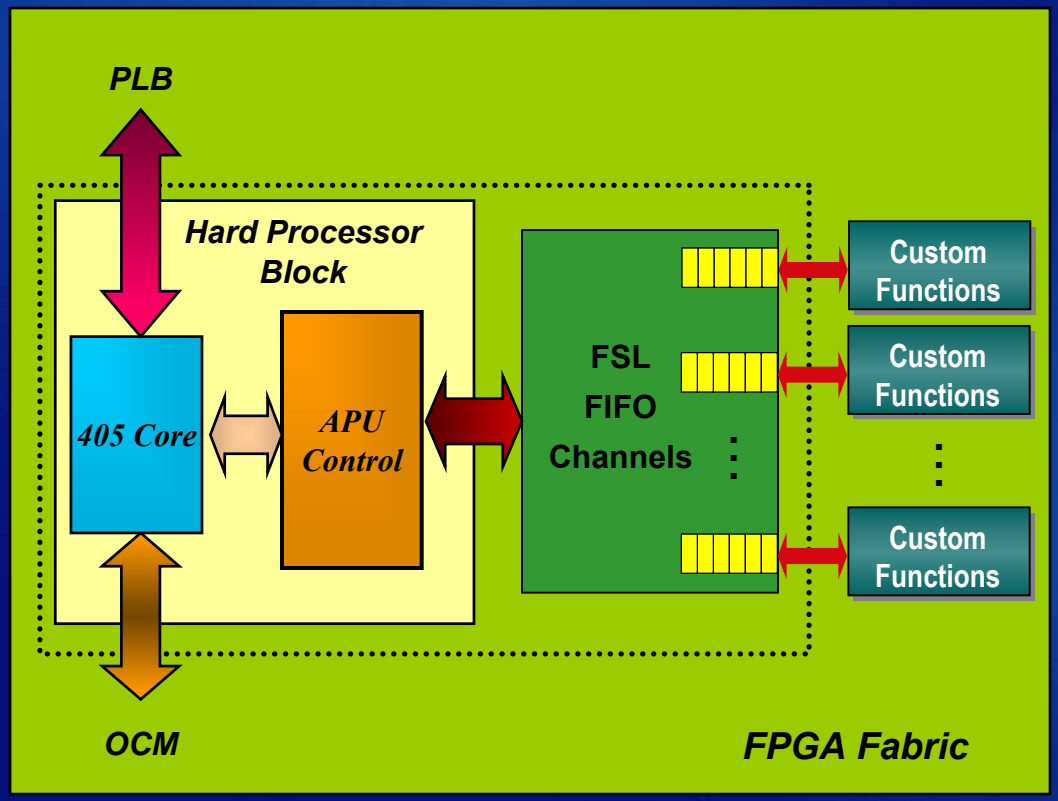
- PPC Floating-Point Instructions
  - Opcodes are already defined by the Architecture
  - Interpreted by APU controller
  - Executed in FPGA fabric
  - Grouped into subclasses
  - Disabled subclasses can be emulated by the CPU
- Soft cores from Alliance partners
  - IEEE-754 Compliant
  - Single and Double Precision
- Performance Target
  - 250 MFlops Peak



Depending on the complexity of the Soft IP, subgroups can be selected Independently

Processor Feature Extension through Fast Simplex Links (FSL)

- FSLs are unidirectional point-to-point data streaming interfaces
- Used to communicate both Control and Data

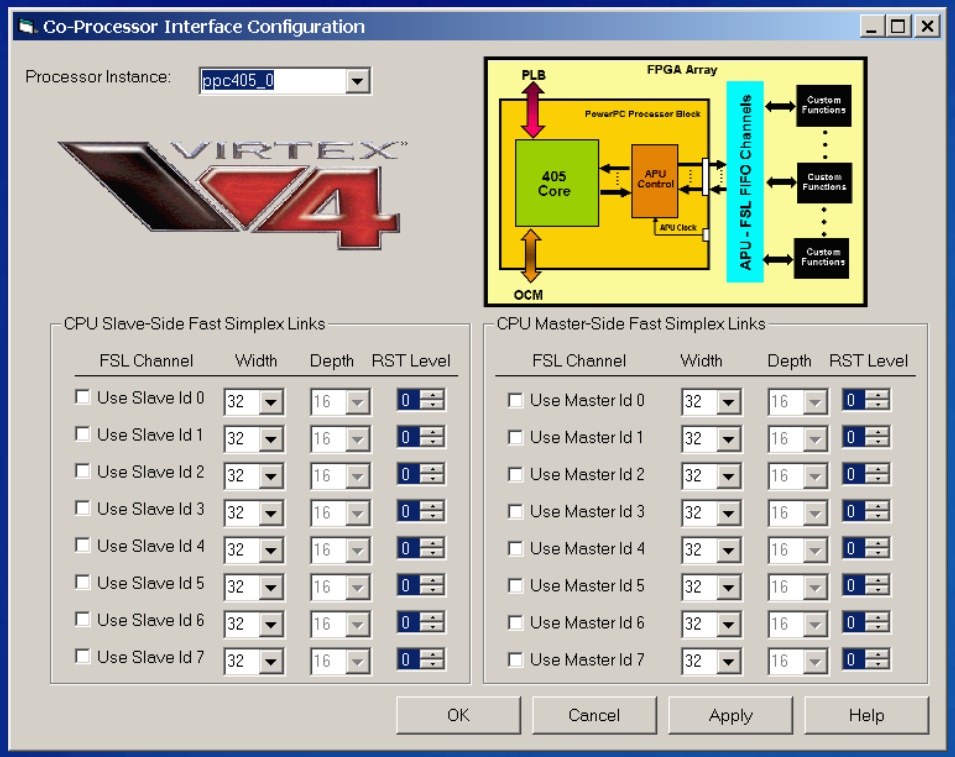


FSL Instructions:

get R_T, FSLx
get data from FSL x (blocking)
nget R_T, FSLx
get data from FSL x (non-blocking)
cgetR_T, FSLx
get control from FSL x (blocking)
ncget R_T, FSLx
get control from FSL x (non-blocking)
put R_A, FSLx
put data to FSL x (blocking)
nput R_A, FSLx
put data to FSL x (non-blocking)
cput R_A, FSLx
put control to FSL x (blocking)
ncput R_A, FSLx
put control to FSL x (non-blocking)

FSL-based Co-Processor Configuration

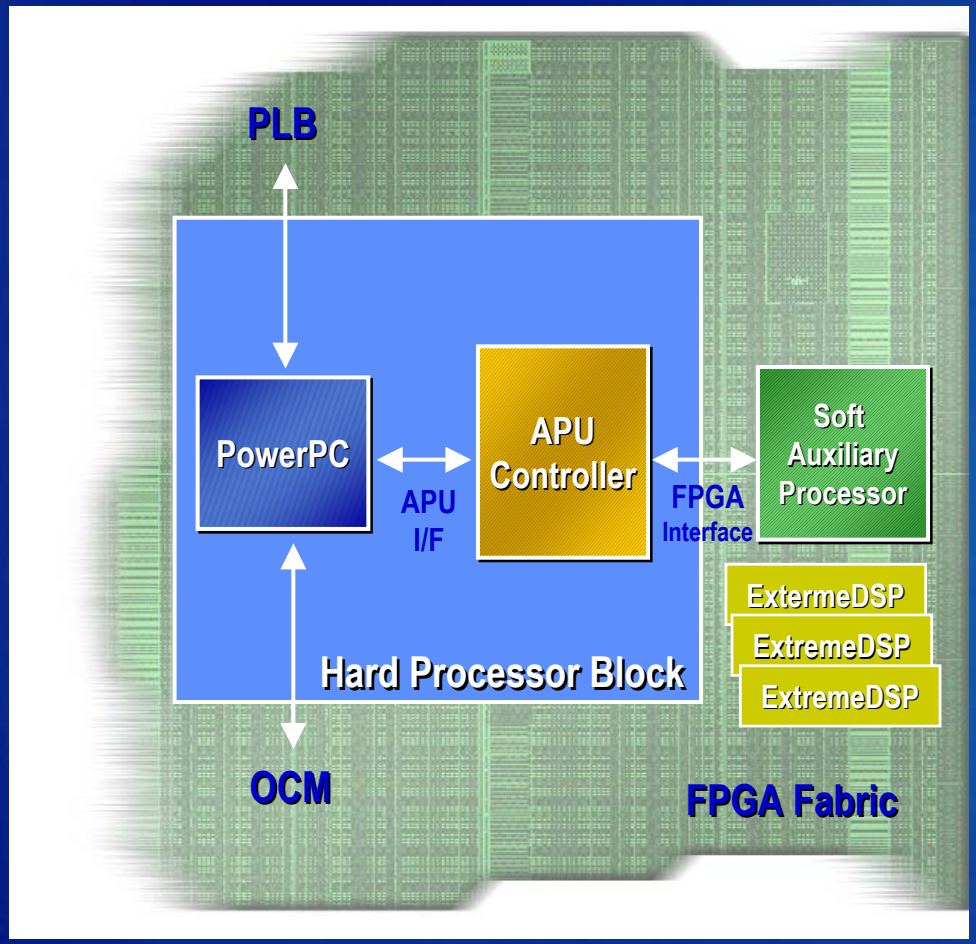
- Utilizes Fast Simplex Link (FSL) FIFO interfaces
  - Available with MicroBlaze™ since EDK 3.2i
- Platform Studio GUI assists users to import the coprocessor into the system
- "C"-based software functions
- Libraries are portable between PowerPC and MicroBlaze™ Soft Core
- Will be Integrated with Xilinx XtremeDSP technology



APU Application Example:

Video Application: Accelerated IDCT MPEG Decompression

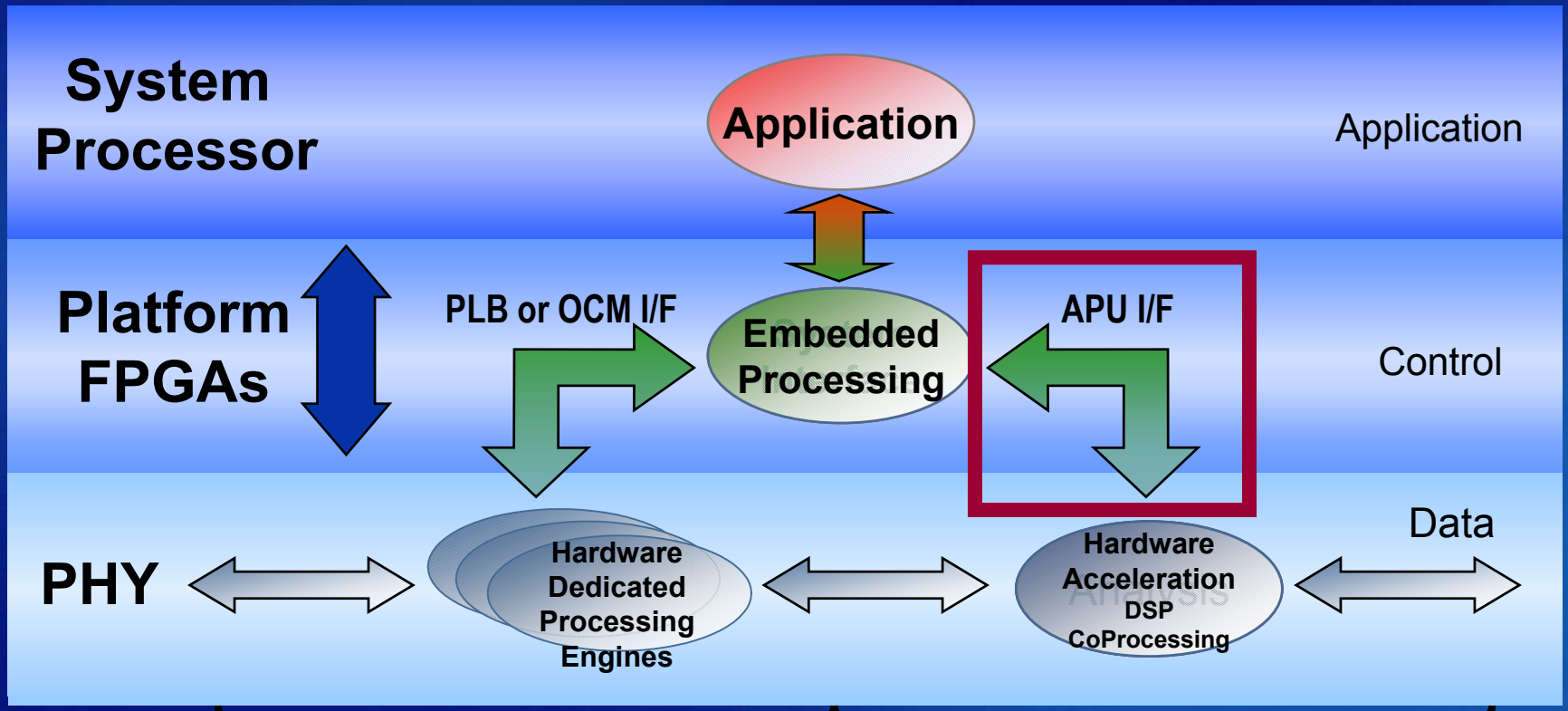
- Leverages Virtex-4 Integrated Features
  - PowerPC, APU, ExtremeDSP
  - Software Emulation – 13000 IDCTs/sec
- HW acceleration over software
  - Lower latency and high bandwidth
- Efficient HW/SW design partitioning
  - Optimized implementation
- Significant performance increase
  - 20x improvement over software
  - Utilizing APU and ExtremeDSP



APU Application Example:
Wireless Applications

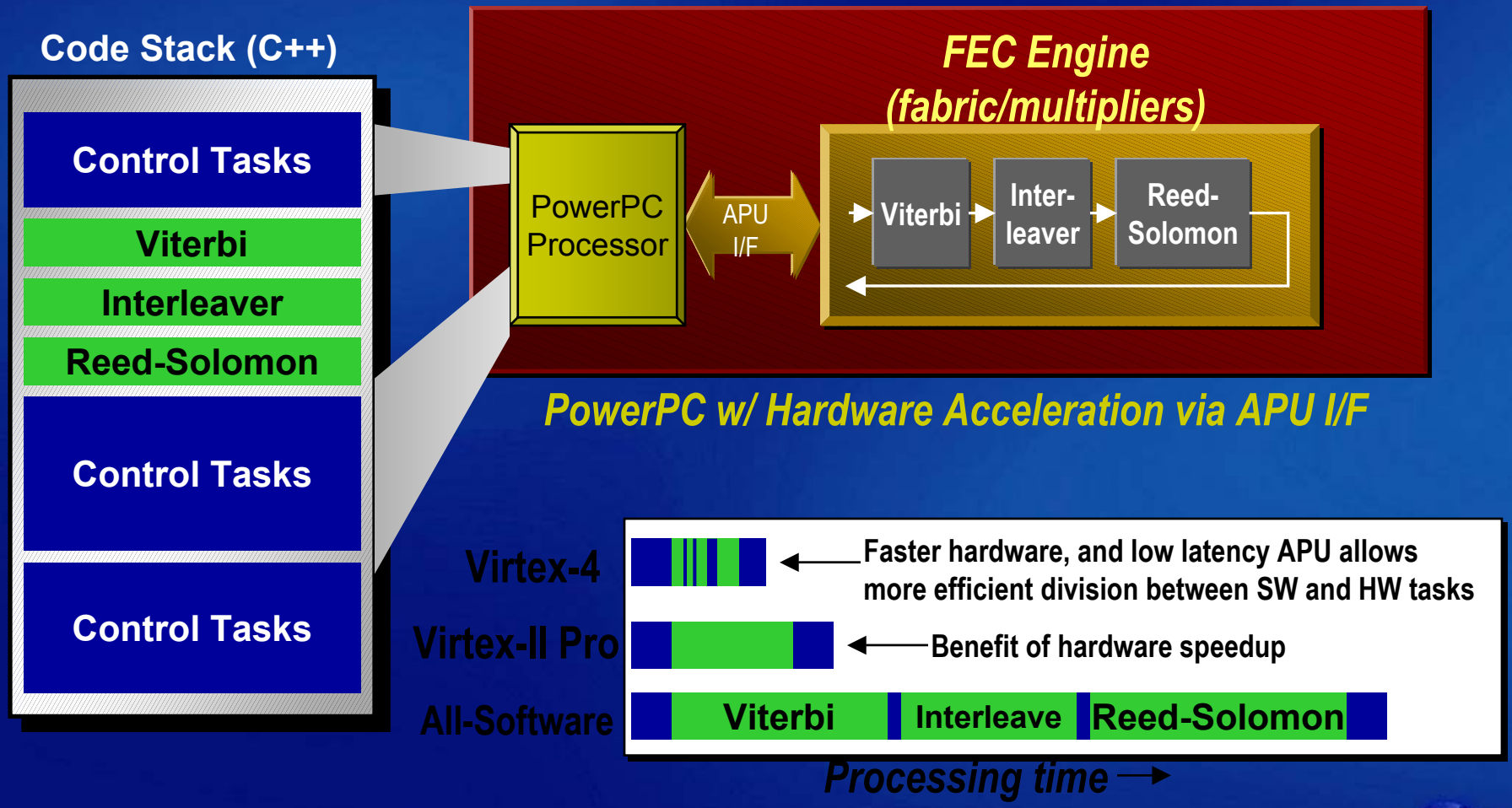
- HW/SW Partitioning a complex system base-station is difficult
  - Partitioning boundaries may not scale as extra channels are added
  - Extra interrupts / scheduling requirements tend to scale non-linearly
- APU simplifies handoff of data between SW/HW processes
  - Enables HW Acceleration ease of integration
  - Easier to change a process from SW to HW and vice-versa
  - Provide path to access and integrate high-speed functions into a software design methodology

High-Level Partitioning



APU Interface simplifies handoff of data between SW/HW processes

Xtreme Forward-Error Correction Benefits in Virtex-4



Summary

- Performance of algorithms have significantly benefited from FPGAs
- Integration of Processor in the FPGA demands a tighter interface between the algorithm and the application
- APU controller provides this efficient interface
- Extends the instruction set of the processor to fit the algorithm