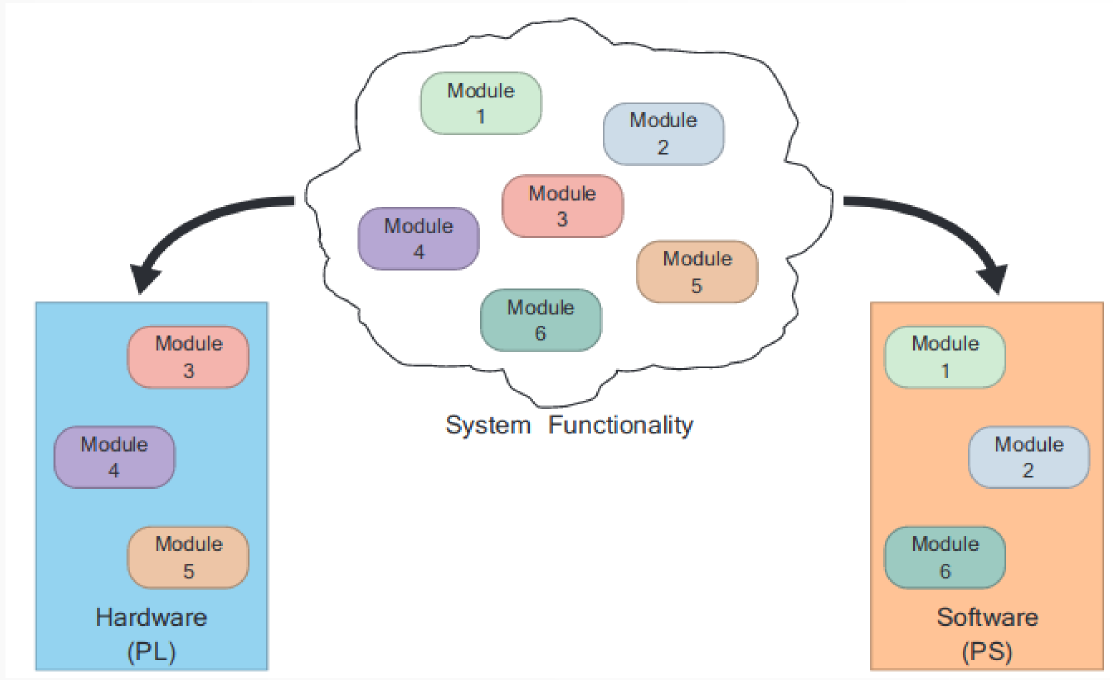
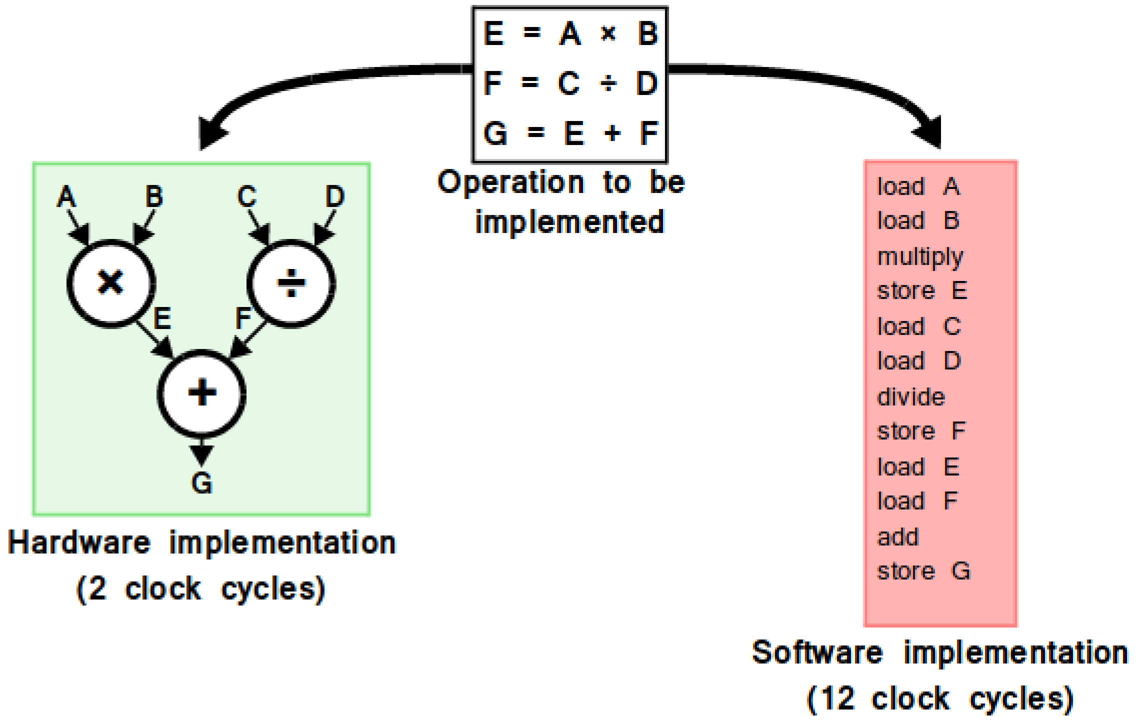
Course 01-2: Zynq architecture overview

Bilel CHERIF
5A-SIEC

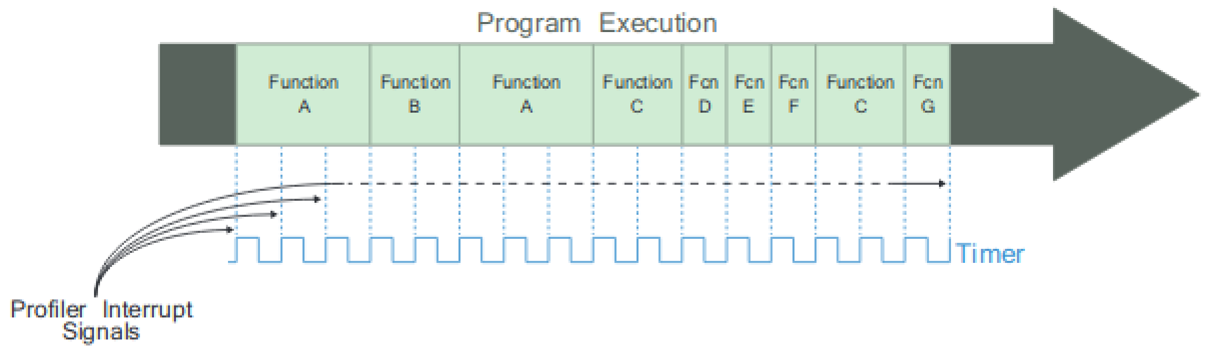
Software/Hardware partitioning



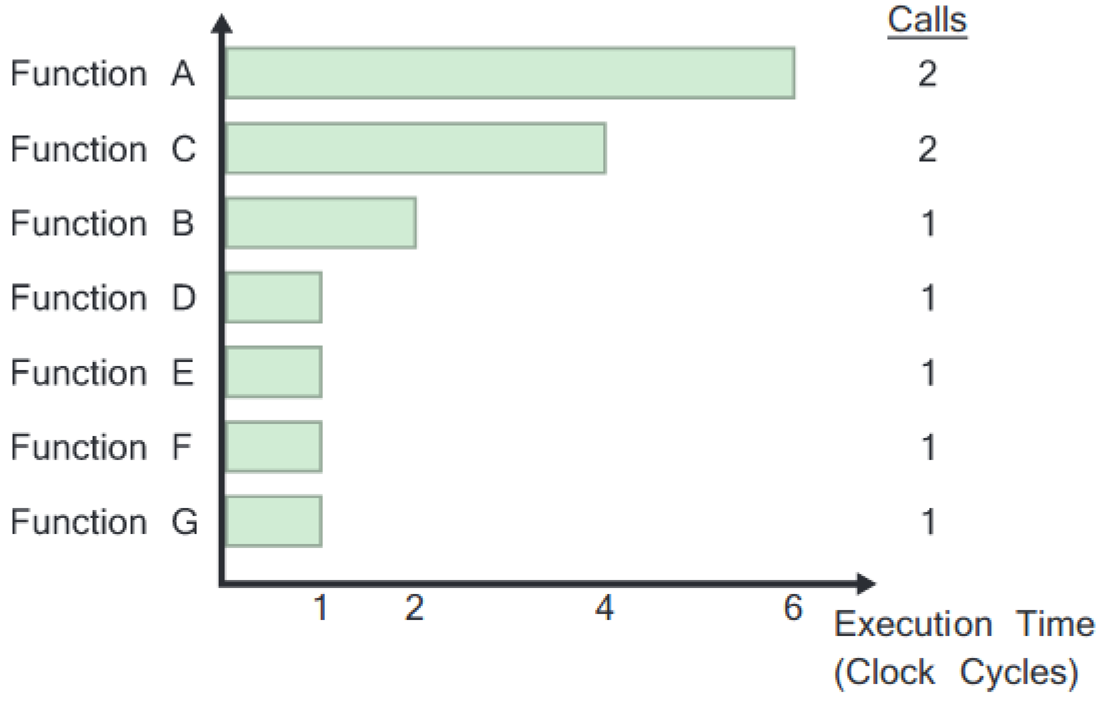


Profiling

Profiling is a form of program analysis that is used to aid the optimization of a software application. It is used to measure a number of properties of application code, including:

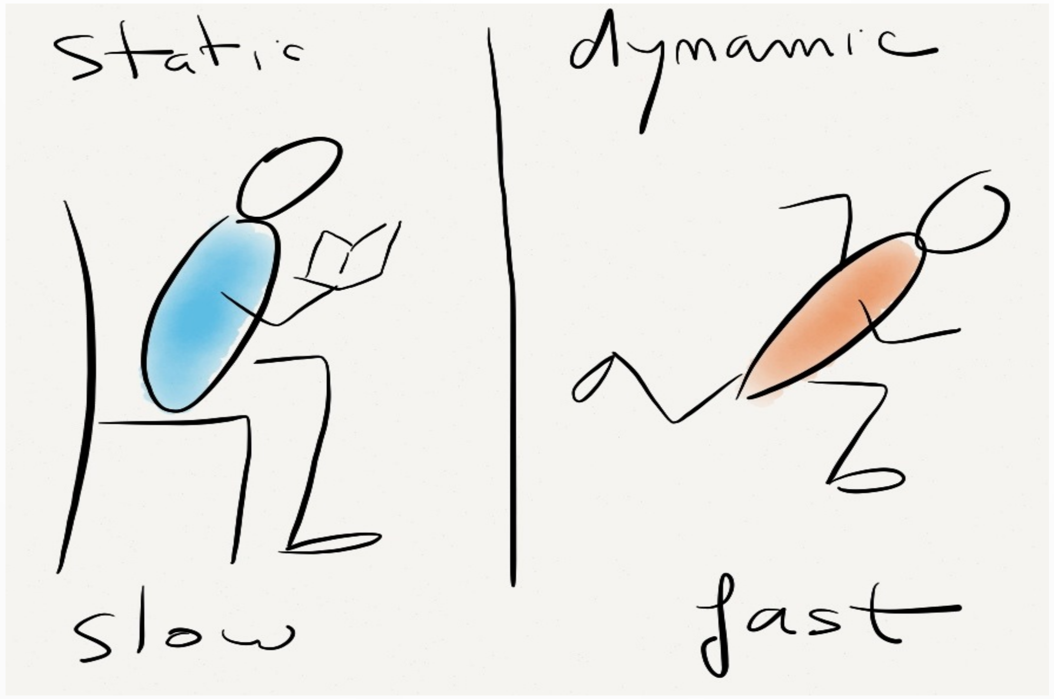


- Memory usage
- Execution time of function calls
- Frequency of function calls
- Instruction usage



Profiling

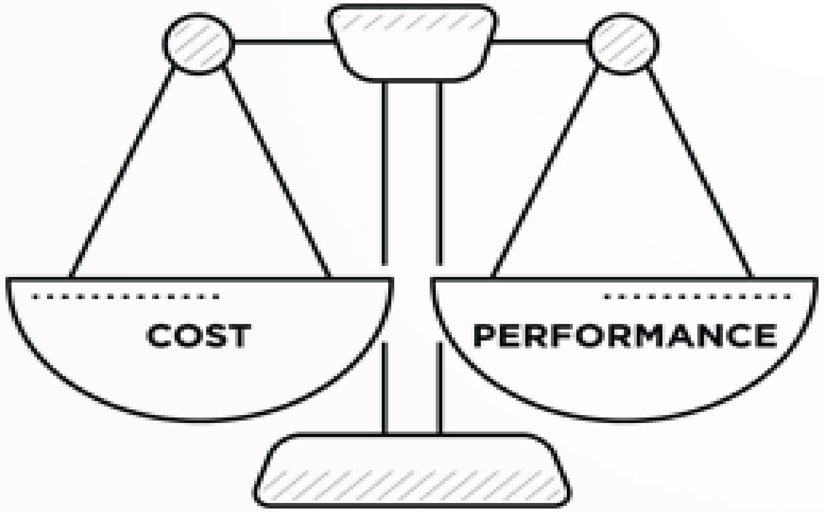
without executing the software program



performed while the software application is running on a physical or virtual processor

The use of profiling allows you to identify bottlenecks in the code execution that may be a result of inefficient code, or poor communication between function interactions with a module in the PL or another function within software.

Example



$$C = 5x_1 + 6x_2 + 4x_3$$

$$x_1 + x_2 + x_3 \leq 10\,ns$$

$$x_1, x_2, x_3 \in \{0,1\}$$

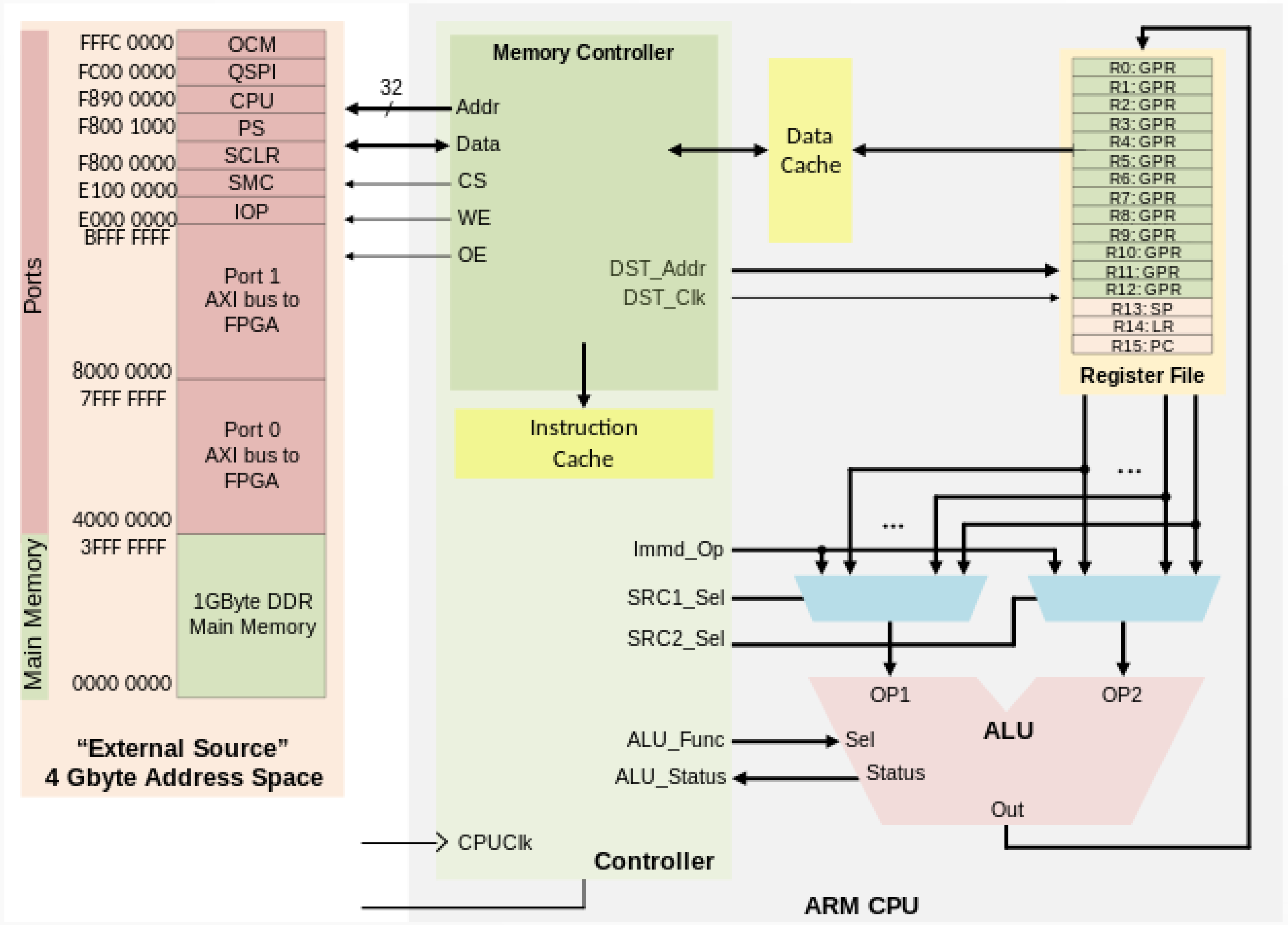
x_1	x_2	x_3	K
0	1	1	10
1	0	1	9
1	1	0	11
1	1	1	15

Lets suppose that we have :

- 500 MHz ==> cpu freq.
- 0.5ns per cycle.
- Three functions : X1, X2, X3

  X1 independent from X2, X3 depends on X1 and X2
- Hardware cost : X2 > X1 >X3
- Cycle cost: X1=10, X2=12, X3=8.

APU



ARM cpu :

- The ARM is a RISC machine that uses a load-store architecture.
- All ARM instruction codes are 32 bits, and the ARM memory bus is 32 bits.
- "Conditional execution" for instructions.

if(x == 0)y = 1;

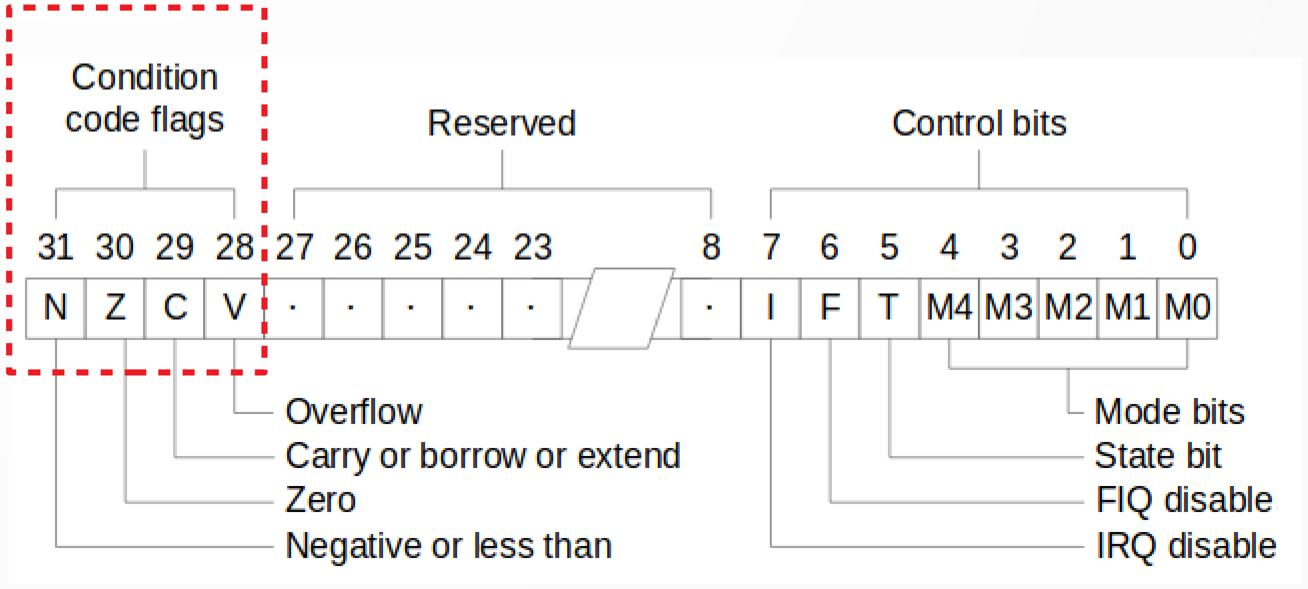
Load x in r0
cmp r0,#0
bne endif
then:
mov r1, #1
// now store r1 in [y]
Endif:
……

APU

Condition flags:

- N is a ‘1’ if the last ALU operation produced a negative number;
- Z is a ‘1’ if the last operation produced a ‘0’ result;
- C is a ‘1’ if the last operation produced a carry-out;
- V is a ‘1’ if the last operation produced an overflow condition.

Current Program Status Register (CPSR)

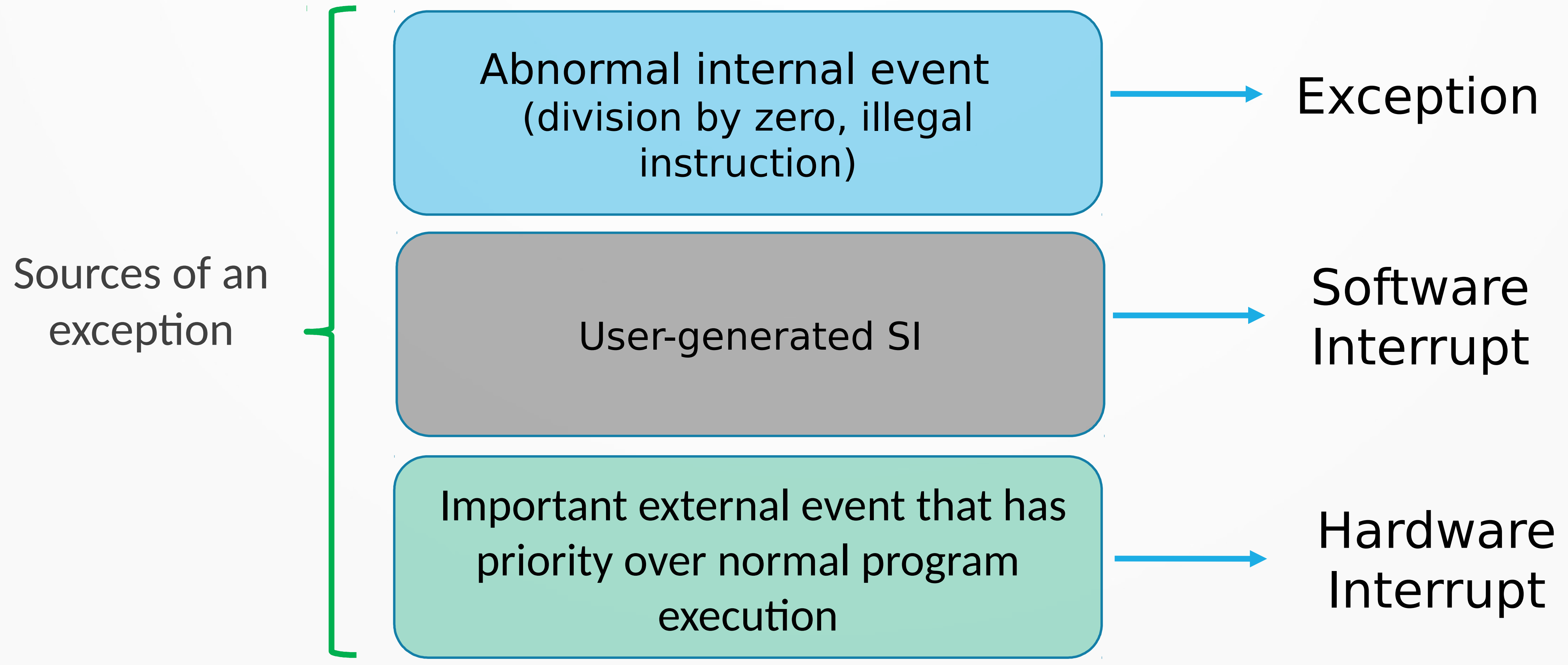


Example

```
MOV R2,#10      @ R2 <- 10; R2 is loop index
myloop:
ADD R1,R1, #1   @ Increment R1
SUBS  R2,R2, #1   @ Decrement R2
BNE  myloop  @ If R2 != 0, branch to myloop
MOV R4, 0xbeef @ Continue on with program...
```

Exceptions

Special condition that requires a processor's immediate attention

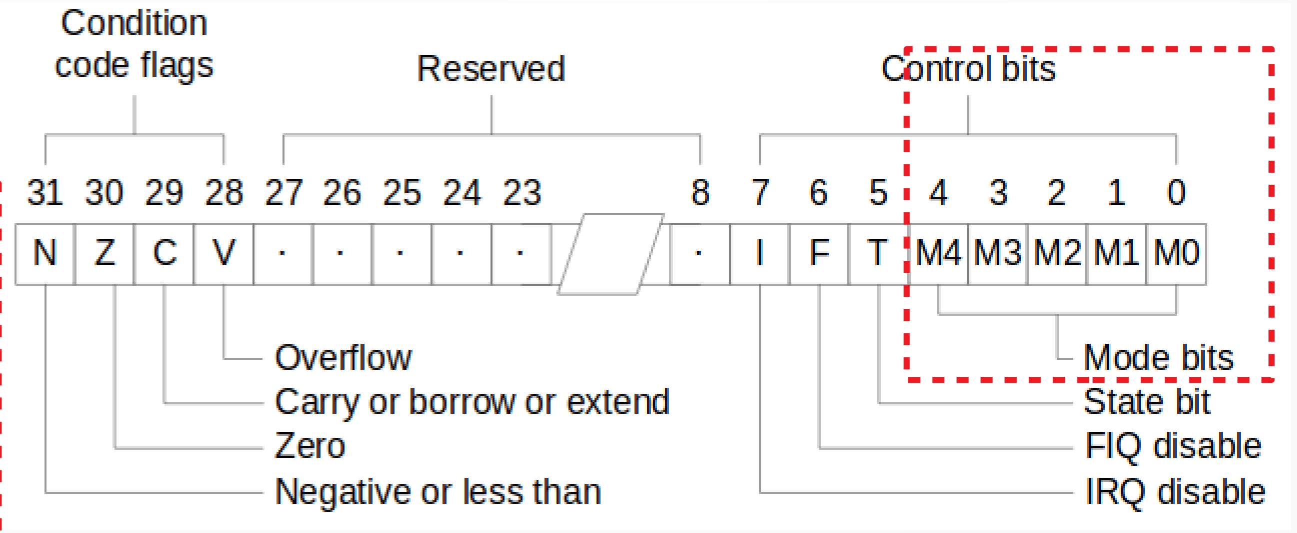

Sources of an exception
Abnormal internal event (division by zero, illegal instruction)
Exception
User-generated SI
Software Interrupt
Important external event that has priority over normal program execution
Hardware Interrupt

Exceptions

ARM processor modes :

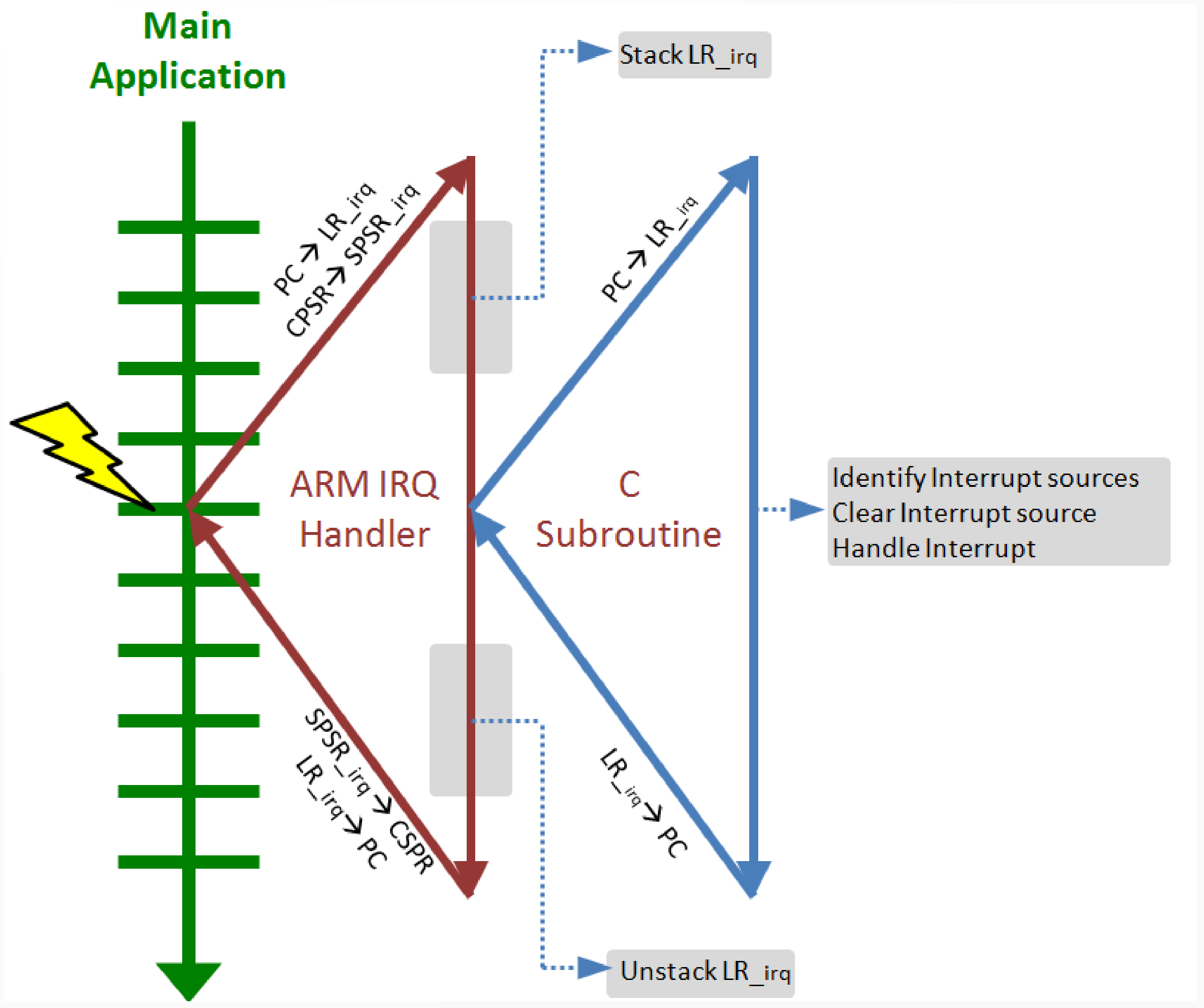
- User mode is the basic mode in which application programs run. User mode is the only unprivileged mode, and it has restricted access to system resources. Typically, a processor spends more than 99% of its time in user mode.
- System mode provides unrestricted access to all system resources.
- Supervisor mode also provides unrestricted access to all system resources.
- Abort mode is entered if a program attempts to access a non-existing memory location.
- Undefined mode is entered for attempt to execute an unimplemented instruction.
- IRQ mode is entered in response to a normal interrupt request from an external device.
- FIQ mode is entered in response to a fast interrupt request from an external device. It is used to provide faster service for more urgent requests.

Current Program Status Register (CPSR)



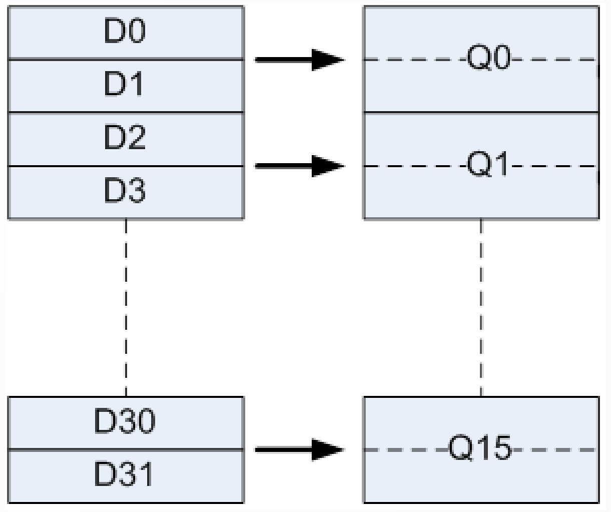
Exceptions

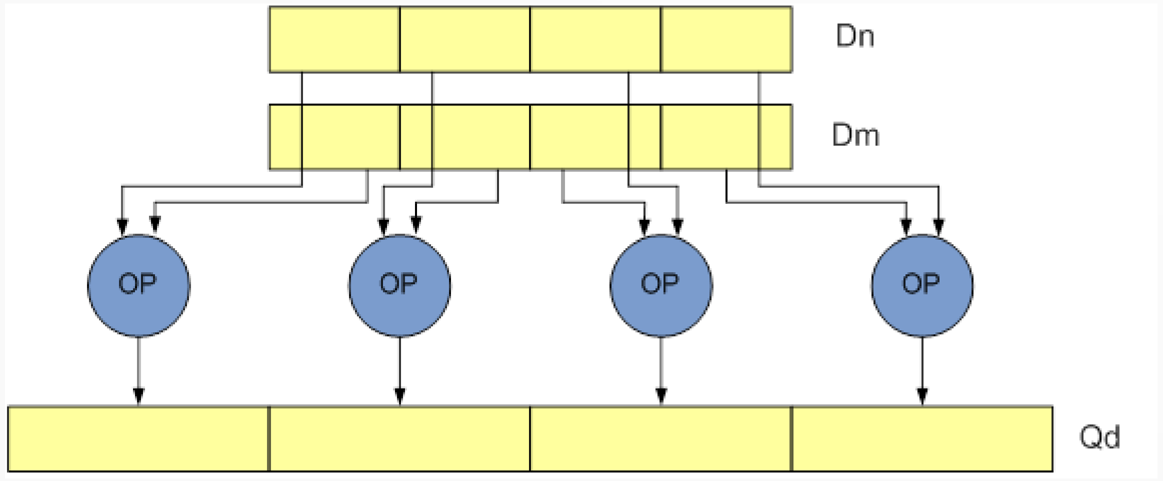
- ARM switches to IRQ (or FIQ) mode.
- Saves the current PC in the local LR and the current CPSR in the local SPSR
- Sets the IRQ (or FIQ) disable bit in the CPSR to a ‘1’.
- Loads the proper vector into the PC (vector 0x18 for IRQ, and 0x1C for FIQ).
- The instruction stored at the vector address will be a branch to a generic Interrupt handler.
- The user-written interrupt handler must determine what the source of the interrupt was (by checking the interrupt’s ID#), and then branch to the code to deal with the actual interrupt.

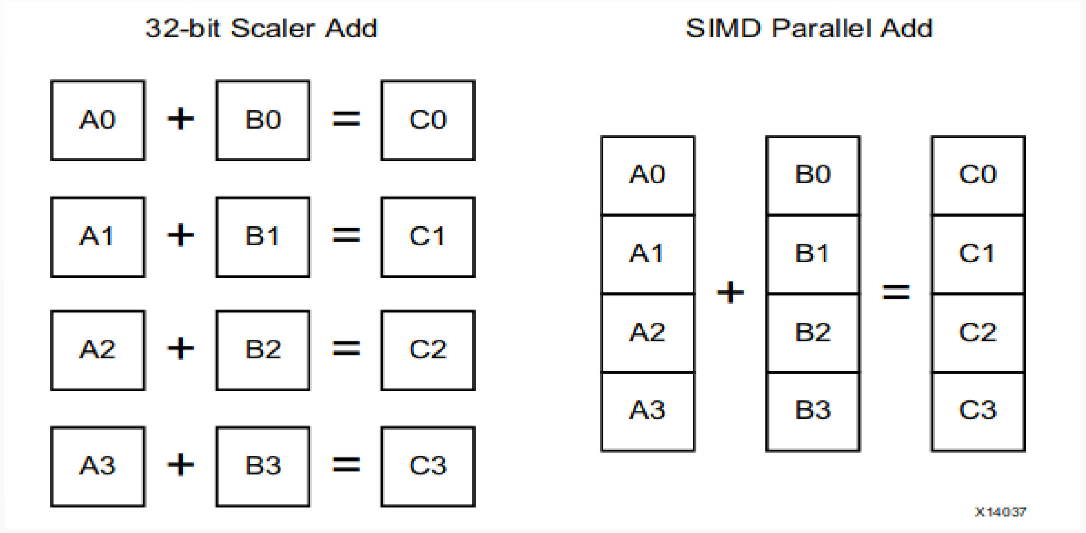


Neon engine

- Armv7-A and AArch32 have 32 x 64-bit NEON registers (D0-D31). These registers can also be viewed as 16x128-bit registers (Q0-Q15).
- Variety of operations are available through special instructions dedicated to be executed on the unit.







Neon engine

Vectorizing compilers

To enable automatic vectorization, you must add -mfpu=neon and -ftree-vectorize to the GCC command line. For example:

```
arm-none-linux-gnueabi-gcc -mfpu=neon -ftree-vectorize -c vectorized.c
```

NEON optimized libraries

Arm’s open source project.

Currently, the Ne10 library provides some math, image processing and FFT function.

NEON assembly

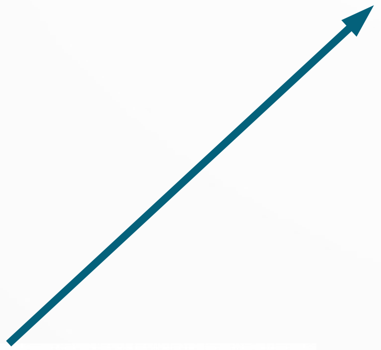
```
VADD.I8 D0, D1, D2
VMULL.S16 Q2, D8, D9
```

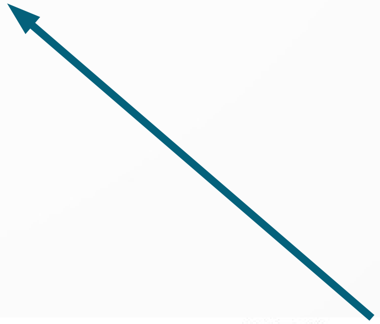
NEON intrinsics

```
#include<arm_neon.h>
........
float32_t in1, in2, out;
in1 = vld1q_f32(src1);
in2 = vld1q_f32(src2);
out = vaddq_f32(in1, in2);
vst1q_f32(dst, out);
```

PS

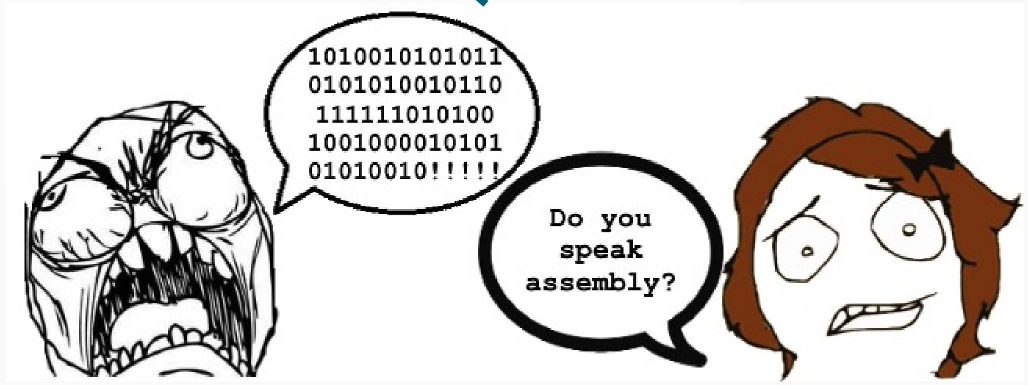
Performance





Co-processors

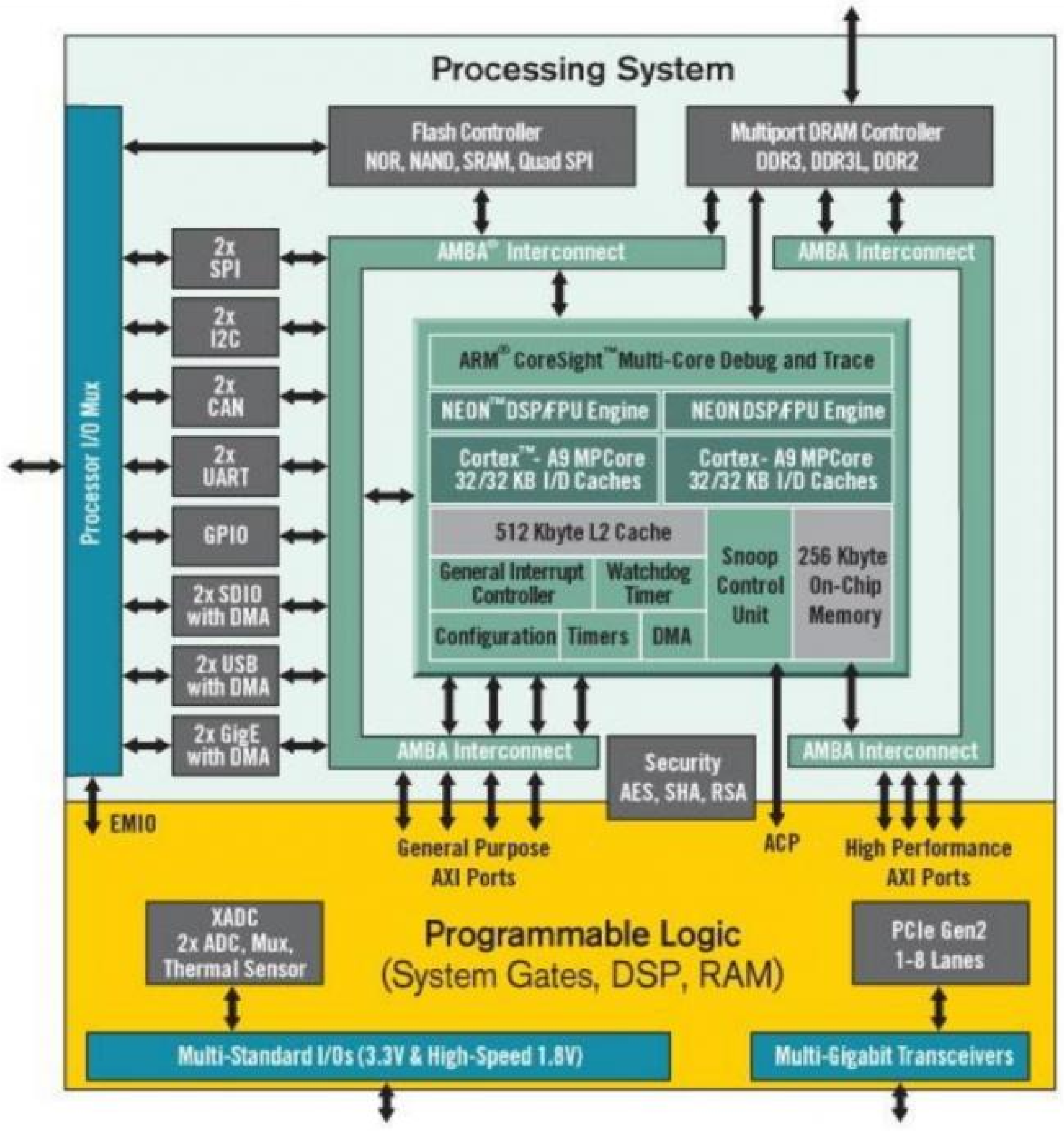
Software optimization



PS

Processing system (PS)

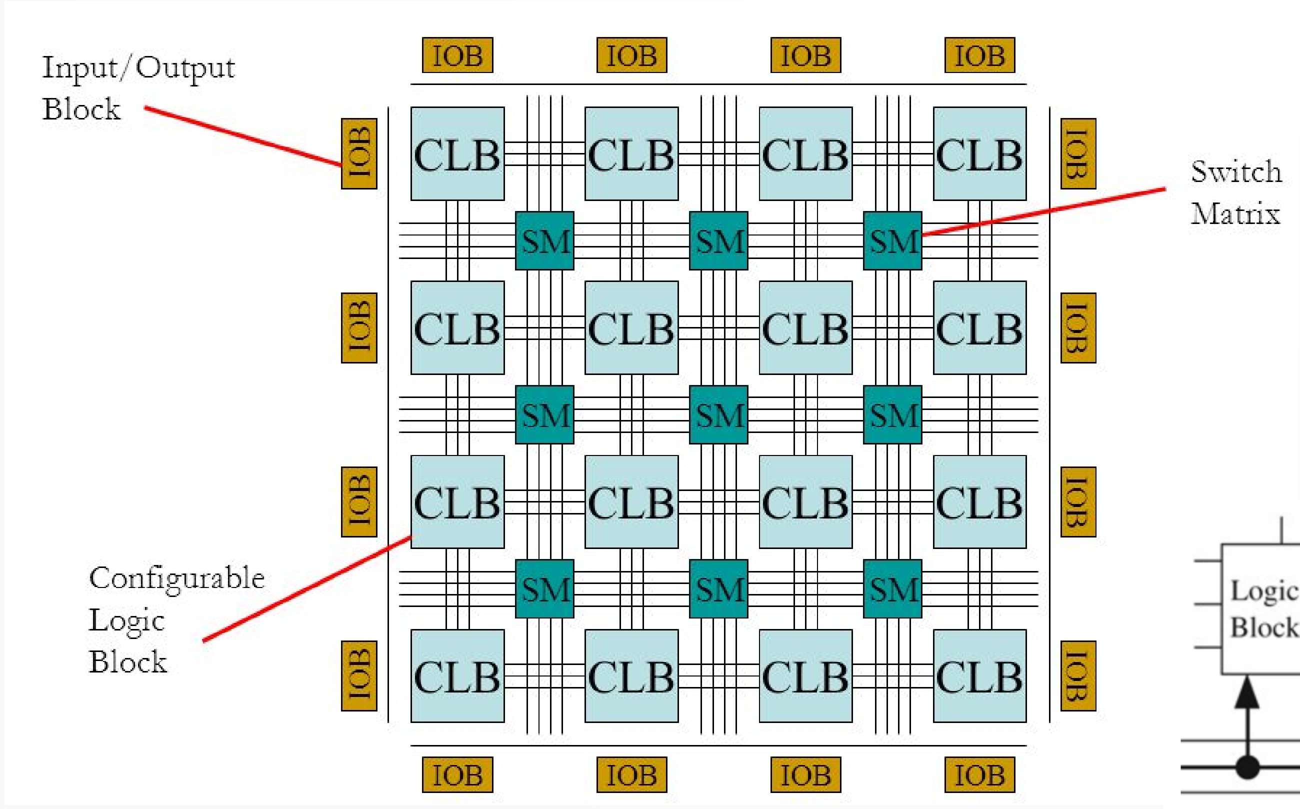
- Double core A9 processor.
- Co-processors.
- Various peripherals.
- MIO
- OCM
- DRAM Controller.
- Flash controller.

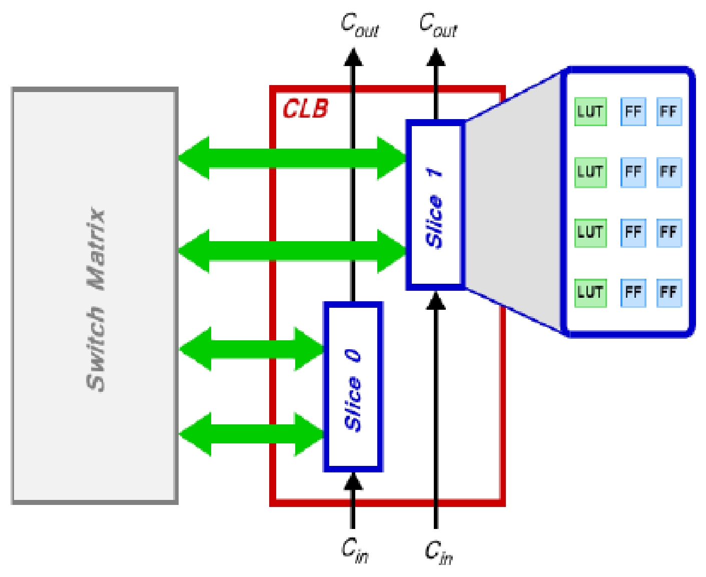


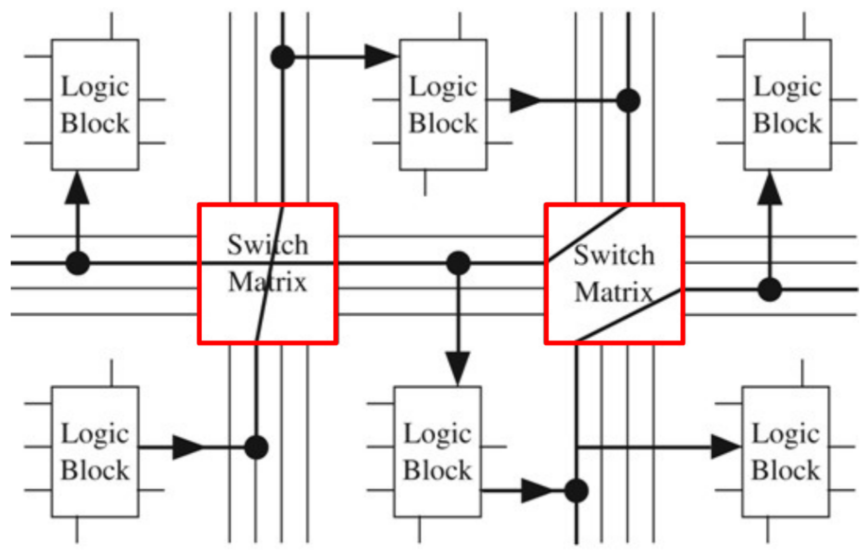
Processing System
Flash Controller NOR, NAND, SRAM, Quad SPI
Multiport DRAM Controller DDR3, DDR3L, DDR2
Processor I/O Mux
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
AMBA® Interconnect
AMBA Interconnect
ARM® CoreSight™ Multi-Core Debug and Trace
NEON™ DSP/FPU Engine
NEON DSP/FPU Engine
Cortex™- A9 MPCore 32/32 KB I/D Caches
Cortex- A9 MPCore 32/32 KB I/D Caches
512 Kbyte L2 Cache
Snoop Control Unit
256 Kbyte On-Chip Memory
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
AMBA Interconnect
Security AES, SHA, RSA
AMBA Interconnect
EMIO
General Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2x ADC, Mux, Thermal Sensor
Programmable Logic (System Gates, DSP, RAM)
PCIe Gen2 1-8 Lanes
Multi-Standard I/Os (3.3V & High-Speed 1.8V)
Multi-Gigabit Transceivers

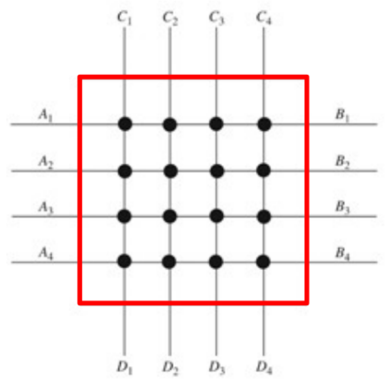
Figure 3. Zynq AP SoC architecture.

PL


Input/Output Block
Switch Matrix
Configurable Logic Block
IOB
CLB
SM


Switch Matrix
CLB
Slice 1
Slice 0
C_{out}
C_{out}
C_{in}
C_{in}
LUT
FF


Logic Block
Switch Matrix


C_1 C_2 C_3 C_4
A_1 A_2 A_3 A_4
B_1 B_2 B_3 B_4
D_1 D_2 D_3 D_4

Operators sharing

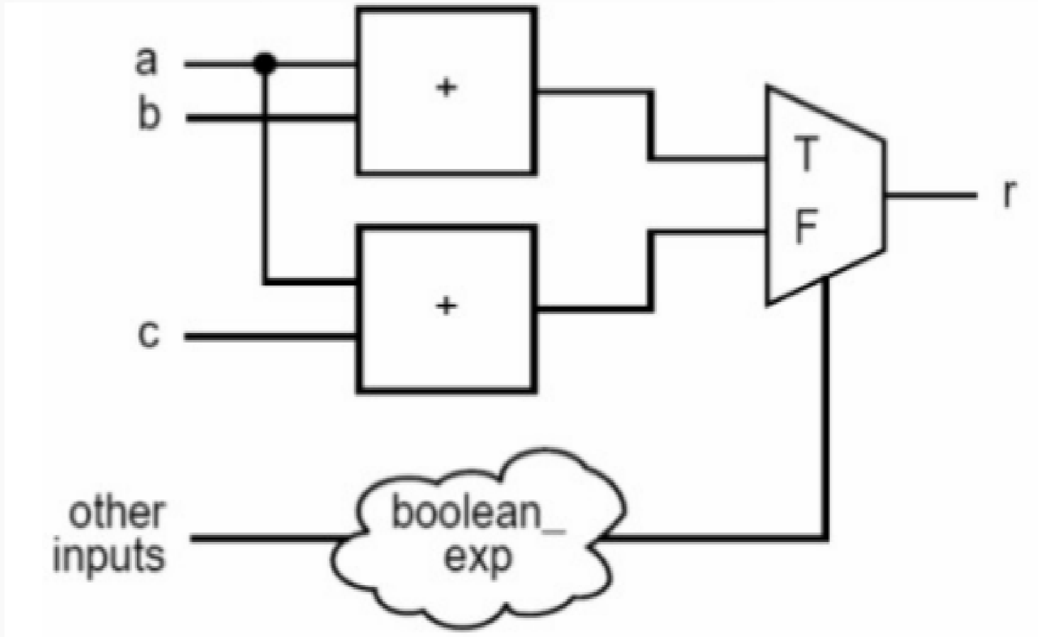
Original code:

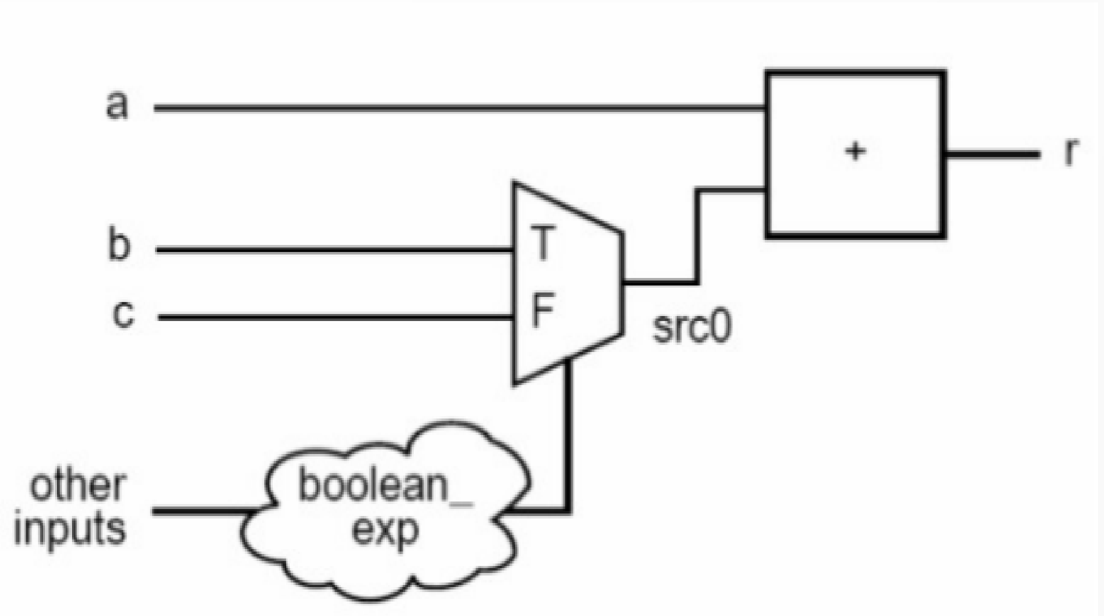
r <= a+b when boolean_exp else a+c;

Revised code:

src0 <= b when boolean_exp else c;

r <= a + src0;





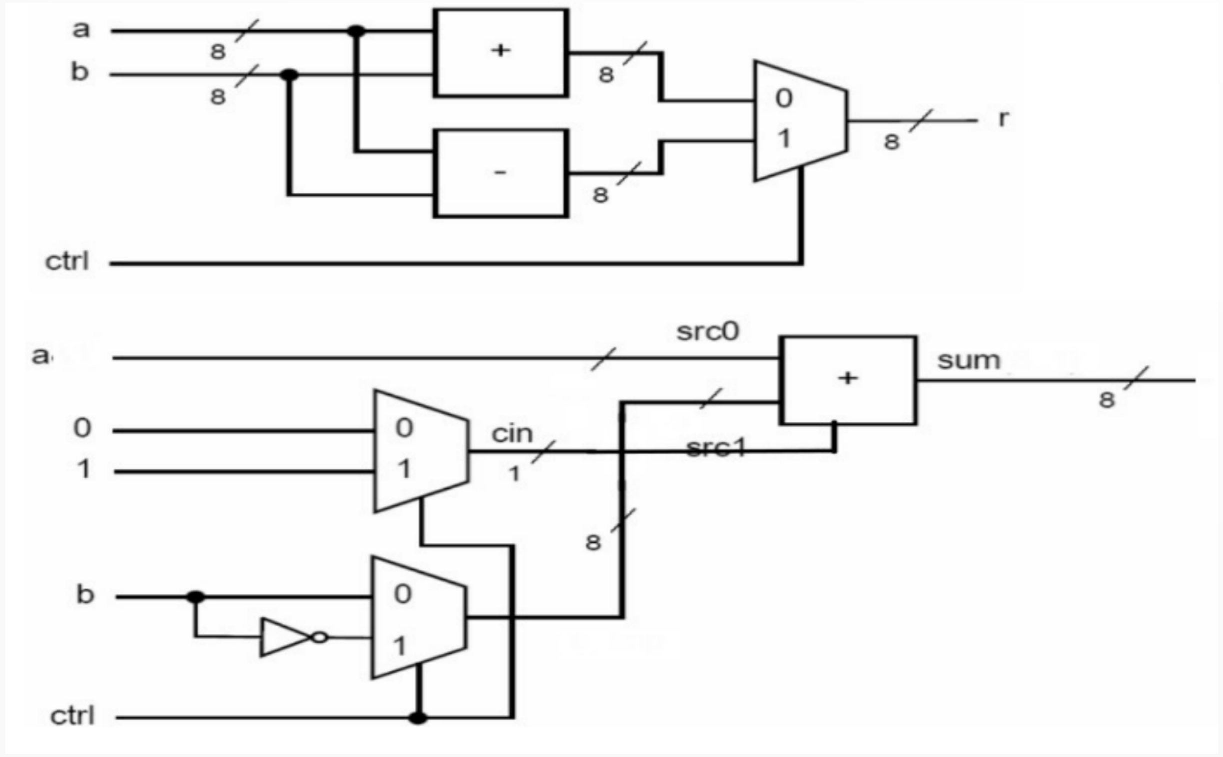
Functionality sharing

A large circuit involves lots of functions.

Several functions may be related and have common characteristics.

Several functions can share the same circuit.

Needs the understanding and insight of the designer.



```
sum<=a+b when ctrl ='0' else a-b;
r<=std_logic_vector(sum);
```

```
src0 <= a;
src1 <= b when ctrl='0' else not b;
cin <= "0" when ctrl='0' else "1";
sum <= src0 + src1+cin;
r <=r<=std_logic_vector(sum);
```

Layout related optimization

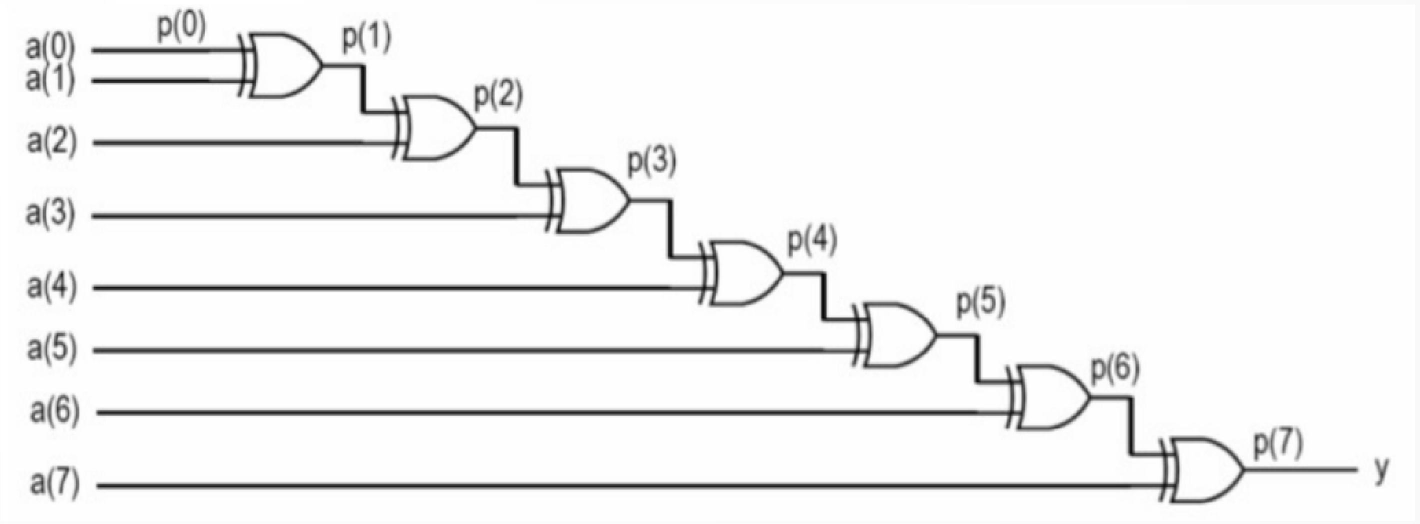
Example :

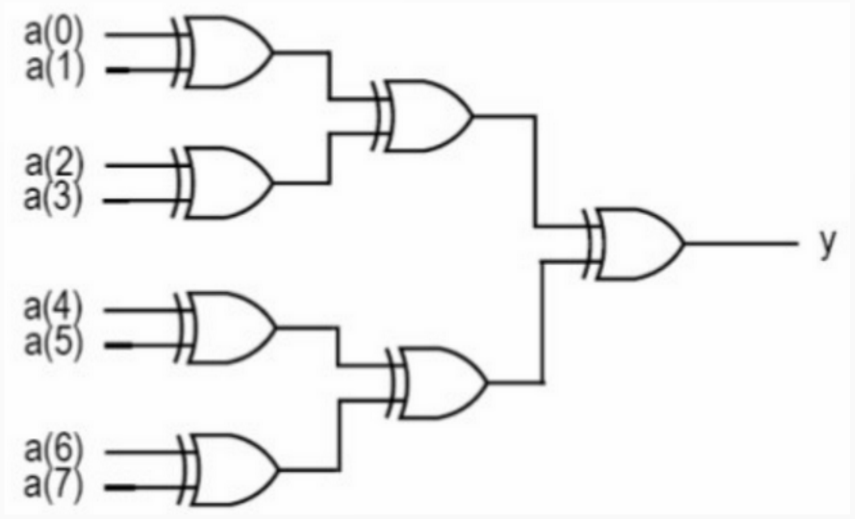
Cascaded xor function

y=a1 xor a2 ….xor a3

- Longer path mean longer delay.

- Rectangular shapes usually are favorable.



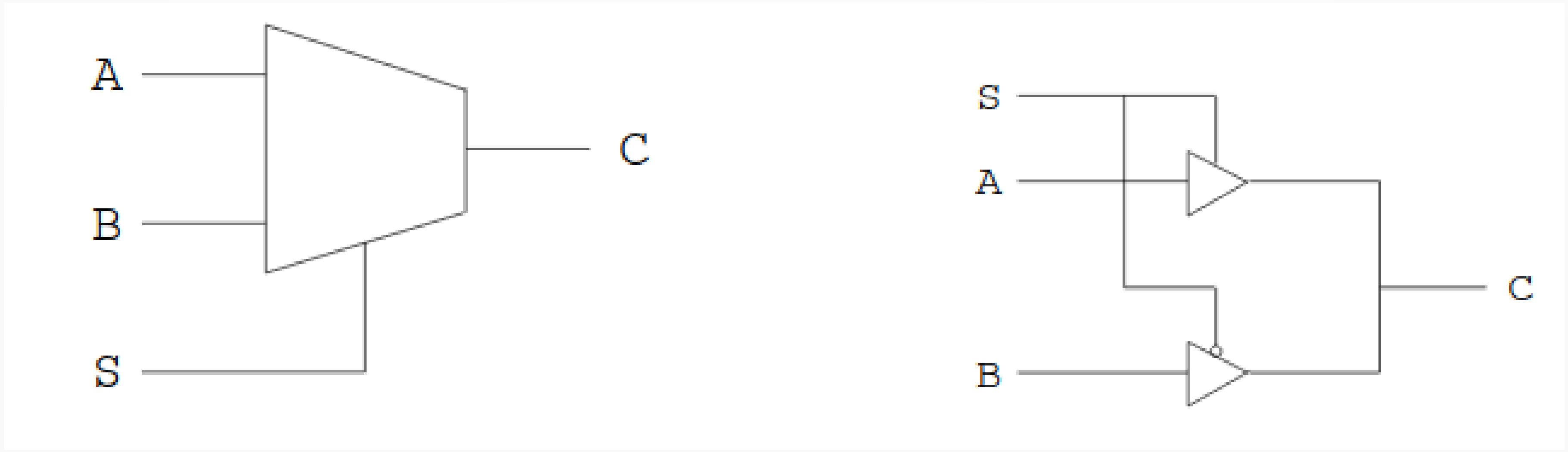


Signal selection

Large multiplixers when implemented in FPGA consume large number of CLB resources, as the number of input increases, tristate solution offers competitive timing.

```
C <= B when C else A
```

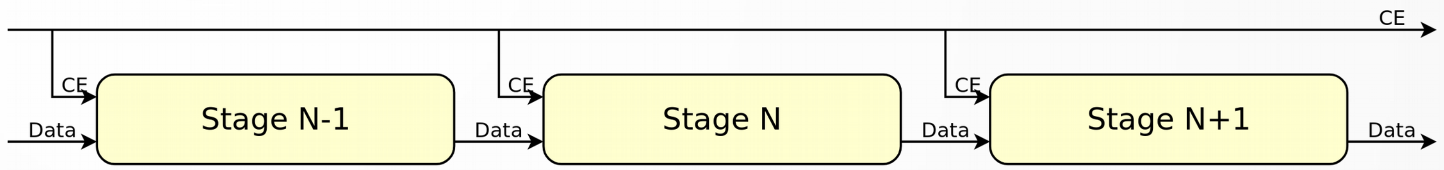
```
If S='0' then
        C <= A;
else
        C <= B;
```



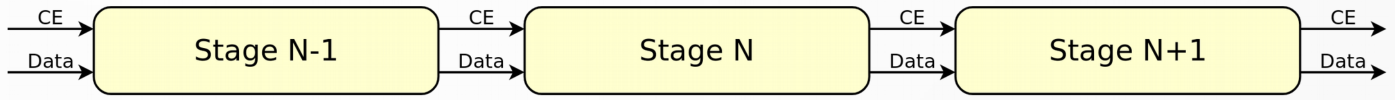
Pipelining

In FPGA digital logic takes place in parallel. One solution to sequencing operations is to create a giant state machine. The reality, though, is that an FPGA tends to create all the logic for every state at once, and then only select the correct answer at the end of each clock tick.

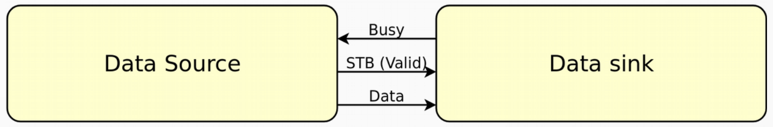
Pipelining tends to be faster than the state machine approach for accomplishing the same algorithm, and it can even be more resource efficient, although it isn't necessarily so



Global valid signal (fixed data rate)



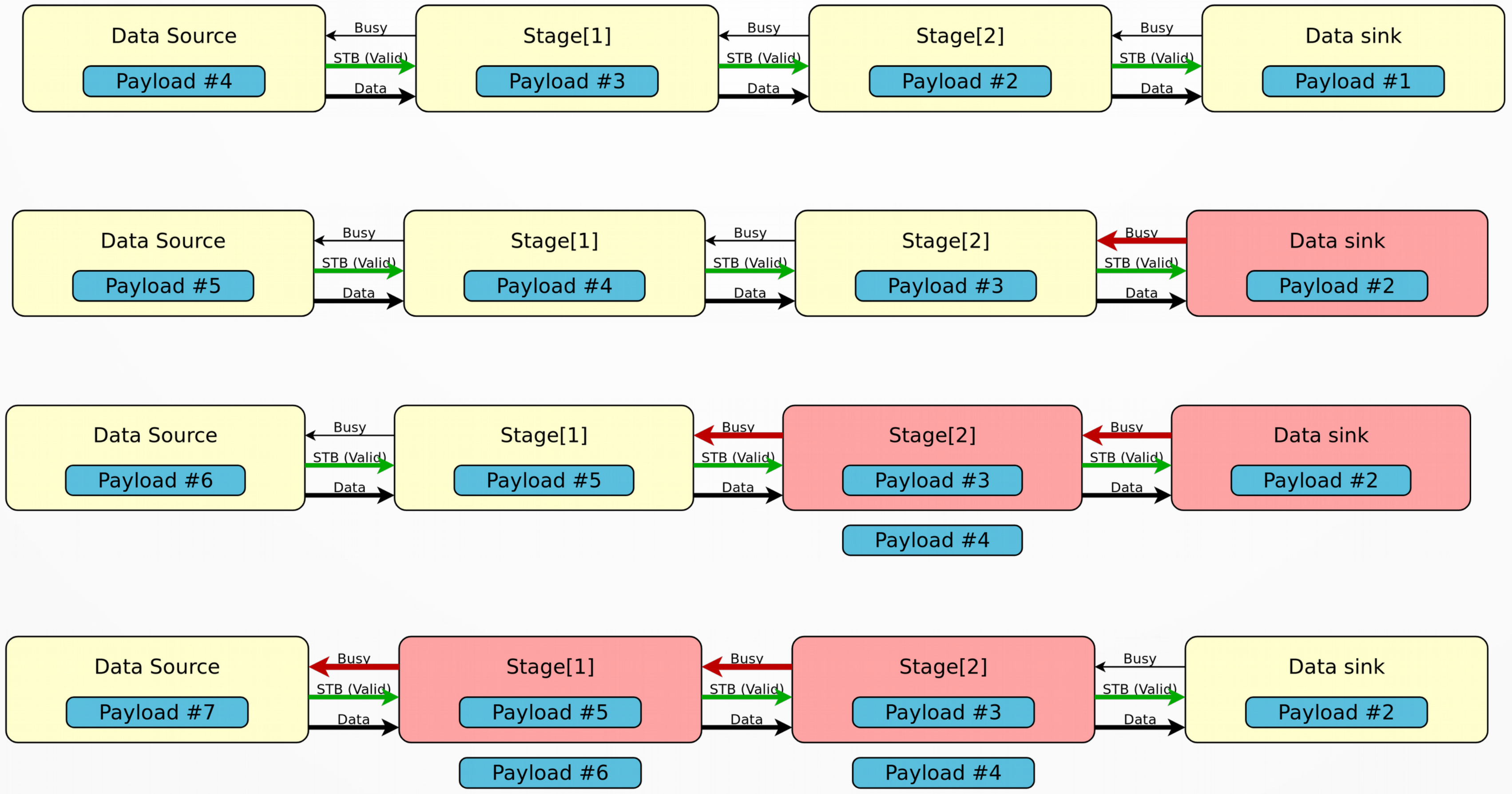
Traveling CE signal (fixed execution time at each stage)



Simple handshake

Pipelining

Buffered handshake

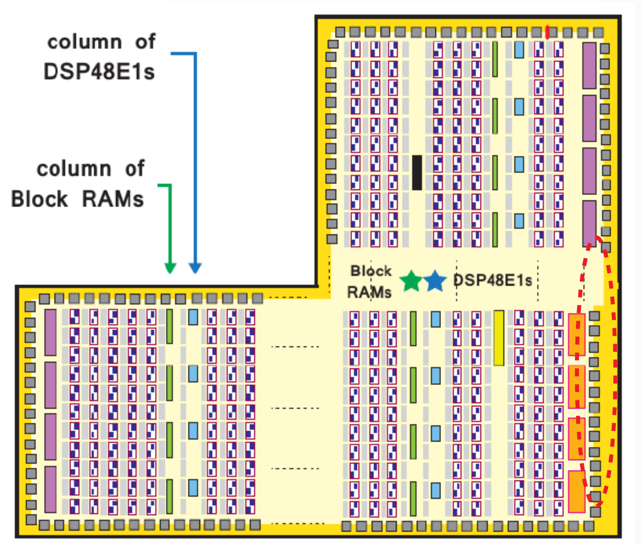


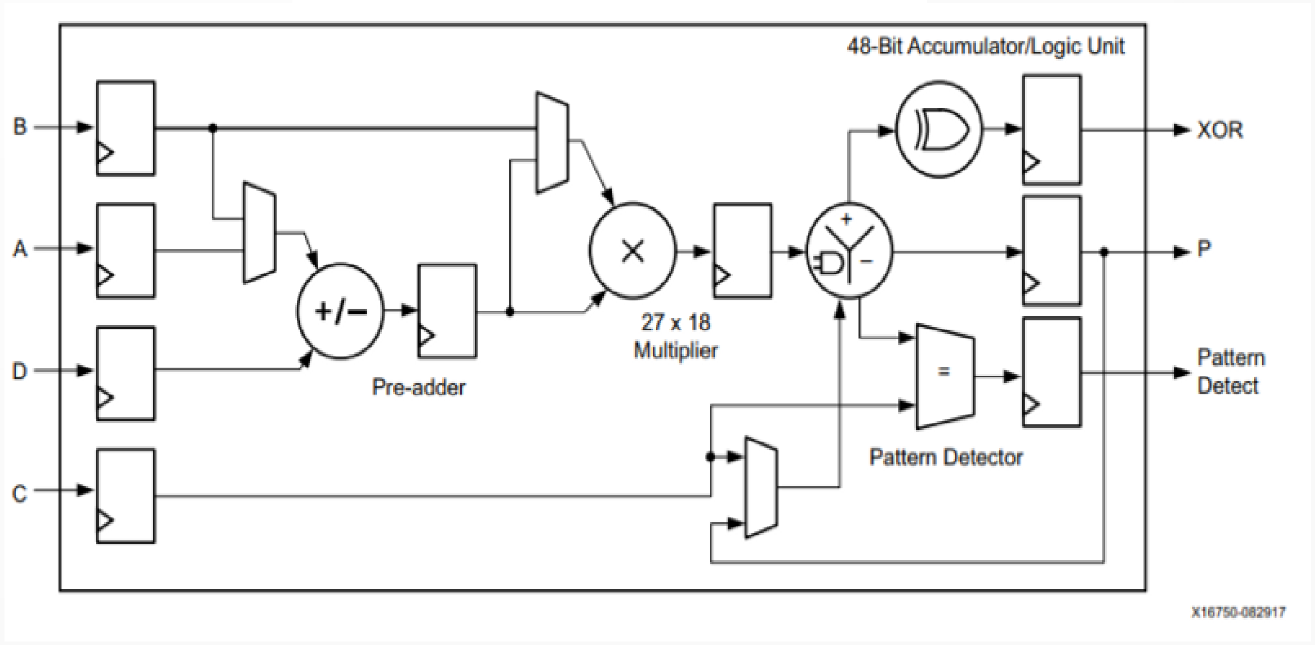
DSP

DSP48E1 slices features:

- 48 bits add/substract/accumulate.
- 27 x 18 bits multiplier.
- 25 bits pre-adder.
- Cascade path.
- Pipeline registers for high speed.
- Pattern detector.
- SIMD operations.

Example : P=(A+D).B





DSP

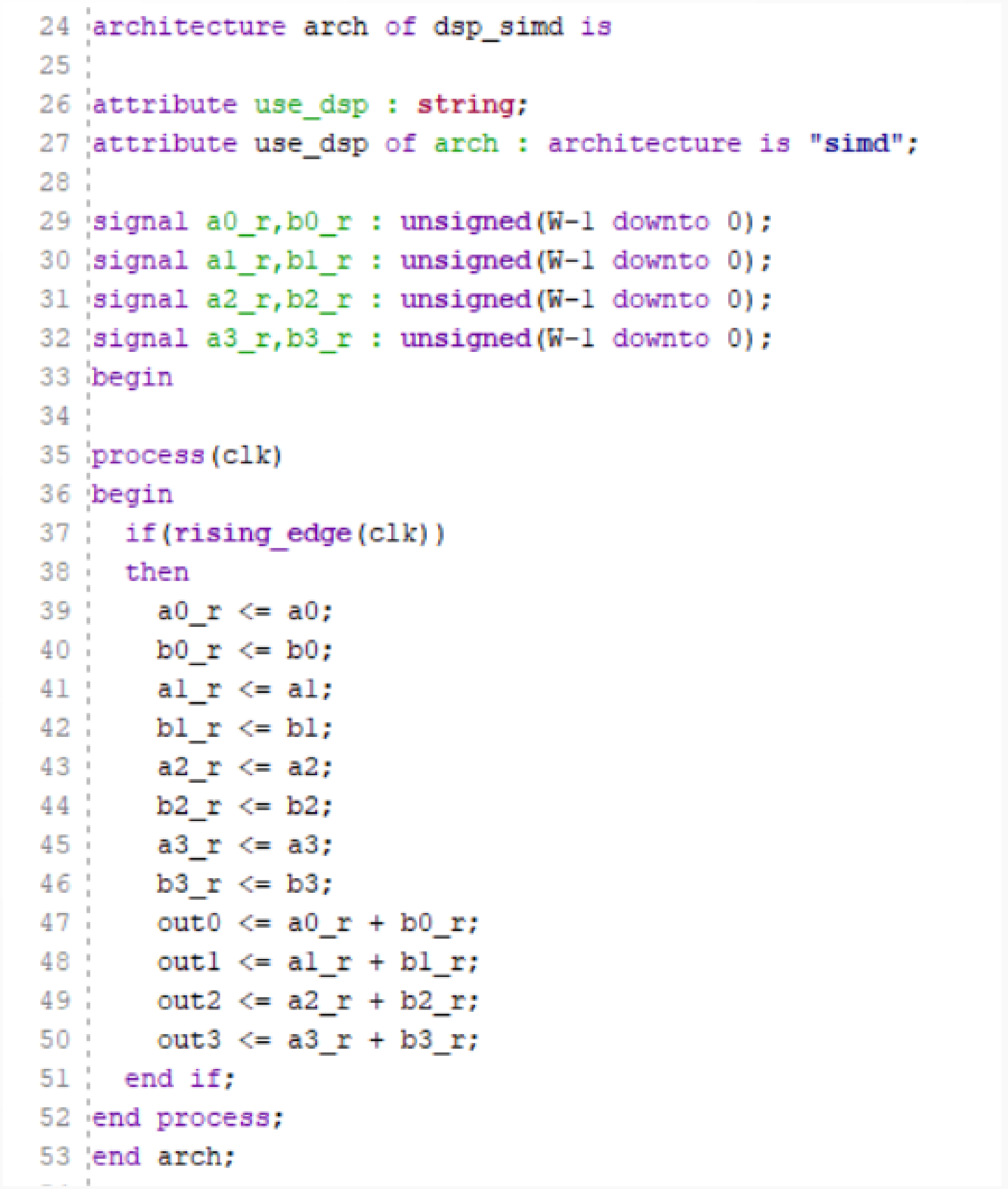
Developing RTL to take advantage of the SIMD capabilities of the DSP48 is straight forward. We can do this in our RTL by setting an attribute for the synthesis tool to detect such that it can infer the correct DSP48 implementation function and mode / OPCODE.

- In our source code, the attributes are defined as shown below:

  (VHDL)

  attribute use_dsp : string;

  attribute use_dsp of arch : architecture is “simd”;

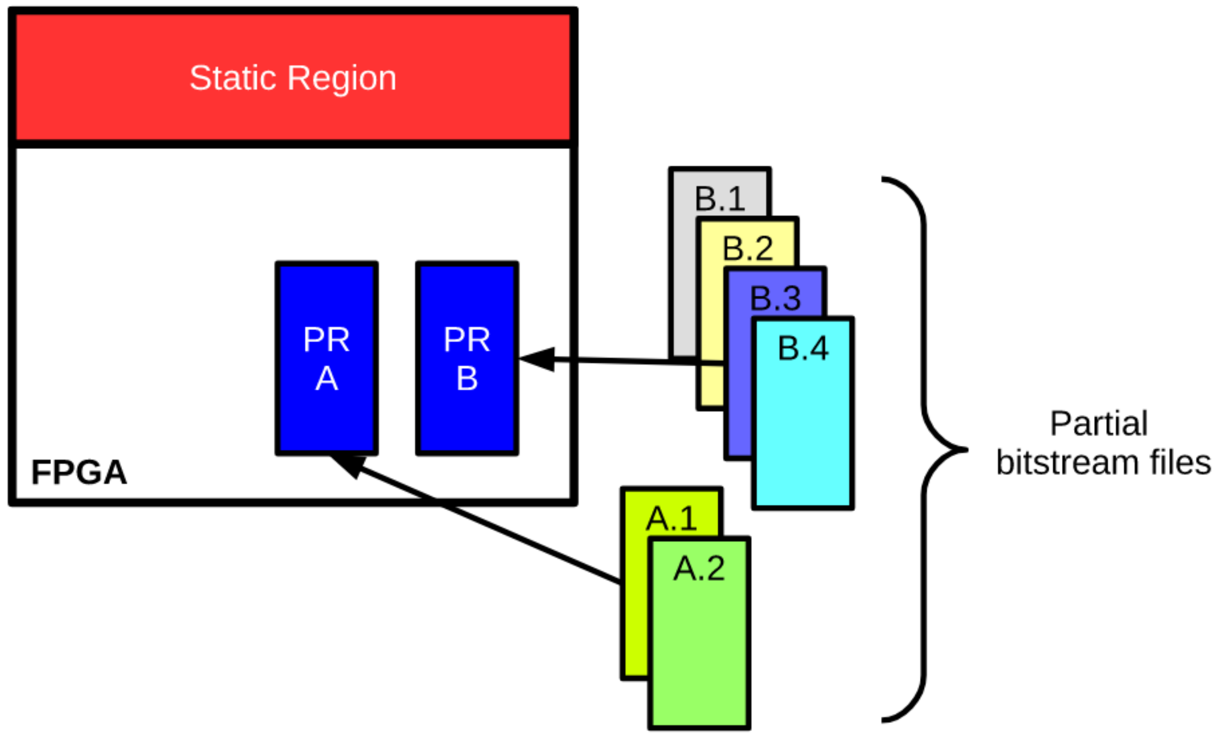


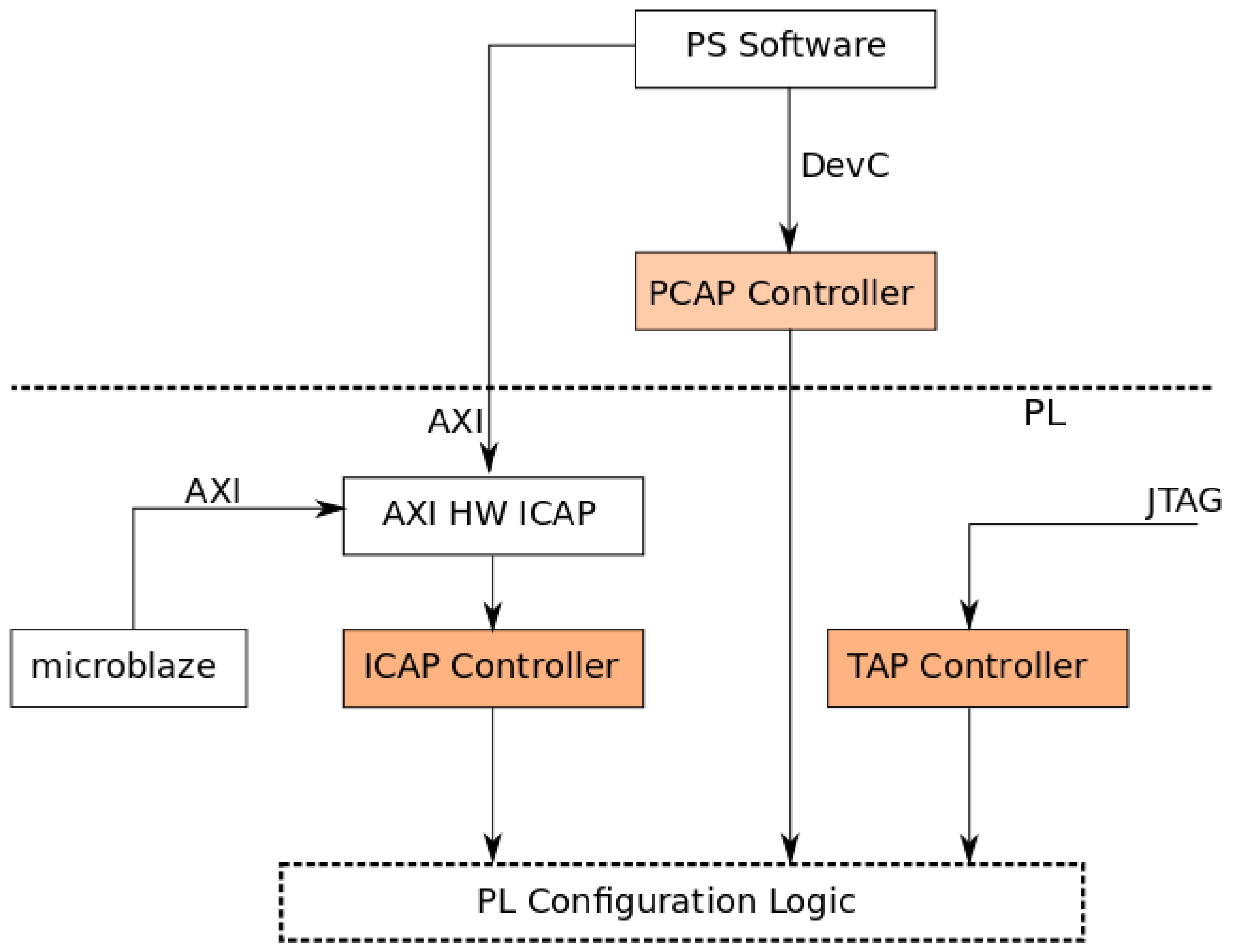
```
architecture arch of dsp_simd is

attribute use_dsp : string;
attribute use_dsp of arch : architecture is "simd";

signal a0_r,b0_r : unsigned(W-1 downto 0);
signal a1_r,b1_r : unsigned(W-1 downto 0);
signal a2_r,b2_r : unsigned(W-1 downto 0);
signal a3_r,b3_r : unsigned(W-1 downto 0);
begin

process(clk)
begin
  if(rising_edge(clk))
  then
    a0_r <= a0;
    b0_r <= b0;
    a1_r <= a1;
    b1_r <= b1;
    a2_r <= a2;
    b2_r <= b2;
    a3_r <= a3;
    b3_r <= b3;
    out0 <= a0_r + b0_r;
    out1 <= a1_r + b1_r;
    out2 <= a2_r + b2_r;
    out3 <= a3_r + b3_r;
  end if;
end process;
end arch;
```

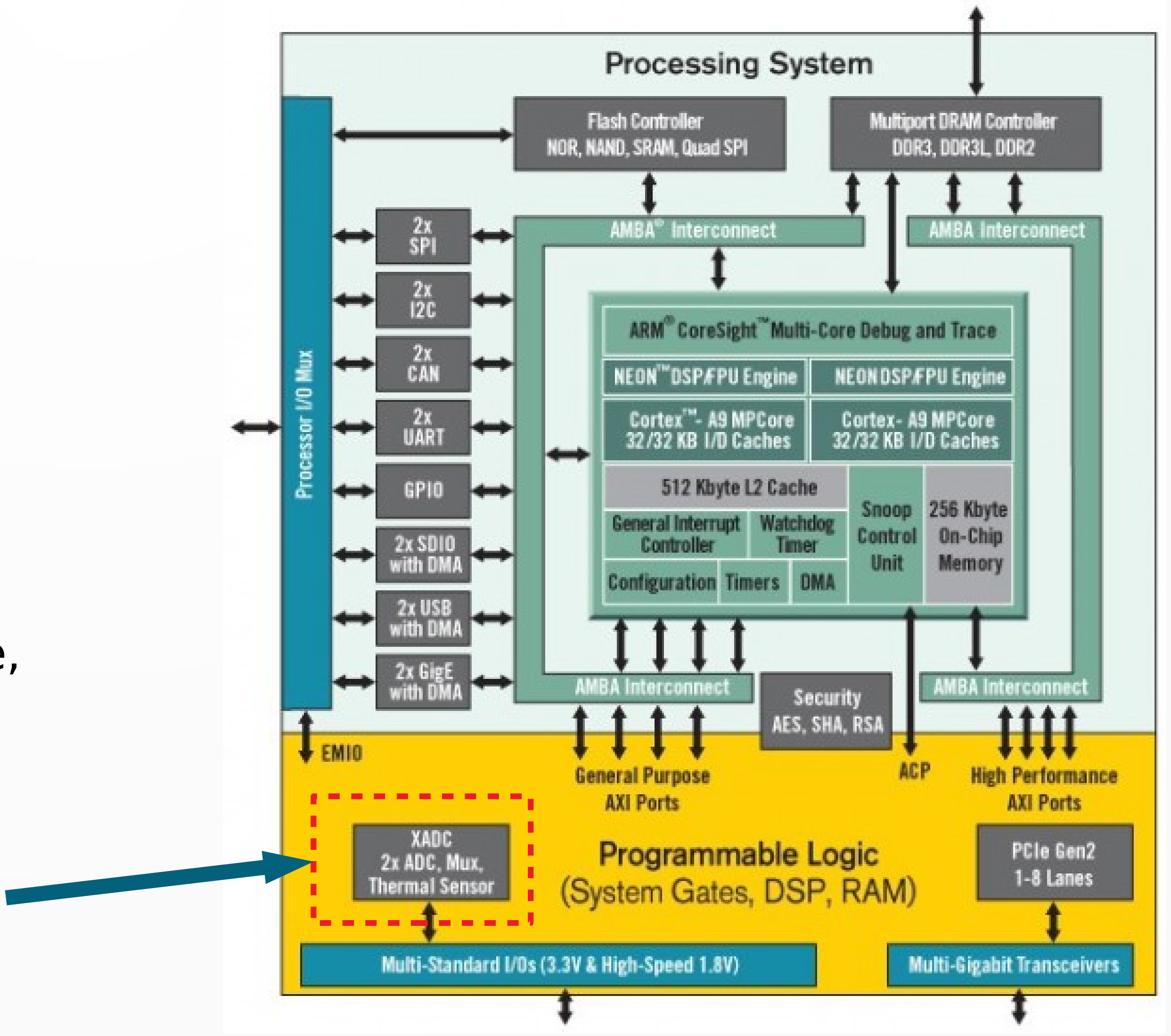

Static Region
PR A
PR B
FPGA
B.1
B.2
B.3
B.4
A.1
A.2
Partial bitstream files


PS Software
DevC
PCAP Controller
PL
AXI
AXI
AXI HW ICAP
microblaze
ICAP Controller
JTAG
TAP Controller
PL Configuration Logic

XADC

The XADC includes:

- Dual 12-bit ADC.
- 1 Mega sample per second (MSPS).
- On-chip sensors(temperature and voltage).
- Range of operating modes, for example, externally triggered and simultaneous sampling on both ADCs.

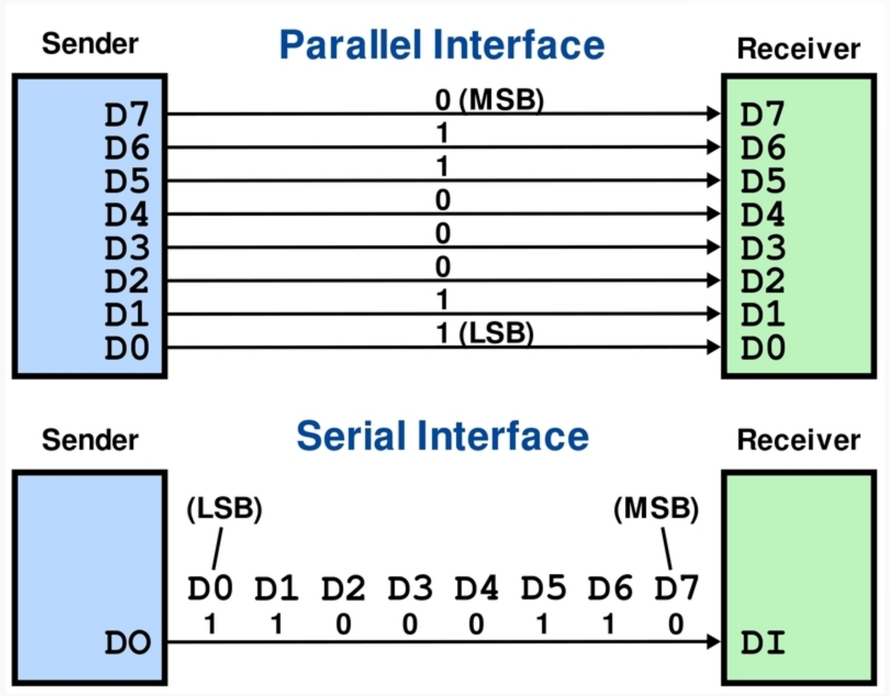


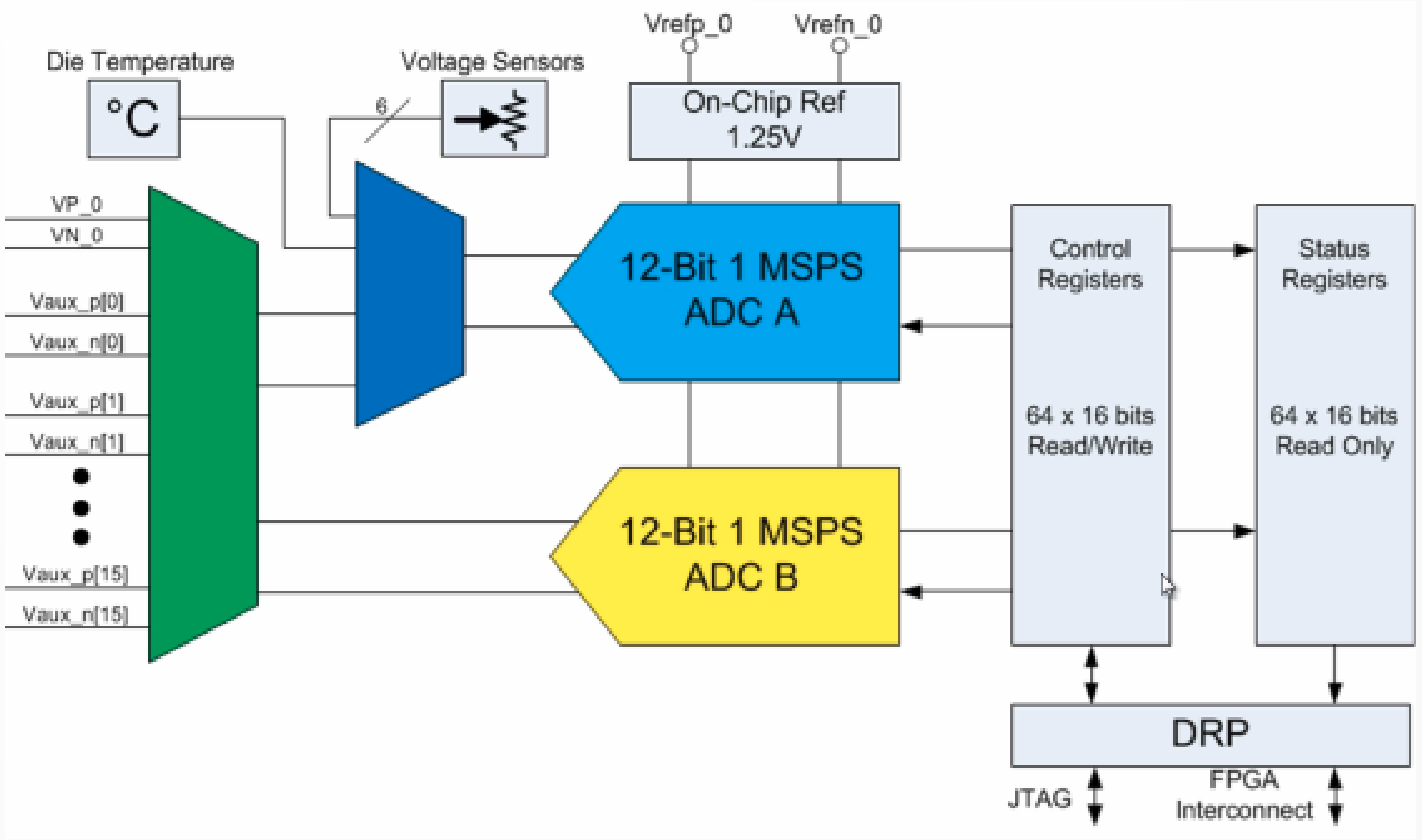
Processing System
Flash Controller NOR, NAND, SRAM, Quad SPI
Multiport DRAM Controller DDR3, DDR3L, DDR2
Processor I/O Mux
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
AMBA® Interconnect
AMBA Interconnect
ARM® CoreSight™ Multi-Core Debug and Trace
NEON™ DSP/FPU Engine
NEON DSP/FPU Engine
Cortex™- A9 MPCore 32/32 KB I/D Caches
Cortex- A9 MPCore 32/32 KB I/D Caches
512 Kbyte L2 Cache
General Interrupt Controller
Watchdog Timer
Configuration
Timers
DMA
Snoop Control Unit
256 Kbyte On-Chip Memory
AMBA Interconnect
Security AES, SHA, RSA
AMBA Interconnect
EMIO
General Purpose AXI Ports
ACP
High Performance AXI Ports
XADC 2x ADC, Mux, Thermal Sensor
Programmable Logic (System Gates, DSP, RAM)
PCIe Gen2 1-8 Lanes
Multi-Standard I/Os (3.3V & High-Speed 1.8V)
Multi-Gigabit Transceivers

XADC

12-bit resolution conversion.

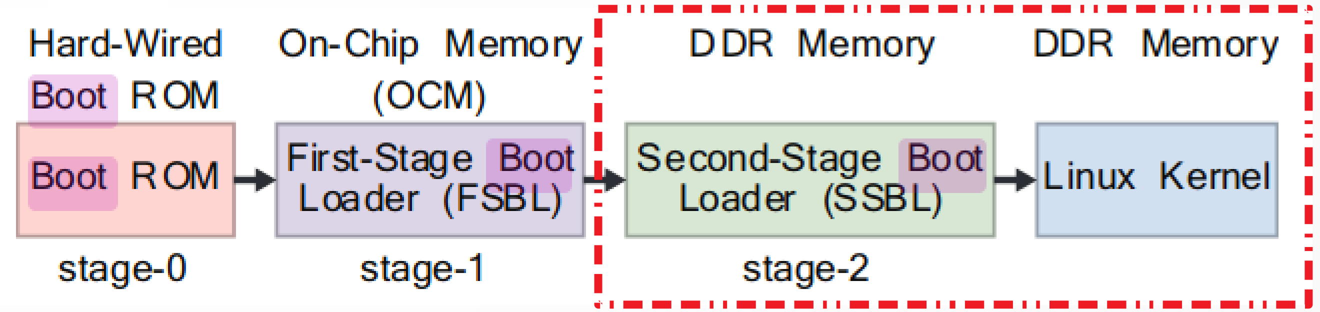
- Built in digital gain and offset calibration.
- On-chip thermal and Voltage sensors.
- Sample rate of 1 MSPS.

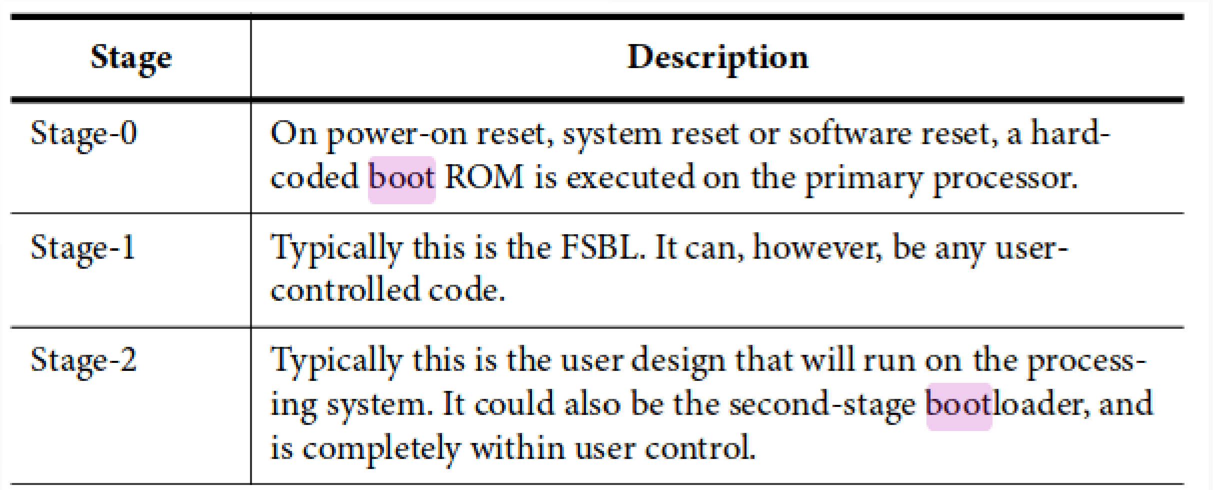




Boot sequence

- Boot ROM which is initialized at power -on. The value of the boot mode strapping pins of the device determines the boot mode.
- The boot mode defines from which of the supported interfaces JTAG, QSPI Flash or SD card .
- Boot Rom load the FSBL image from the specified interface to the OCM. Once the image is transferred to the OCM, the control of the CPU is handed over to the FSBL.
- Typically, the FSBL contains instructions for the CPU to further configure the PS , and to configure the PL using Processor Configuration Access Port (PCAP) if the a bitstream is available.





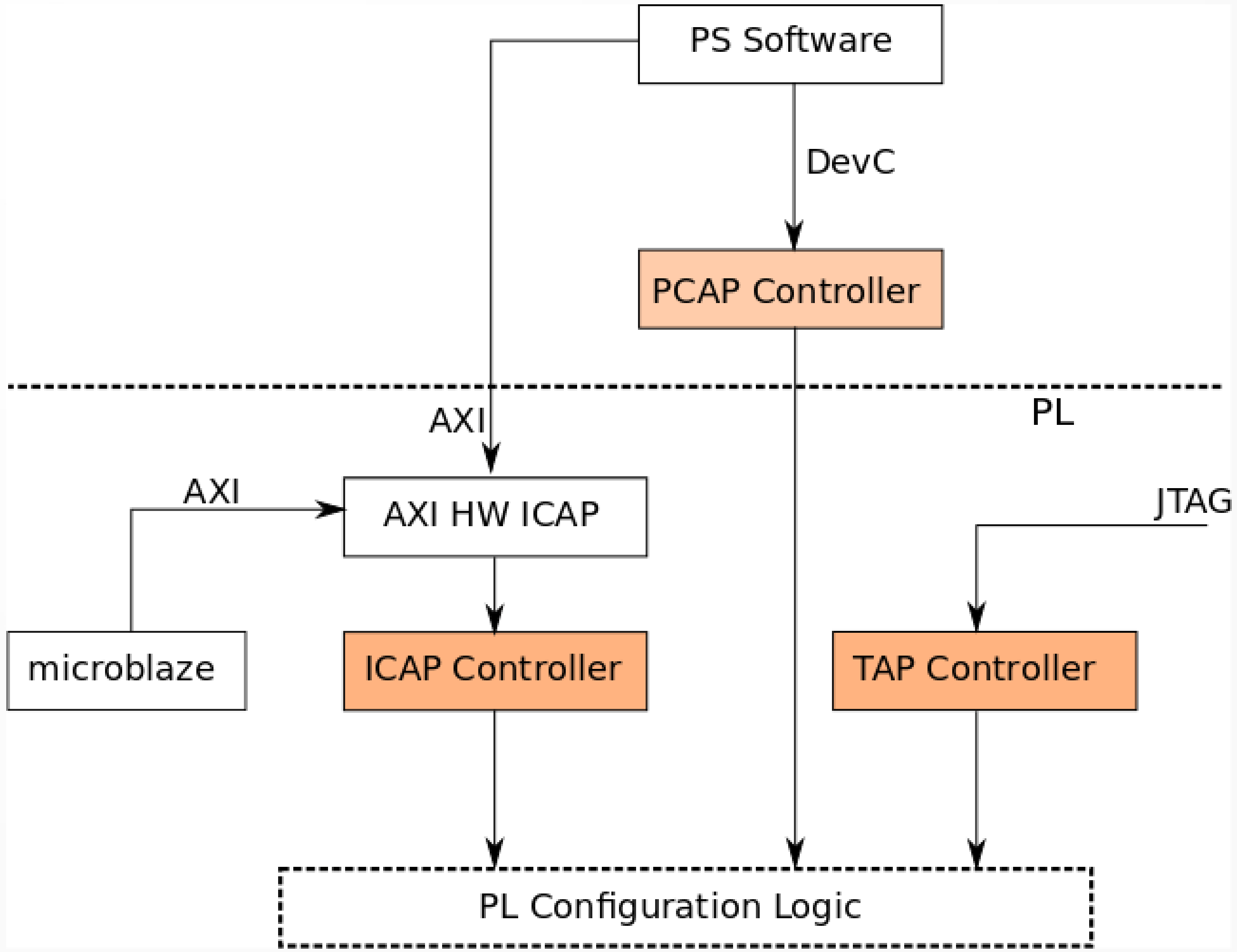
Stage	Description
Stage-0	On power-on reset, system reset or software reset, a hard-coded boot ROM is executed on the primary processor.
Stage-1	Typically this is the FSBL. It can, however, be any user-controlled code.
Stage-2	Typically this is the user design that will run on the processing system. It could also be the second-stage bootloader, and is completely within user control.

Boot sequence

JTAG Mode :

- Initialize cpu.
- Determine boot mode(JTAG).
- Configure JTAG chain.
- Configure PL using JTAG
- Upload PS software application to the OCM.
- The CPU start application execution.

Note: Internal Configuration Access Port (ICAP). enables a user to write software programs that can modify FPGA circuit structure and functionality during the operation of the circuit.

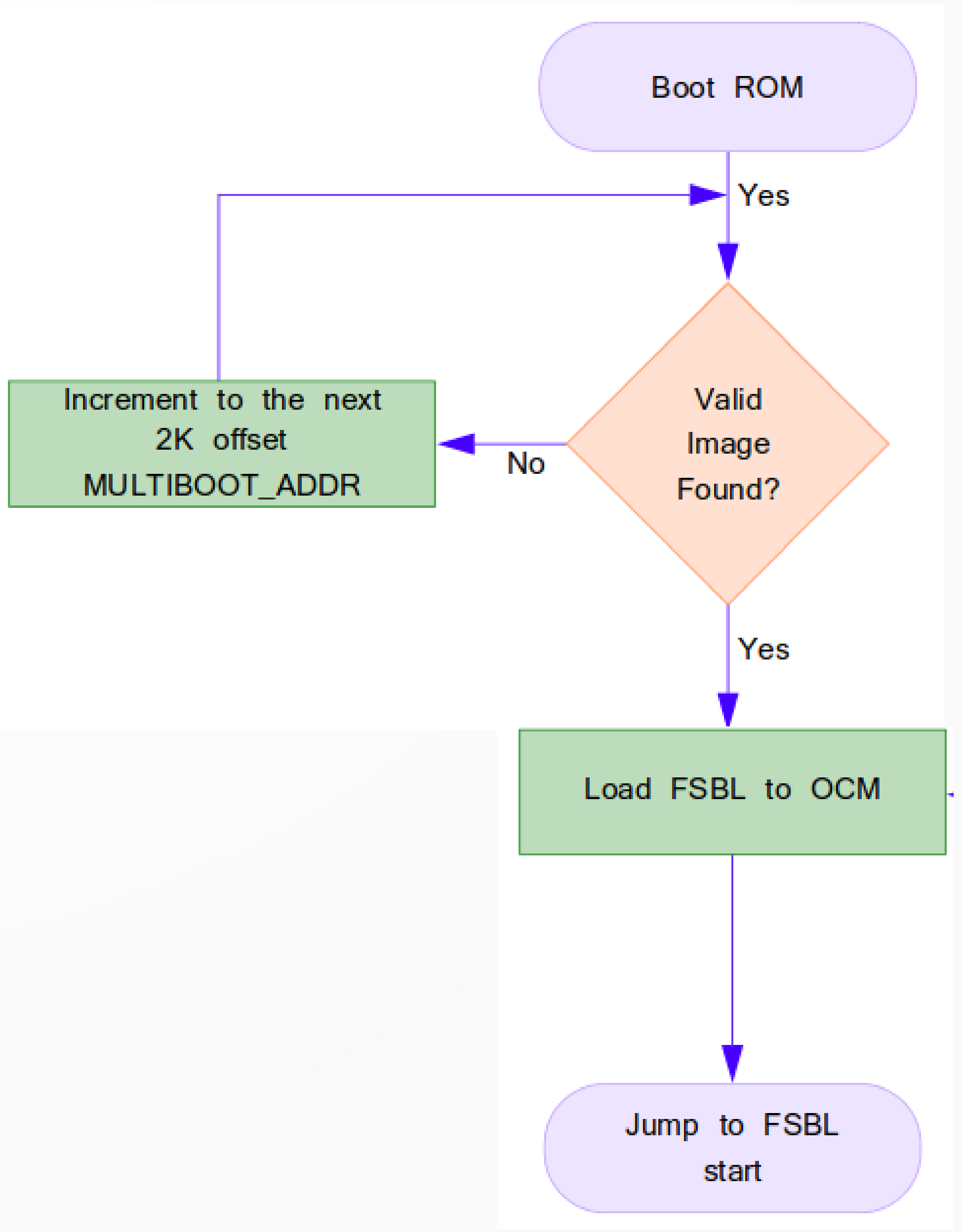


Boot sequence

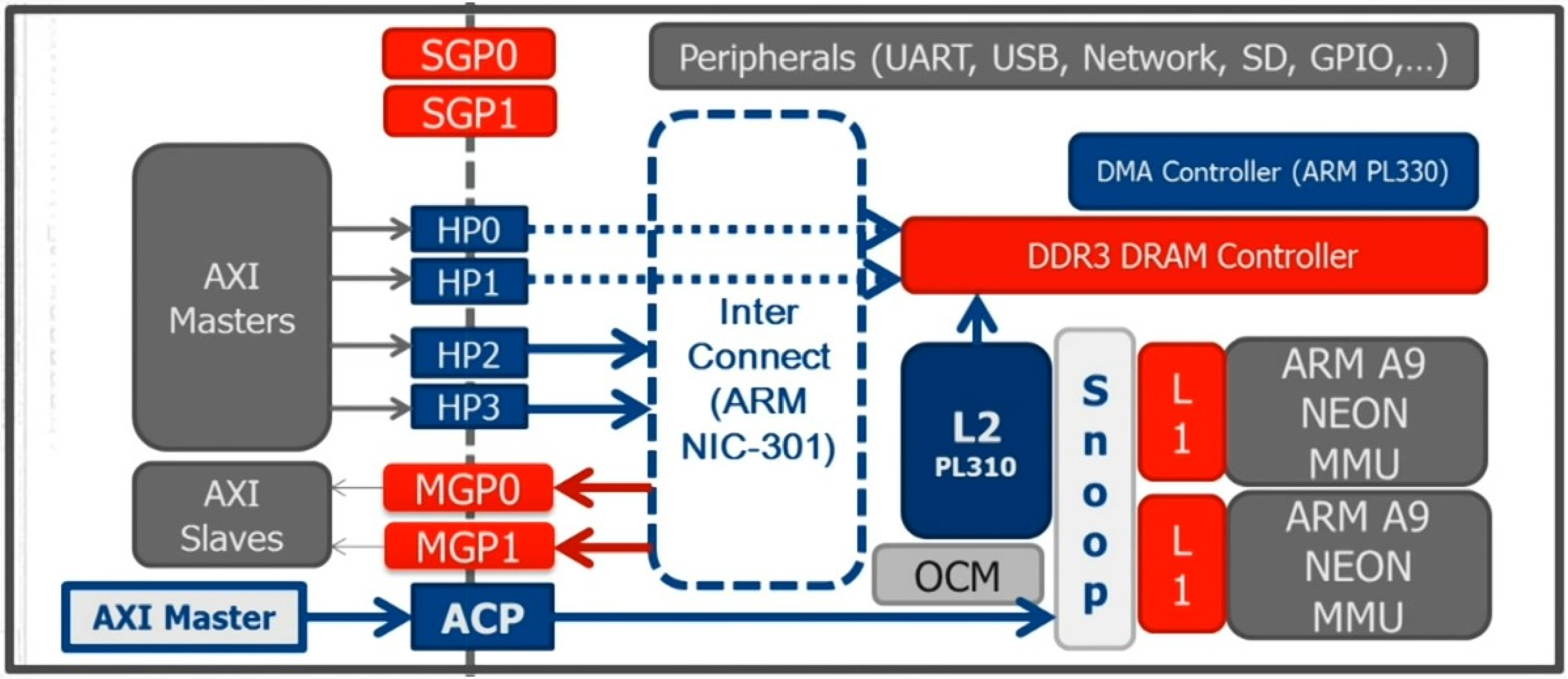
Two boot modes are available:

- Secured boot.
- Unsecured boot.

- The boot ROM provides support for loading both encrypted (secure) and unencrypted (non-secure) boot images. The boot ROM also supports execution of the stage-1 boot image directly from linear flash sources (QSPI) only when using the eXecute-In-Place (XIP) function. This feature is only available when using non-secure boot images.



Zynq


SGP0
SGP1
Peripherals (UART, USB, Network, SD, GPIO,...)
DMA Controller (ARM PL330)
AXI Masters
HP0
HP1
HP2
HP3
Inter Connect (ARM NIC-301)
DDR3 DRAM Controller
L2 PL310
Snoop
L1
ARM A9 NEON MMU
L1
ARM A9 NEON MMU
AXI Slaves
MGP0
MGP1
OCM
AXI Master
ACP

Thank you :)

Appendix

IO Peripherals

I2C

- I^{2}C bus specification version 2
- Programmable to use normal (7-bit) or extended (10-bit) addressing
- Programmable rates: fast mode (400 kbit/s) , standard (10 0kbits/s), and low (10 kbits/s)
  - Rates higher than 400 kbits/sec are not supported
- Programmable as either a master or slave interface
- Capable of clock synchronization and bus arbitration
- Fully programmable slave response address
- Reversible FIFO operation supported
- 16-byte buffer size
- Slave monitor mode when set up as master
- I^{2}C bus hold for slow host service
- Slave timeout detection with programmable period
- Transfer status interrupts and flags

CAN

- Up to 24-MHz CAN_REF clock as system clock
- 64 message-deep receiver and transmitter buffer
- Full CAN 2.0B compliant; conforms to ISO 11898-1
- Maximum baud rate of 1 Mb/s
- Four message filters required for buffer mode
- Listen-only mode for test and debug
- External PHY I/O
- “Wake-on-message”
- Time-stamping for receive messages
- TX and RX FIFO watermarking
- Exception: no power-down mode

SD-SDIO

- Support for version 2.0 of SD Specification
- Full-speed (4 MB/s) and low-speed (2 MB/s) support
  - Low-speed clock (400 KHz) used until bandwidth negotiated
- 1-bit and 4-bit data interface support
- Host mode support only
- Built-in DMA controller
- Full-speed clock (0-50 MHz) with maximum throughput at 25 MB/s
- 1 KB data FIFO interface
- Support for MMC 3.31 card at 52 MHz
- Support for memory, I/O, and combo cards
- Support for power control modes and interrupts

SPI

- Master or slave SPI mode
- Four wire bus: MOSI, MISO, SCK, nSS
- Supports up to three slave select lines
- Supports multi-master environment
- Identifies an error condition if more than one master detected
- Software can poll for status or function as interrupt-driven device
- Programmable interrupt generation
- 50-MHz maximum external SPI clock rate
- Selectable master clock reference
- Integrated 128-byte deep read and write FIFOs
- Full-duplex operation offers simultaneous receive and transmit

UART

- Two UARTs
- Programmable baud rate generator
- 64-byte receive and transmit FIFOs
- 6, 7, or 8 data bits and 1, 1.5, or 2 stop bits
- Odd, even, space, mark, or no parity with parity, framing, and overrun error detection
- "Line break" generation and detection
- Normal, automatic echo, local loopback, and remote loopback channel modes
- Interrupts generation
- Support 8 Mb/s maximum baud rate with additional reference clock or up to 1.5 Mb/s with a 100-MHz peripheral bus clock
- Modem control signals: CTS, RTS, DSR, DTR, RI, and DCD (through EMIO)
- Simple UART: only two pins (TX and RX through MIO)

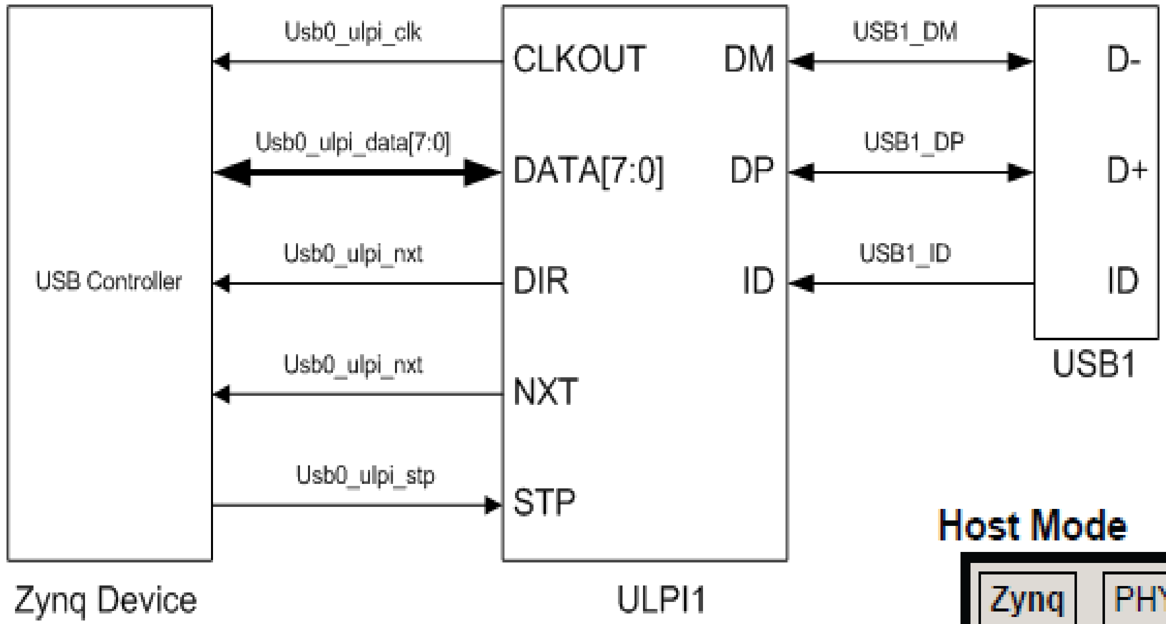
USB

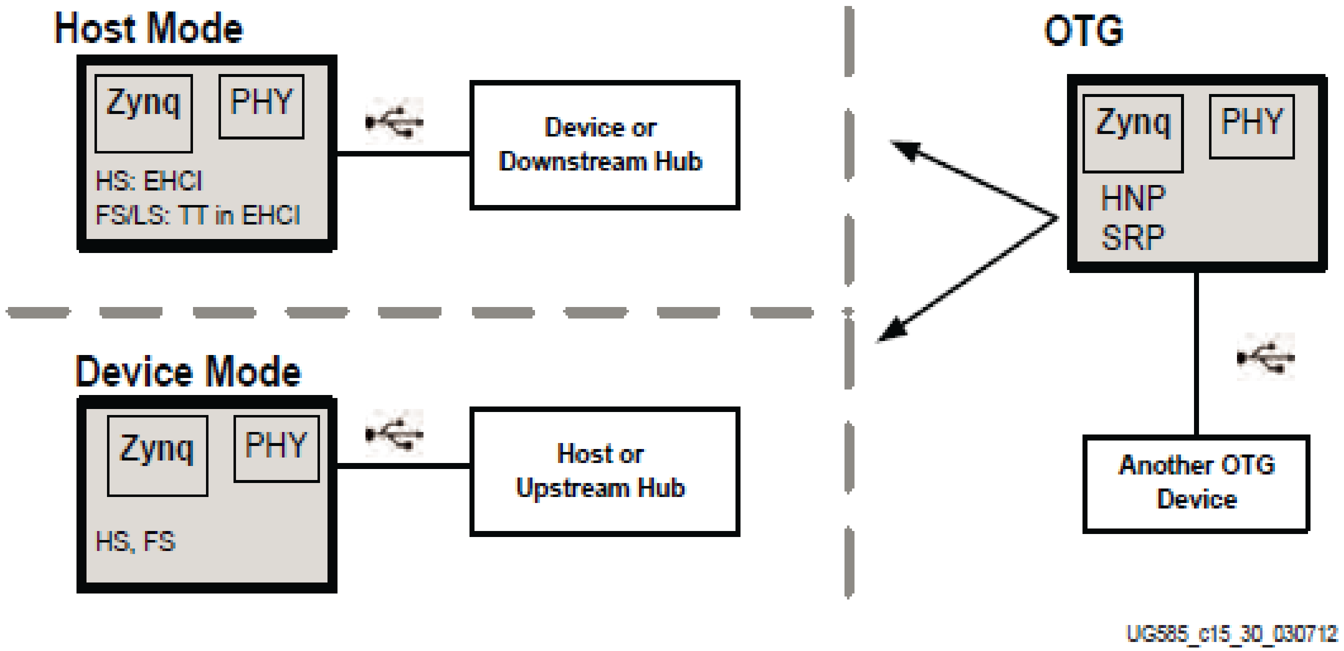
- Two USB 2.0 hardened IP peripherals per Zynq device
  - Each independently controlled and configured
- Supported interfaces
  - High-speed USB 2.0: 480 Mbit/s
  - Full-speed USB 1.1: 12 Mbit/s
  - Low-speed USB 1.0: 1.5 Mbit/s
  - Communication starts at USB 2.0 speed and drops until sync is achieved
- Each block can be configured as host, device, or on-the-go (OTG)
- 8-bit ULPI interface
- All four transfer types supported: isochronous, interrupt, bulk, and control
- Supports up to 12 endpoints per USB block in the Zynq device
  - Running in host mode
- Source-code drivers

USB 2.0 OTG

- Control and configuration registers for each USB block
- Software-ready with standalone and OS linux source-code delivered drivers
- EHCI compliant host registers
- USB host controller registers and data structures compliant to Intel EHCI specifications
- Internal DMA
- Must use the MIO pins

USB 2.0 Usage Example

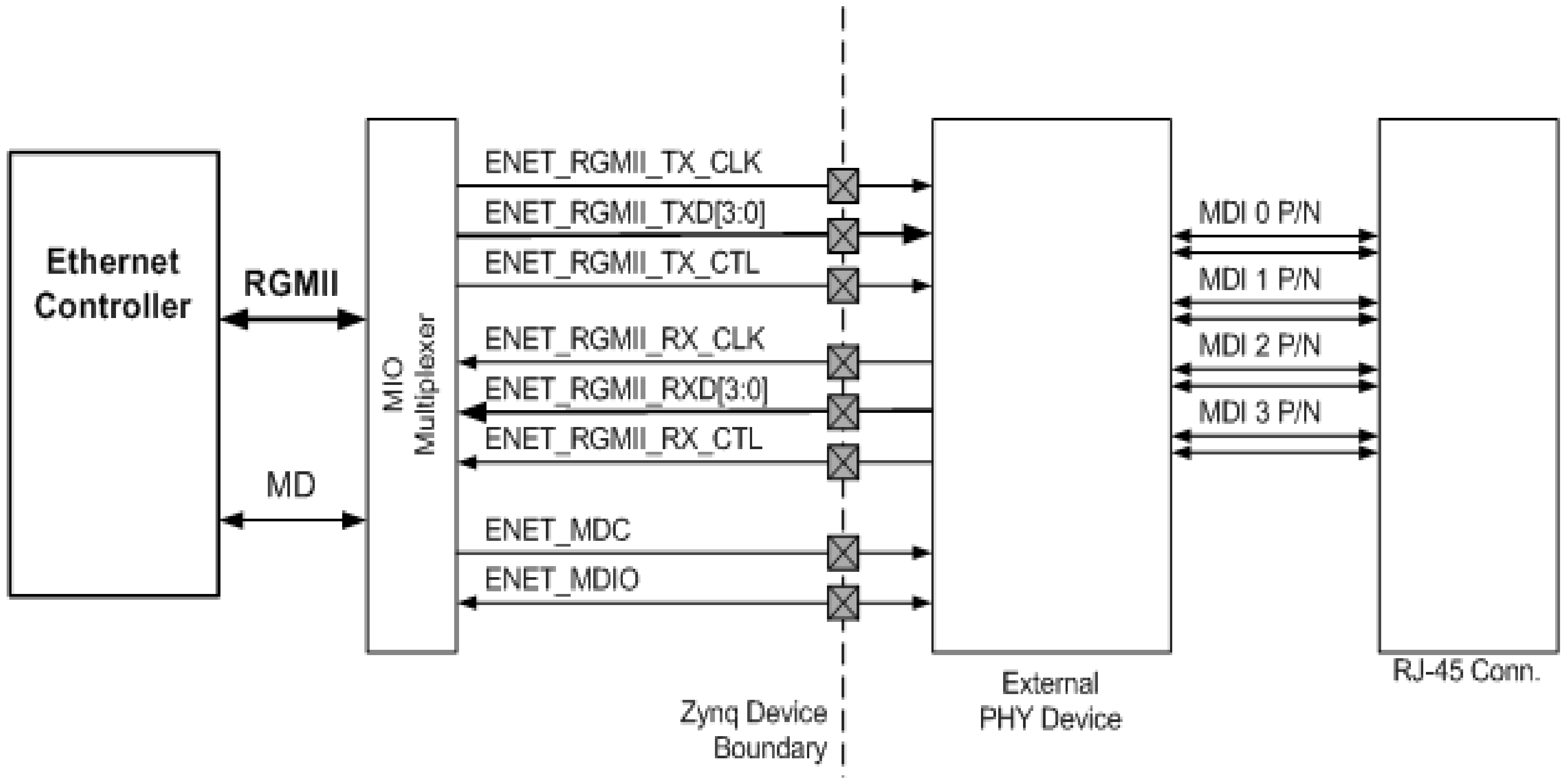




Gigabit Ethernet Controller

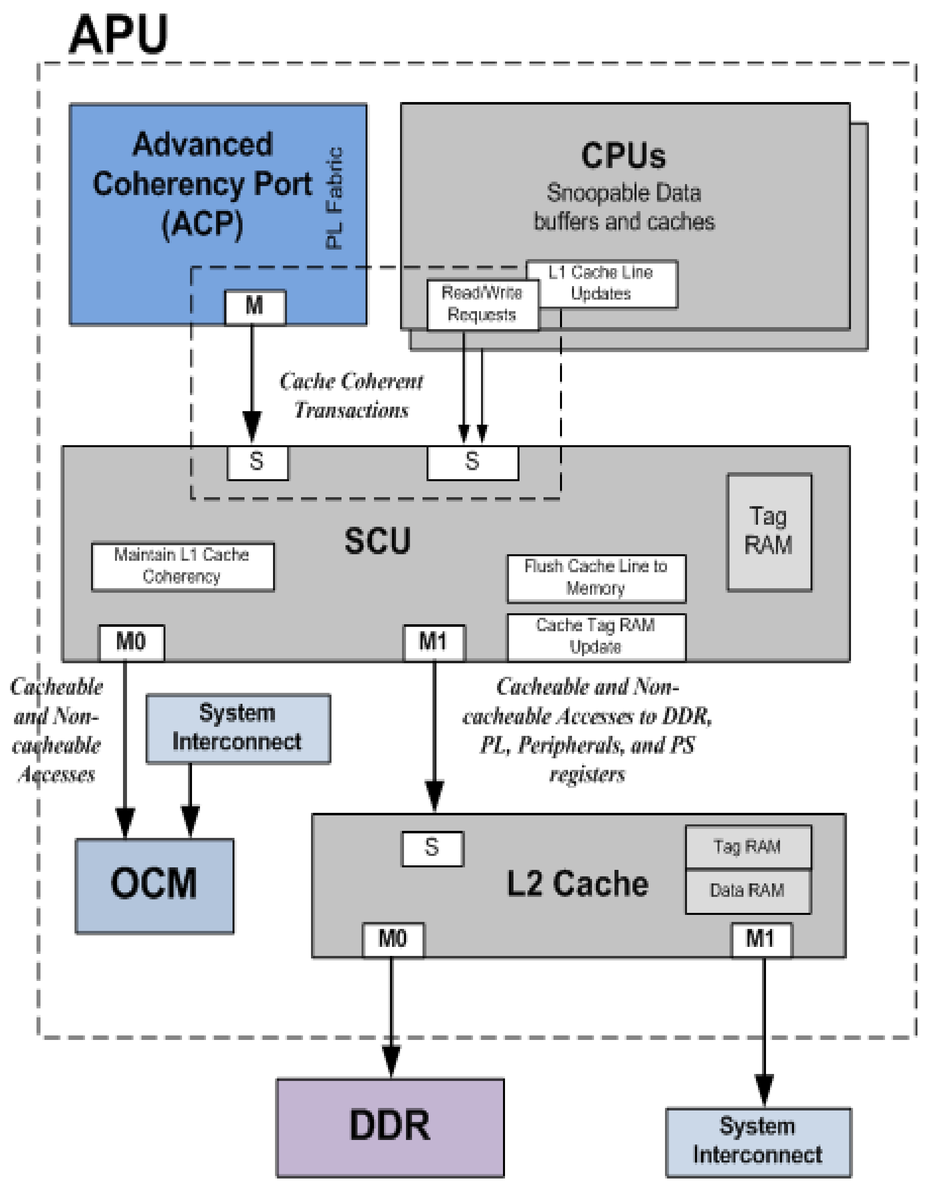
- Tri-mode Ethernet MAC (10/100/1G) with native GMII interface
- IEEE1588 rev 2.0
  - Time stamp support
  - 1 us resolution
- IEEE802.3
- RGMII v2.0 (HSTL) interface to MIO pins
  - Need MIO set at 1.8V to support RGMII speed
  - Need to use large bank of MIO pins for two Ethernets
- MII/GMII/SGMII/RGMII ver1.3 (LVCMOS) and ver2.0 (HSTL) interface available through EMIO (programmable logic I/O)
- TX/RX checksum offload for TCP and UDP
- Internal DMA and wake on LAN

Gigabit Ethernet Controller



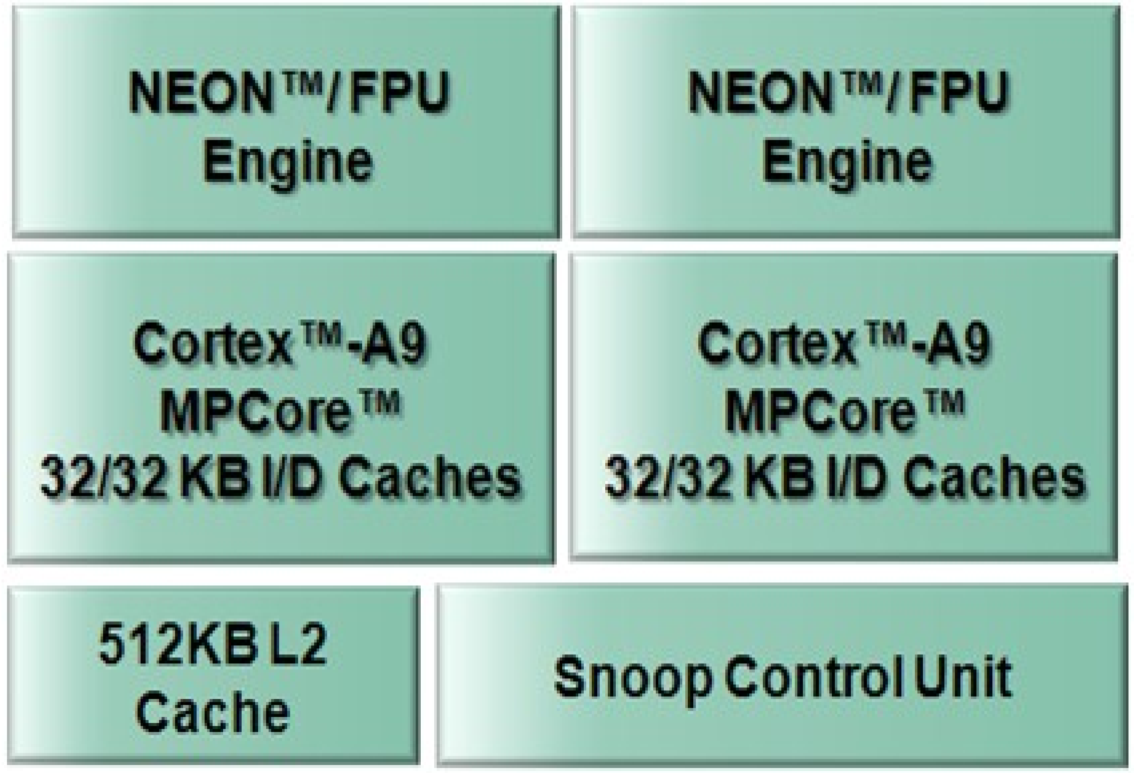
Application Processor Unit (APU)

APU


APU
Advanced Coherency Port (ACP)
PL Fabric
M
CPUs
Snoopable Data buffers and caches
L1 Cache Line Updates
Read/Write Requests
Cache Coherent Transactions
S
S
SCU
Tag RAM
Maintain L1 Cache Coherency
Flush Cache Line to Memory
Cache Tag RAM Update
M0
M1
Cacheable and Non-cacheable Accesses
System Interconnect
Cacheable and Non-cacheable Accesses to DDR, PL, Peripherals, and PS registers
OCM
S
L2 Cache
Tag RAM
Data RAM
M0
M1
DDR
System Interconnect

APU Components

- Dual ARM® Cortex™-A9 MPCore with NEON extensions
  - Up to 800-MHz operation
  - 2.5 DMIPS/MHz per core
  - Separate 32KB instruction and data caches
- Snoop Control Unit (SCU)
  - L1 cache snoop control
    - Accelerator coherency port
- Level 2 cache and controller
  - Shared 512 KB cache with parity


NEON™/FPU Engine
NEON™/FPU Engine
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
512KB L2 Cache
Snoop Control Unit

APU Sub-Components

- General interrupt controller (GIC)
- On-chip memory (OCM): RAM and boot ROM
- Central DMA (eight channels)
- Device configuration (DEVCFG)
- Private watchdog timer and timer for each CPU
- System watchdog and triple timer counters shared between CPUs
- ARM CoreSight debug technology

APU Address Map

- All registers for both CPUs are grouped into two contiguous 4KB pages
  - Accessed through a dedicated internal bus
- Fixed at 0xF8F0_0000 with a register block size of 8 KB
  - Each CPU uses an offset into this base address

0x0000-0x00FC	SCU registers
0x0100-0x01FF	Interrupt controller interface
0x0200-0x02FF	Global timer
0x0600-0x06FF	Private timers and watchdog timers
0x1000-0x1FFF	Interrupt distributor

NEON Main Features

- NEON is the ARM codename for the vector processing unit
  - Provides multimedia and signal processing support
- FPU is the floating-point unit extension to NEON
  - Both NEON and FPU share a single set of registers
- NEON technology is a wide single instruction, multiple data (SIMD) parallel and co-processing architecture
  - 32 registers, 64-bits wide (dual view as 16 registers, 128-bits wide)
  - Data types can be: signed/unsigned 8-bit, 16-bit, 32-bit, 64-bit, or 32-bit float

L1 Cache Features

- Separate instruction and data caches for each processor
- Caches are four-way, set associative and are write-back
- Non-lockable
- Eight words cache length
- On a cache miss, critical word first filling of the cache is performed followed by the next word in sequence

L2 Cache Features

- 512K bytes of RAM built into the SCU
  - Latency of 25 CPU cycles
  - Unified instruction and data cache
- Fixed, 256-bit (32 words) cache line size
- Support for per-master way lockdown between multiple CPUs
- Eight-way, set associative
- Two AXI interfaces
  - One to DDR controller
  - One to programmable logic master (to peripherals)

On-Chip Memory (OCM)

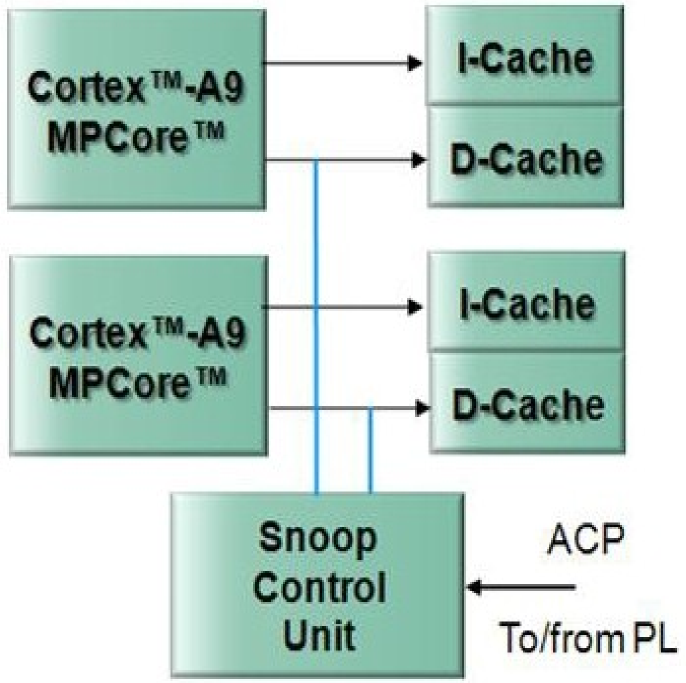
- The on-chip memory (OCM) module contains 256 KB of RAM and 128 KB of ROM (BootROM).
- It supports two 64-bit AXI slave interface ports, one dedicated for CPU/ACP access via the APU snoop control unit (SCU), and the other shared by all other bus masters within the processing system (PS) and programmable logic (PL).
- The BootROM memory is used exclusively by the boot process and is not visible to the user.

Snoop Control Unit (SCU)

- Shares and arbitrates functions between the two processor cores
  - Data cache coherency between the processors
  - Initiates L2 AXI memory access
  - Arbitrates between the processors requesting L2 accesses
  - Manages ACP accesses
  - A second master port with programmable address filtering between OCM and L2 memory support

Cache Coherency Using SCU

- High-performance, cache-to-cache transfers
- Snoop each CPU and cache each interface independently
- Coherency protocol is MESI
  - M: Cache line has been modified
  - E: Cache line is held exclusively
  - S: Cache line is shared with another CPU
  - I: Cache line is invalidated
- Uses Accelerator Coherence Port (ACP) to allow coherency to be extended to PL



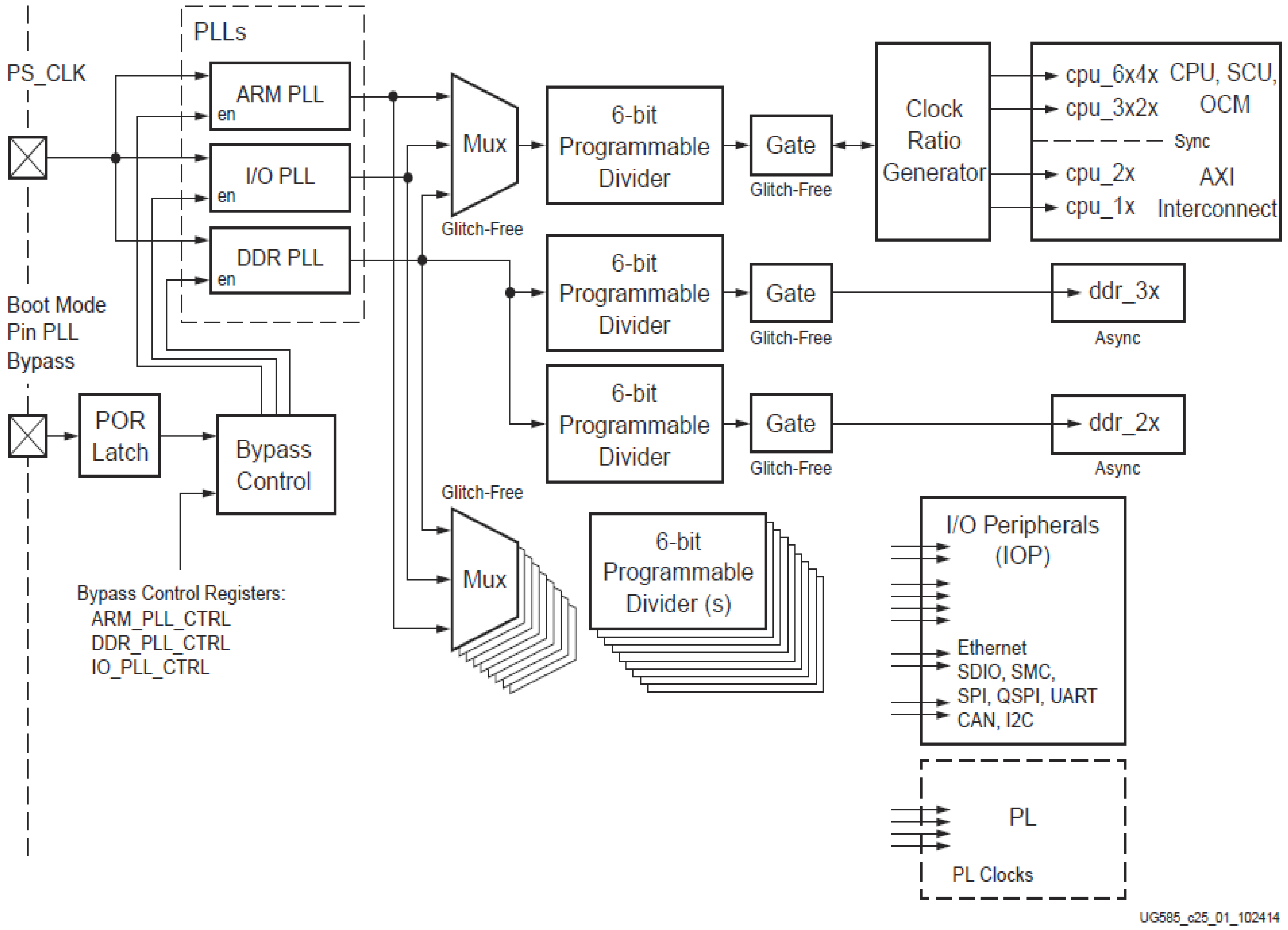
System Level Control Register (SLCR)

- A set of of special registers in the APU used to configure the PS
  - Power and clock management
  - Reset control
  - MIO/EMIO management
- Accessible through software
  - Standalone BSP support

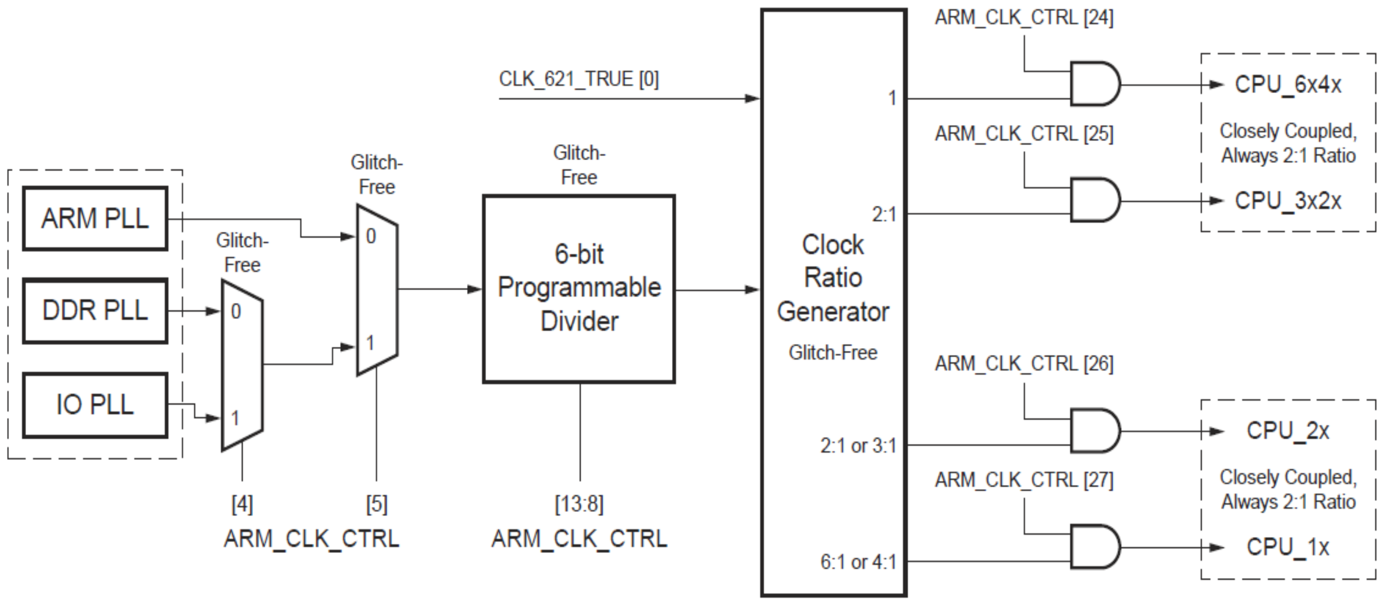
SLCR Categories	
System clock and reset control/status registers	TrustZone control register
APU control registers	SoC debug control registers
DMA initialization registers	MIO/IOP control/status registers
DDR control registers	Miscellaneous control registers
PL reset registers	RAM and ROM control registers

Zynq Clocks

System Clocks

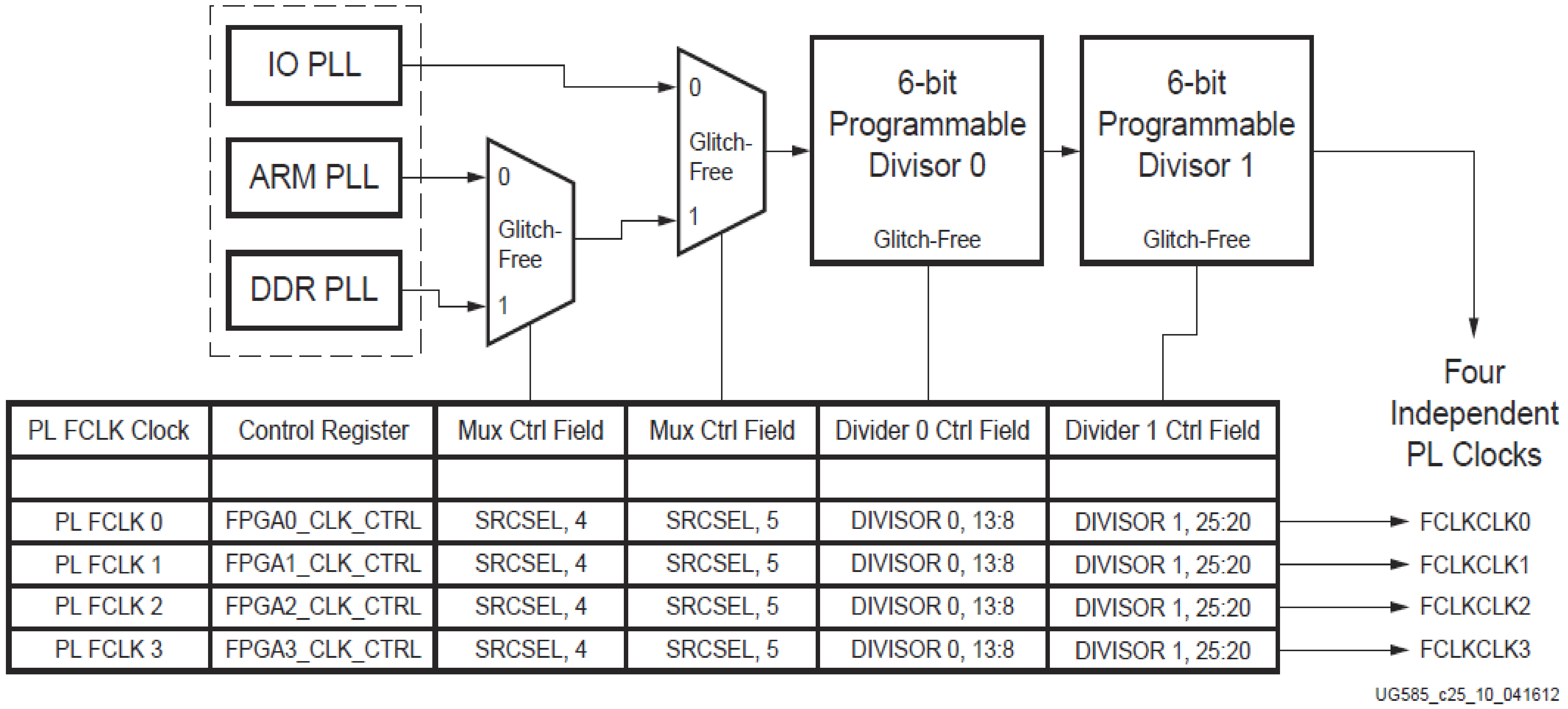


CPU Clock



CPU Clock	6:2:1	4:2:1	Clock Domain Modules
CPU_6x4x	800 MHz (6 times faster than CPU_1x)	600 MHz (4 times faster than CPU_1x)	CPU clock freq, SCU, OCM arbitrer, NEON and L2 Cache
CPU_3x2x	400 MHz (3 times faster than CPU_1x)	300 MHz (2 times faster than CPU_1x)	APU Timers
CPU_2x	266MHz (2 times faster than CPU_1x)	300 MHz (2 times faster than CPU_1x)	IOP, central interconnect, master interconeect, slave interconnect and OCM RAM
CPU_1x	133 MHz	150 MHz	IOP, AHB and APB interface busses

PL Clocks



PL FCLK Clock	Control Register	Mux Ctrl Field	Mux Ctrl Field	Divider 0 Ctrl Field	Divider 1 Ctrl Field
PL FCLK 0	FPGA0_CLK_CTRL	SRCSEL, 4	SRCSEL, 5	DIVISOR 0, 13:8	DIVISOR 1, 25:20
PL FCLK 1	FPGA1_CLK_CTRL	SRCSEL, 4	SRCSEL, 5	DIVISOR 0, 13:8	DIVISOR 1, 25:20
PL FCLK 2	FPGA2_CLK_CTRL	SRCSEL, 4	SRCSEL, 5	DIVISOR 0, 13:8	DIVISOR 1, 25:20
PL FCLK 3	FPGA3_CLK_CTRL	SRCSEL, 4	SRCSEL, 5	DIVISOR 0, 13:8	DIVISOR 1, 25:20