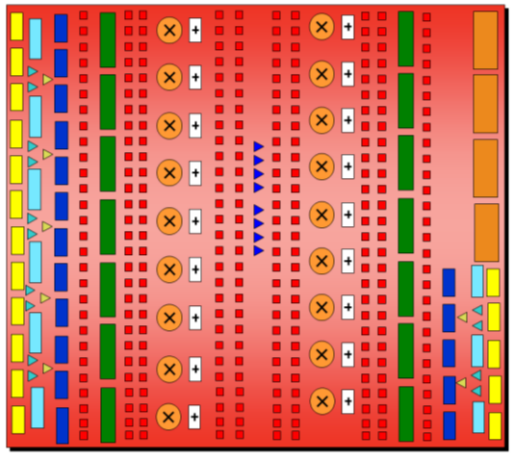
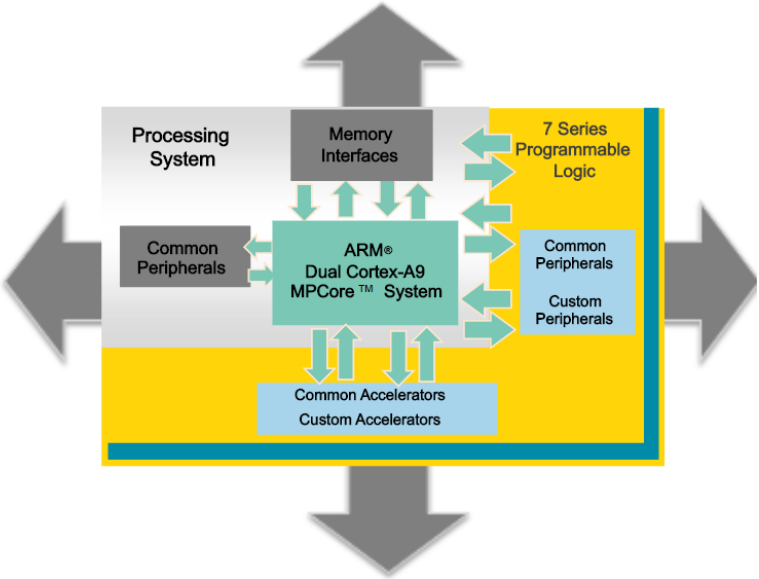
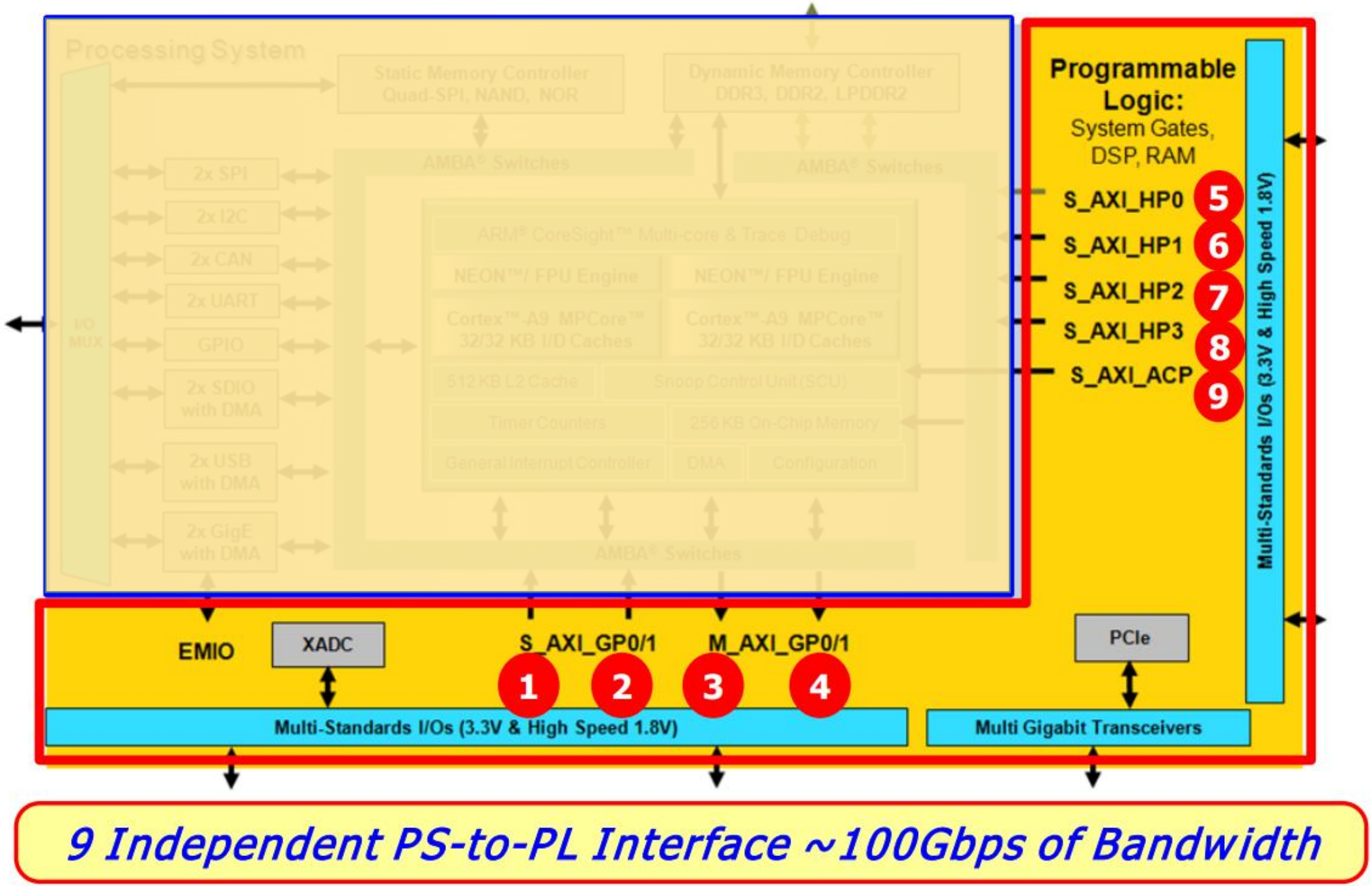
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

A2A
Algorithms to Architecture

ECE 270: Embedded Logic Design

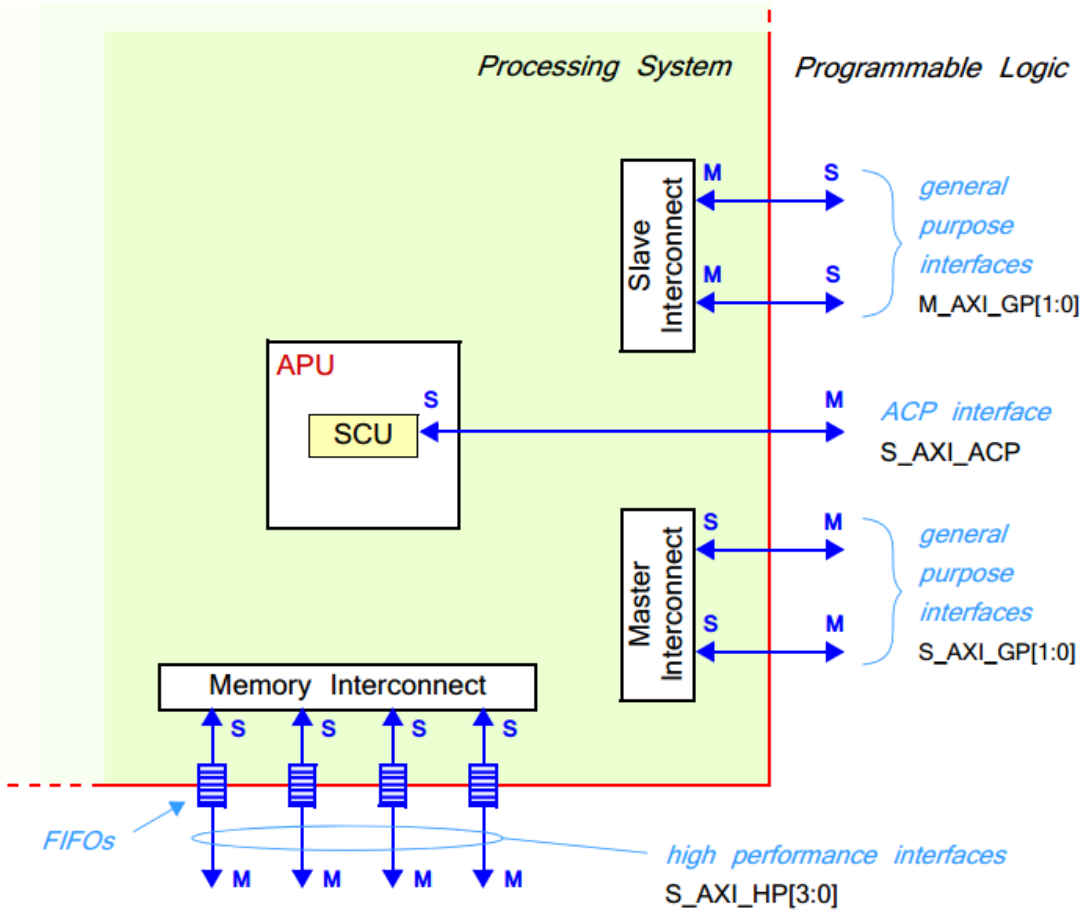




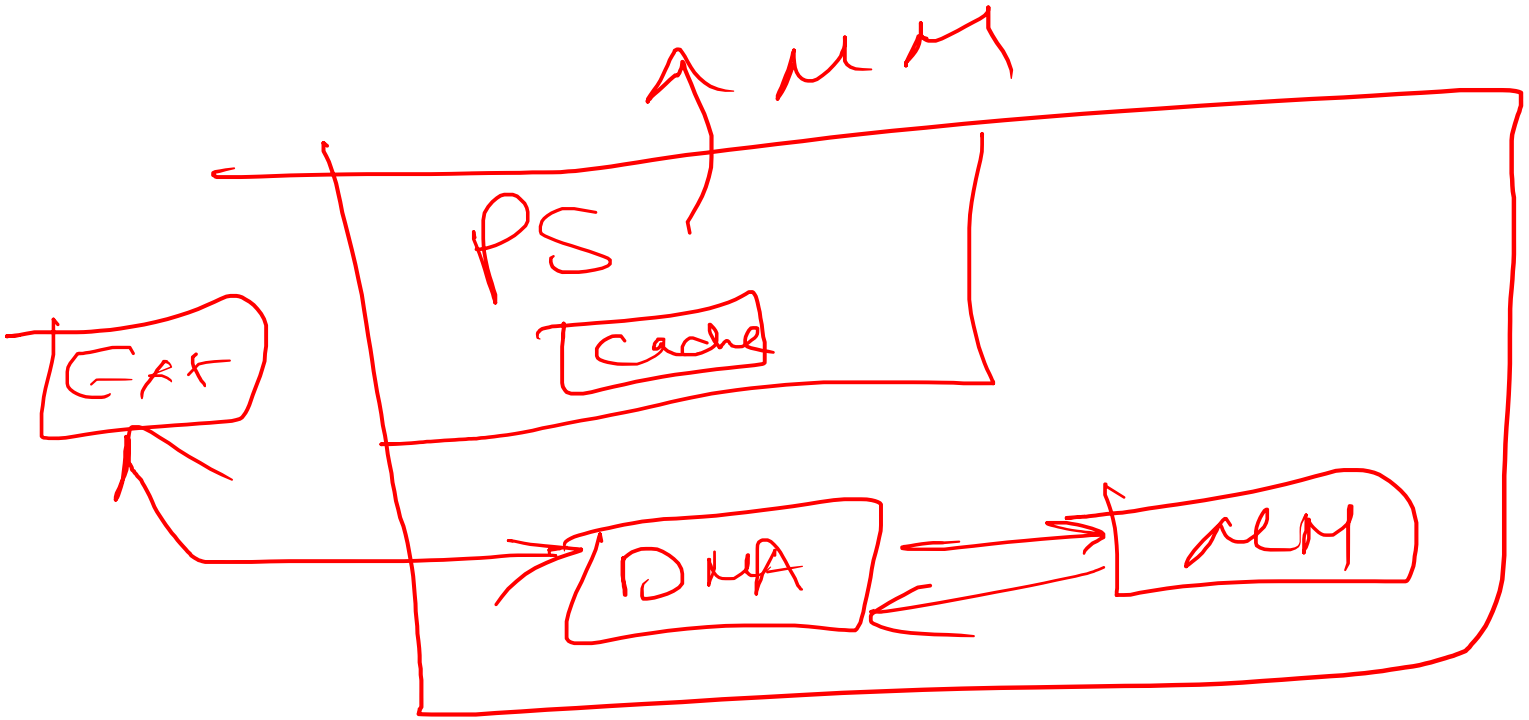


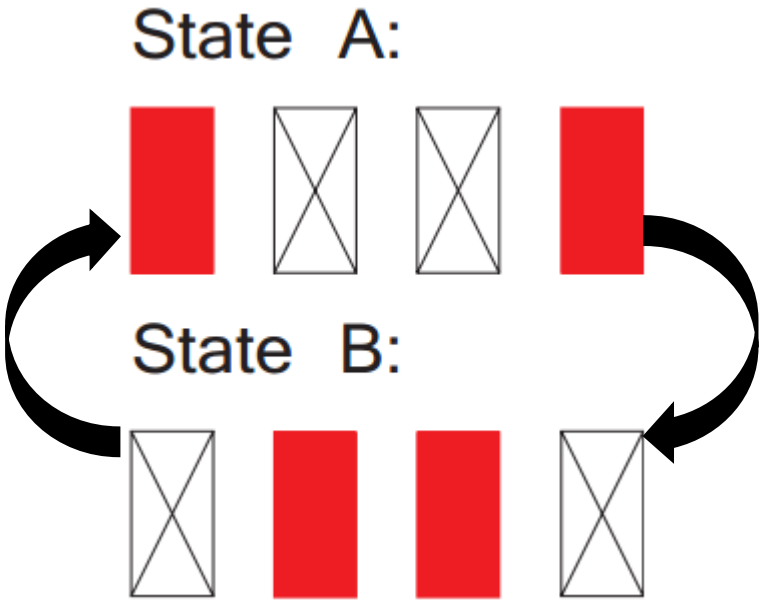
9 Independent PS-to-PL Interface ~100Gbps of Bandwidth

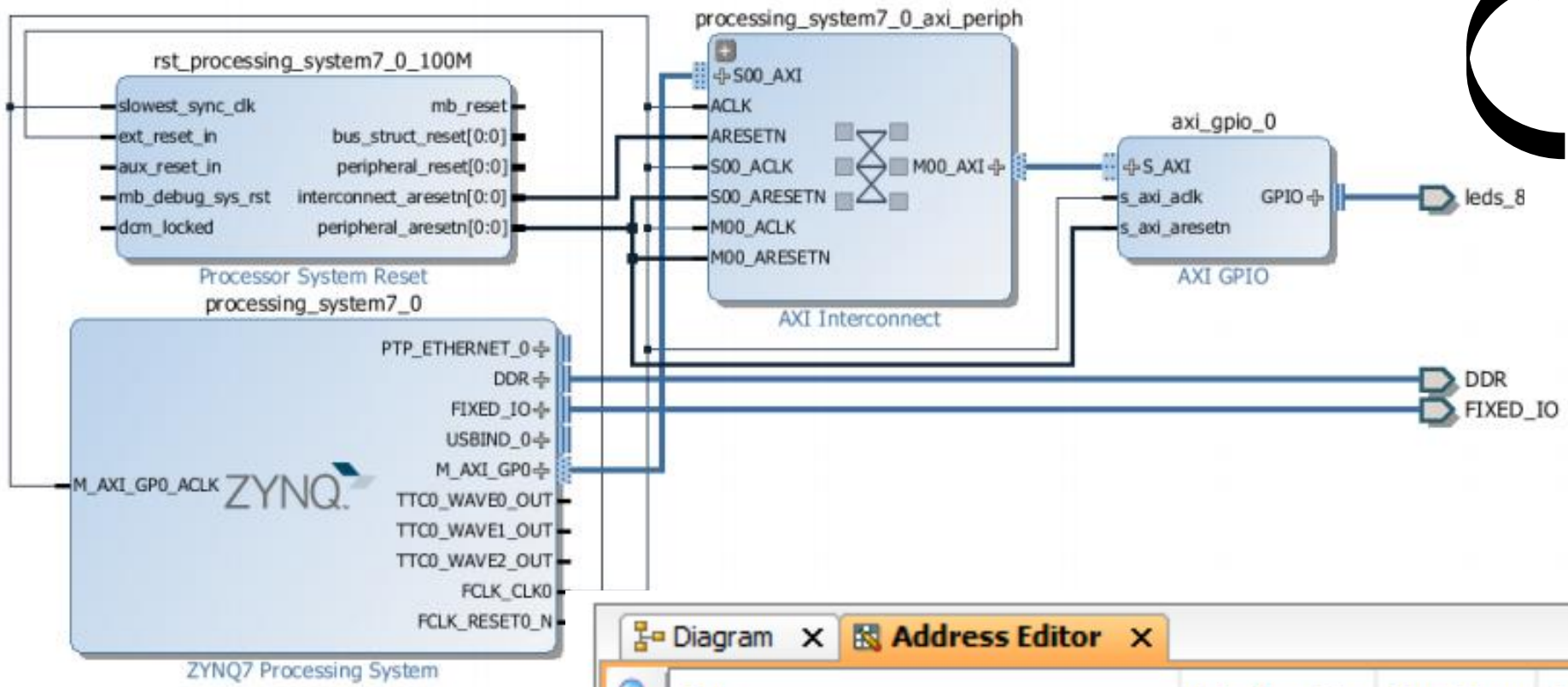
Zynq PS-PL Interface

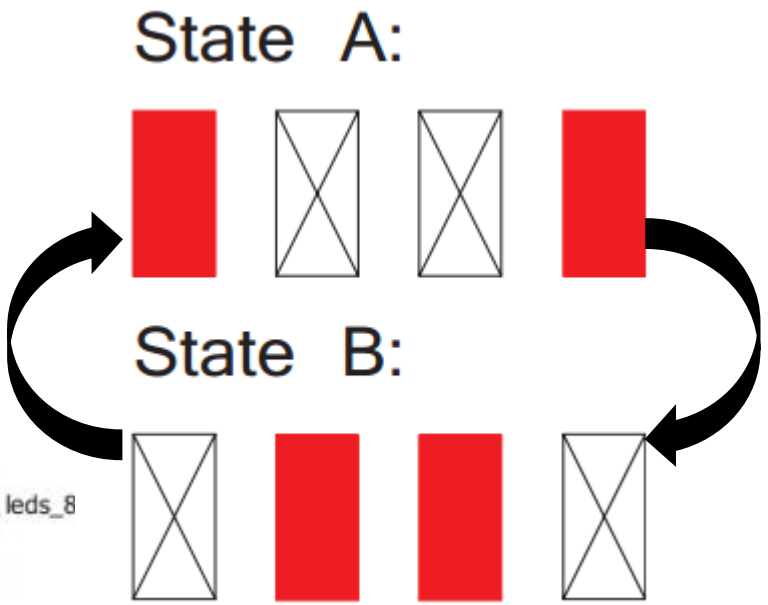


Interface Name	Interface Description	Master	Slave
M_AXI_GP0	General Purpose (AXI_GP)	PS	PL
M_AXI_GP1		PS	PL
S_AXI_GP0	General Purpose (AXI_GP)	PL	PS
S_AXI_GP1		PL	PS
S_AXI_ACP	Accelerator Coherency Port, cache-coherent transaction (ACP)	PL	PS
S_AXI_HP0	High Performance ports (AXI_HP) with read/write FIFOs and two dedicated memory ports on DDR controller and a path to the OCM. The AXI_HP interfaces are known also as AFI.	PL	PS
S_AXI_HP1		PL	PS
S_AXI_HP2		PL	PS
S_AXI_HP3		PL	PS


PS
Cache
DMA


State A:
State B:


rst_processing_system7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
processing_system7_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M00_AXI
AXI Interconnect
axi_gpio_0
S_AXI
s_axi_aclk
s_axi_aresetn
GPIO
AXI GPIO
leds_8
processing_system7_0
PTP_ETHERNET_0
DDR
FIXED_IO
USBIND_0
M_AXI_GP0
M_AXI_GP0_ACLK
TTC0_WAVE0_OUT
TTC0_WAVE1_OUT
TTC0_WAVE2_OUT
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
DDR
FIXED_IO


State A:
State B:

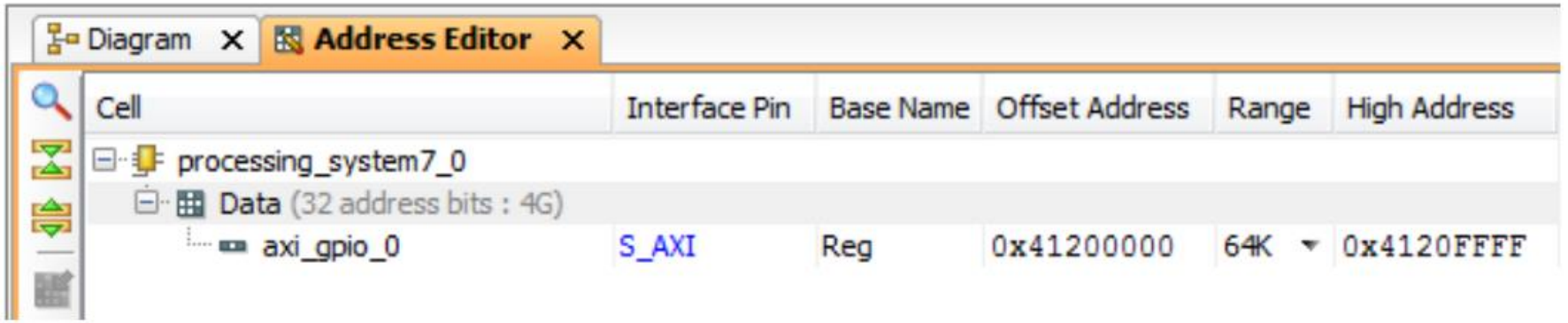


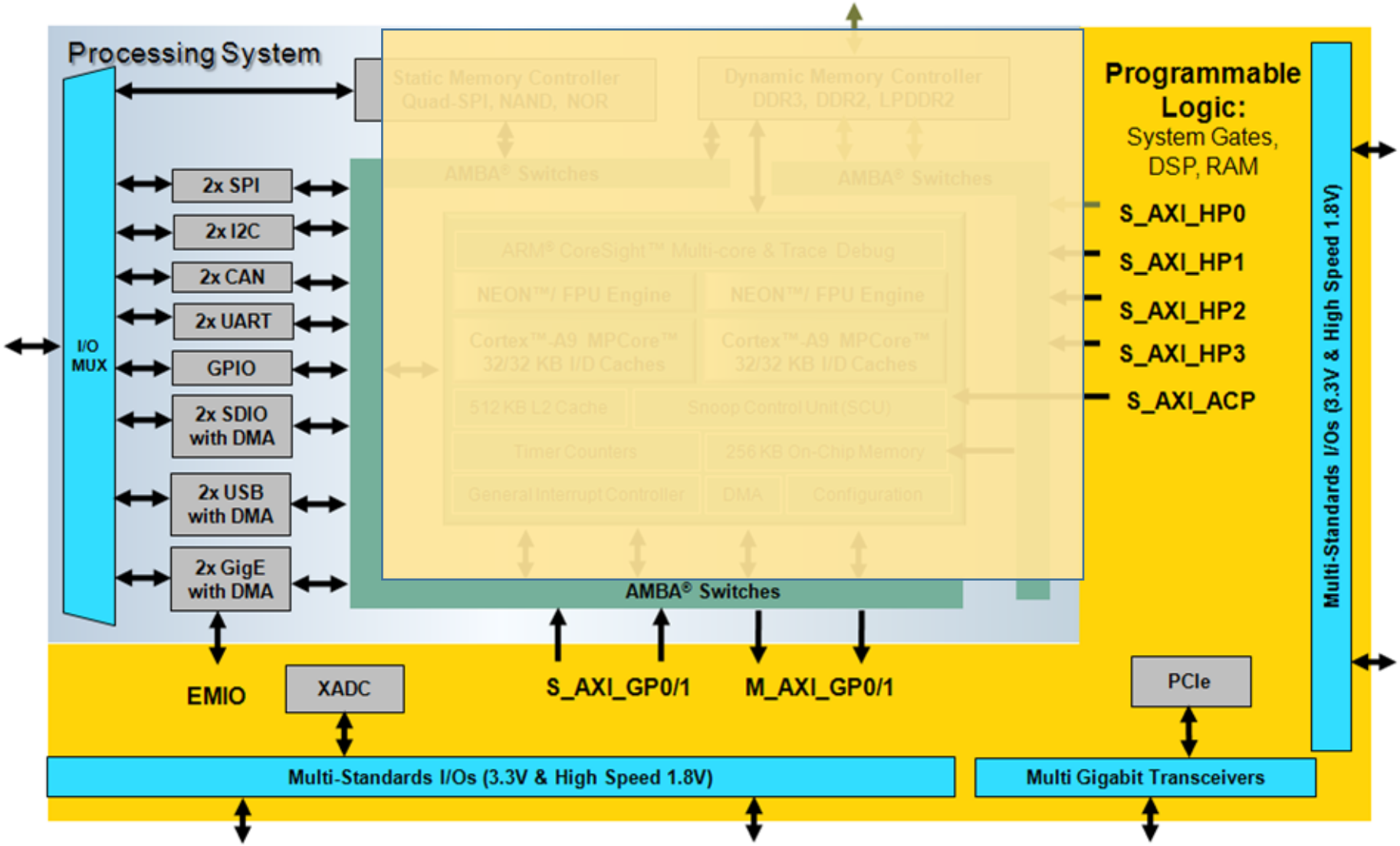
Diagram
Address Editor
Cell
Interface Pin
Base Name
Offset Address
Range
High Address
processing_system7_0
Data (32 address bits : 4G)
axi_gpio_0
S_AXI
Reg
0x41200000
64K
0x4120FFFF

C-Code

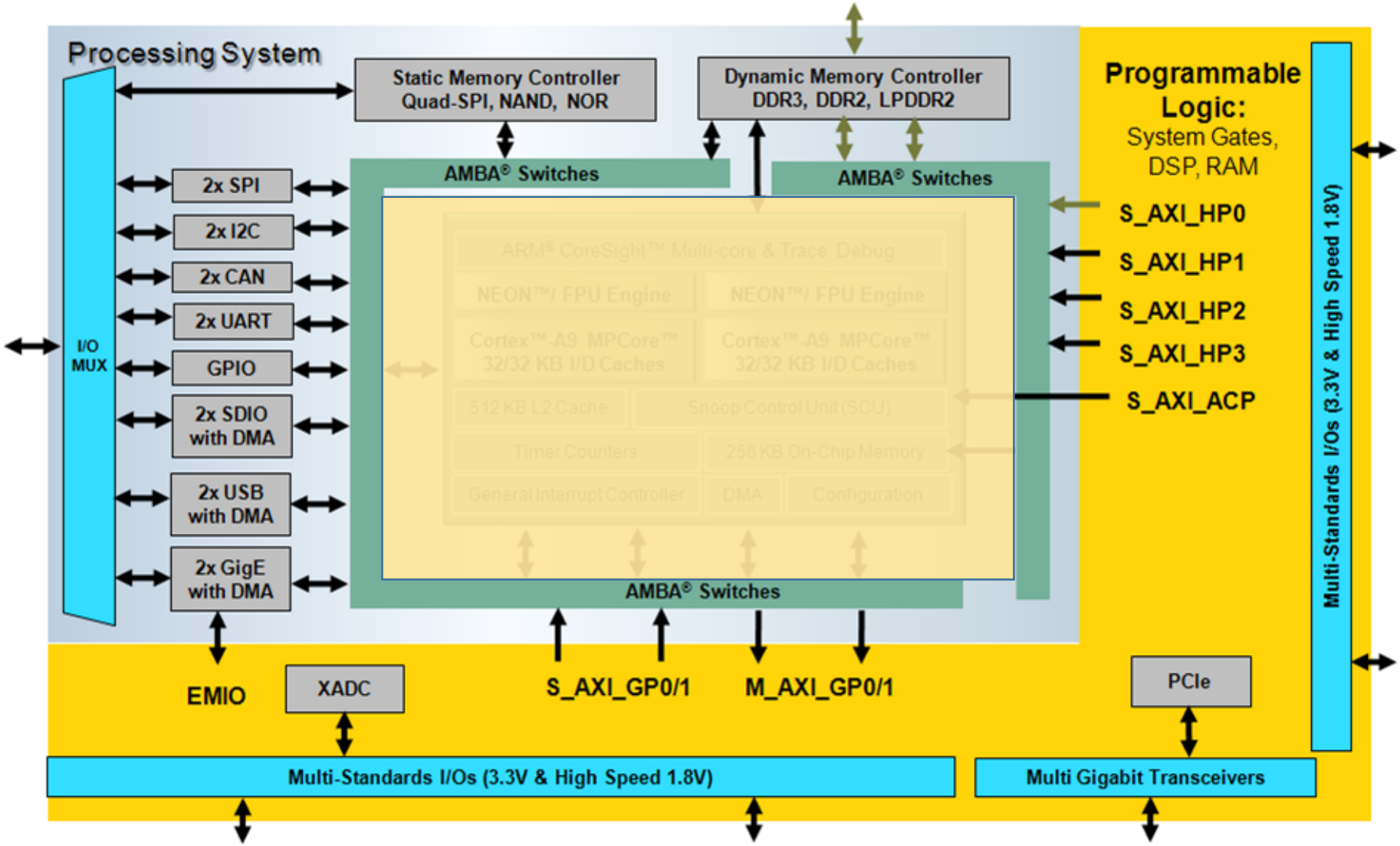
```c
/* Include Files */
#include "xparameters.h"
#include "xgpio.h"
#include "xstatus.h"
#include "xil_printf.h"

/* Definitions */
#define GPIO_DEVICE_ID  XPAR_AXI_GPIO_0_DEVICE_ID /* GPIO device that LEDs are connected to */
#define LED 0x9                                   /* Initial LED value - X00X */
#define LED_DELAY 10000000                        /* Software delay length */
#define LED_CHANNEL 1                             /* GPIO port for LEDs */
#define printf xil_printf                         /* smaller, optimised printf */

XGpio Gpio;

/* Main function. */
int main(void){

    int Status;

    /* Execute the LED output. */
    Status = LEDOutputExample();
    if (Status != XST_SUCCESS) {
        xil_printf("GPIO output to the LEDs failed!\r\n");
    }

    return 0;
}
```

```c
/* GPIO Device driver instance */
int LEDOutputExample(void)
{
    volatile int Delay;
    int Status;
    int led = LED; /* Hold current LED value. Initialise to LED definition */
        /* GPIO driver initialisation */
        Status = XGpio_Initialize(&Gpio, GPIO_DEVICE_ID);
        if (Status != XST_SUCCESS) {
            return XST_FAILURE;
        }
        /*Set the direction for the LEDs to output. */
        XGpio_SetDataDirection(&Gpio, LED_CHANNEL, 0x0);
        /* Loop forever blinking the LED. */
            while (1) {
                /* Write output to the LEDs. */
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, led);
                /* Flip LEDs. */
                led = ~led;
                /* Wait a small amount of time so that the LED blinking is visible. */
                for (Delay = 0; Delay < LED_DELAY; Delay++);
            }
        return XST_SUCCESS; /* Should be unreachable */
}
```

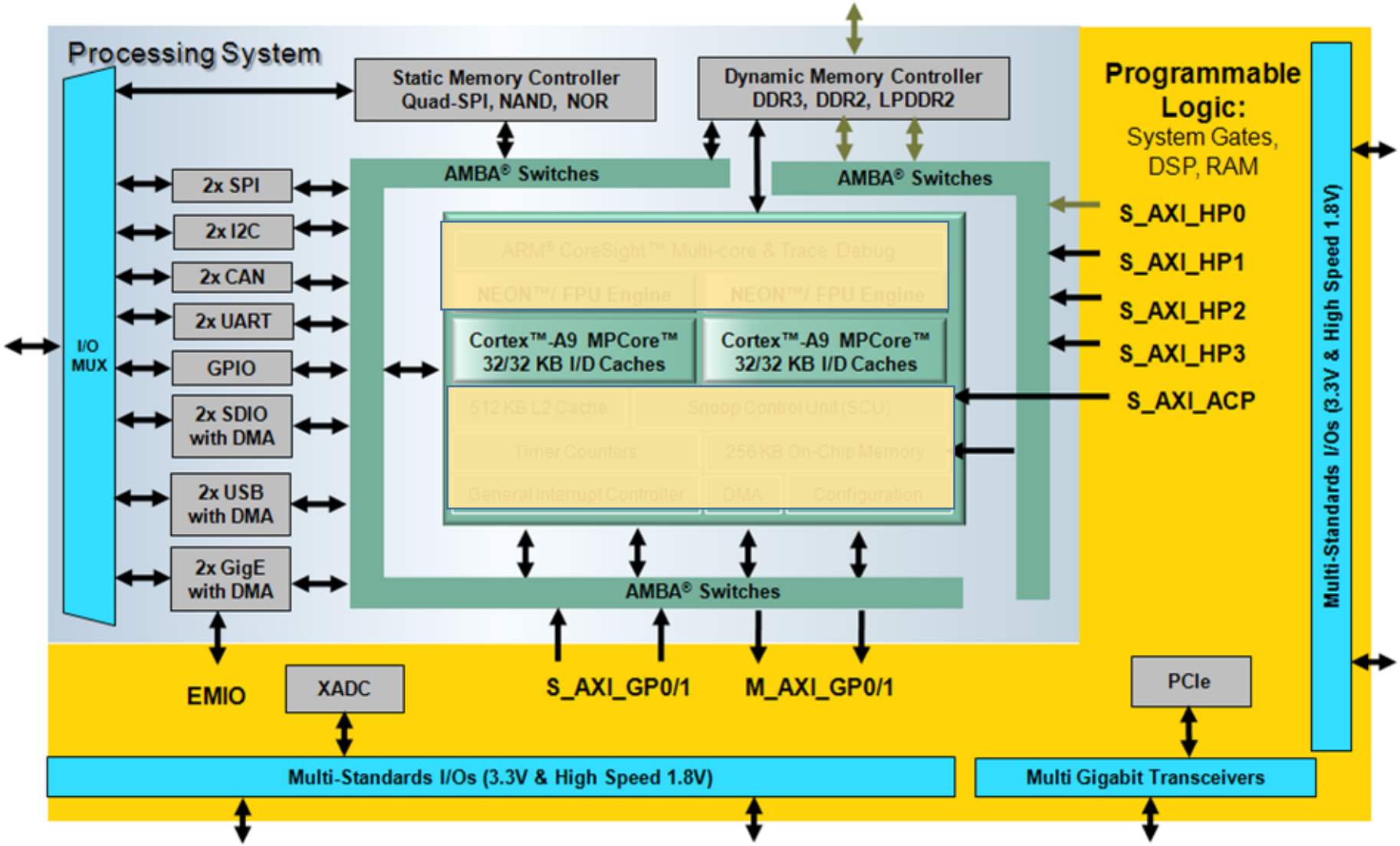
Zynq Architecture: PS and PL



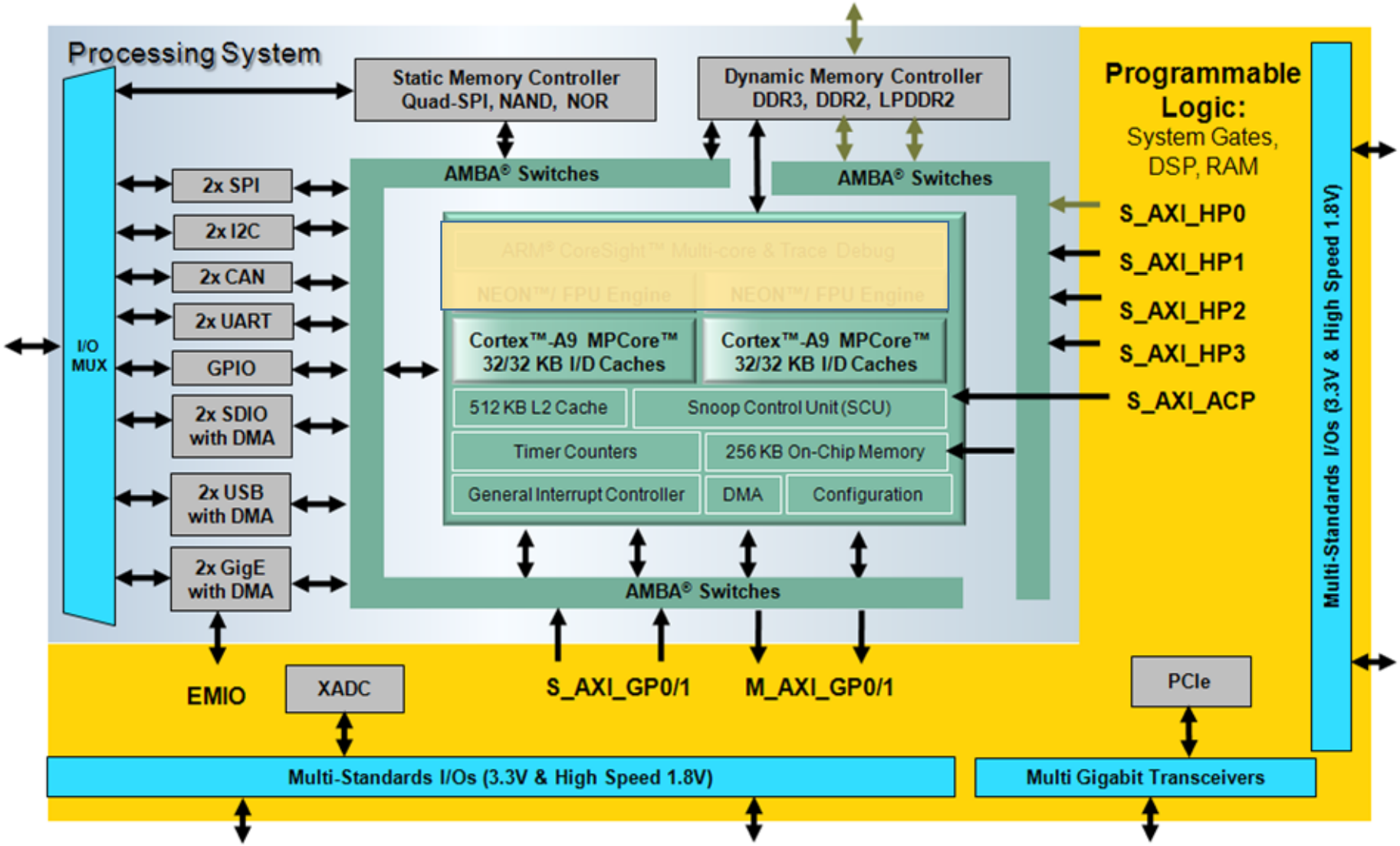
Zynq Architecture: PS and PL



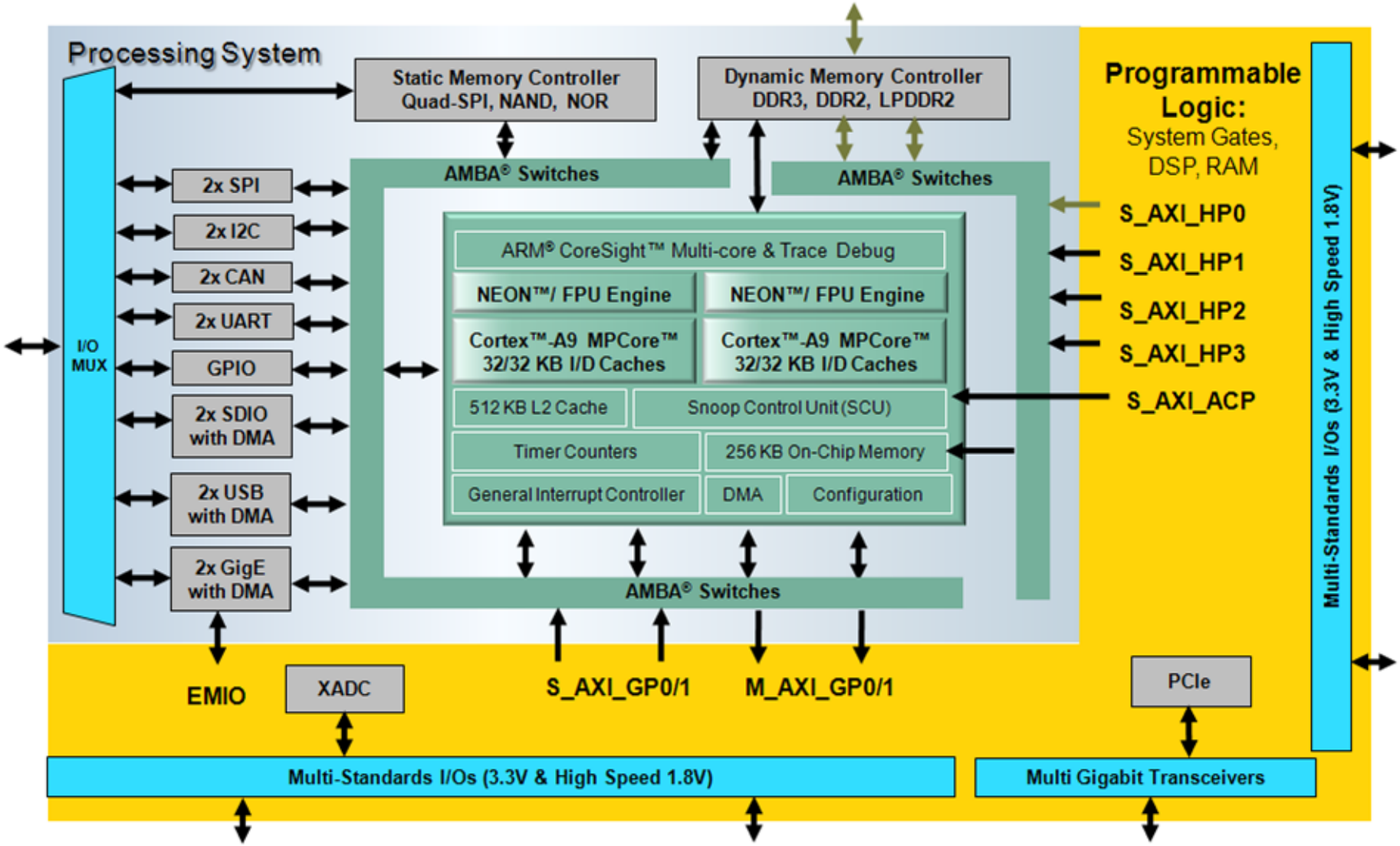
Zynq Architecture: PS and PL



Zynq Architecture: PS and PL



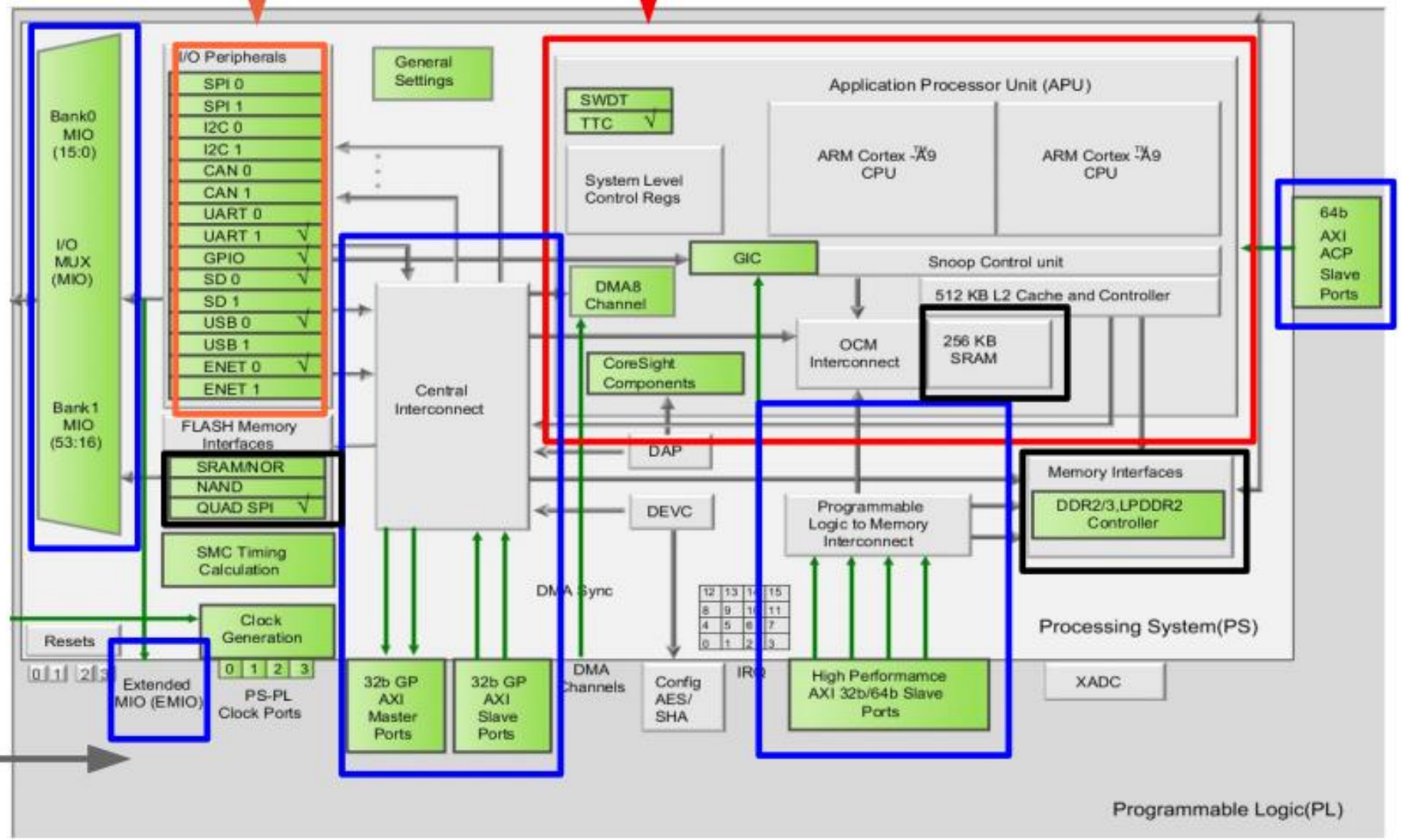
Zynq Architecture: PS and PL



Interconnections
AXI
Peripherals
Memory
User defined

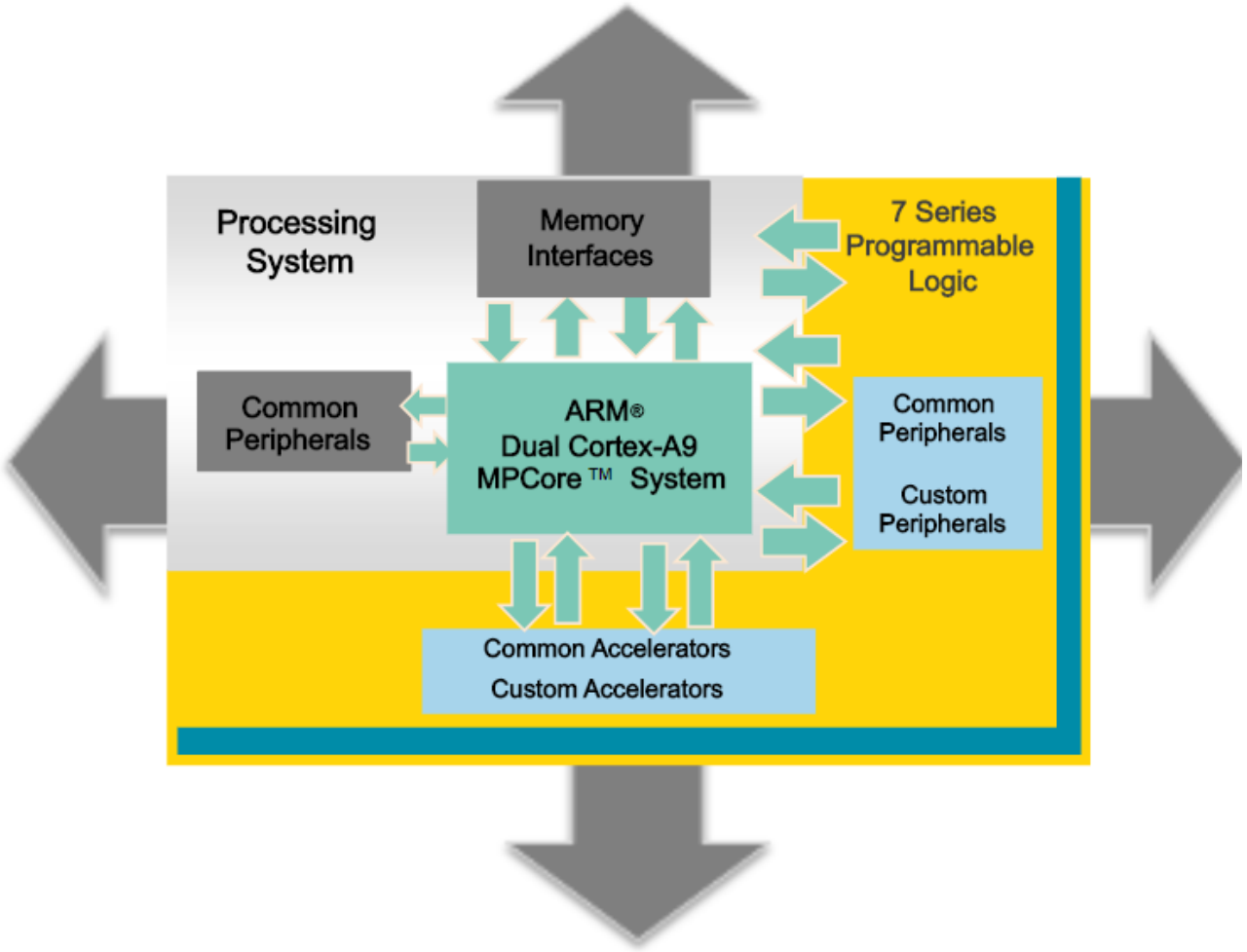
Memory
On-chip
DDR
Flash

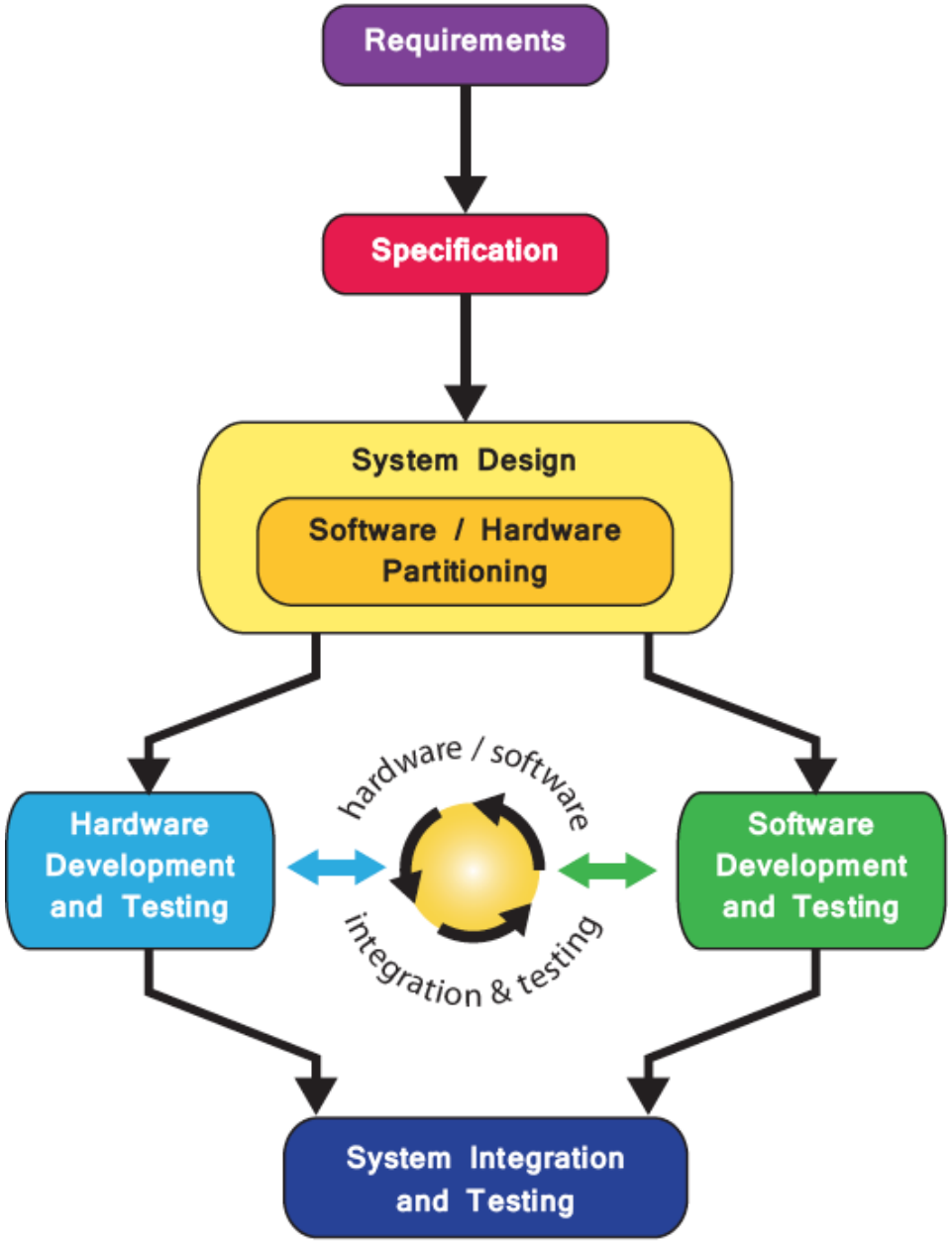
Programmable Logic



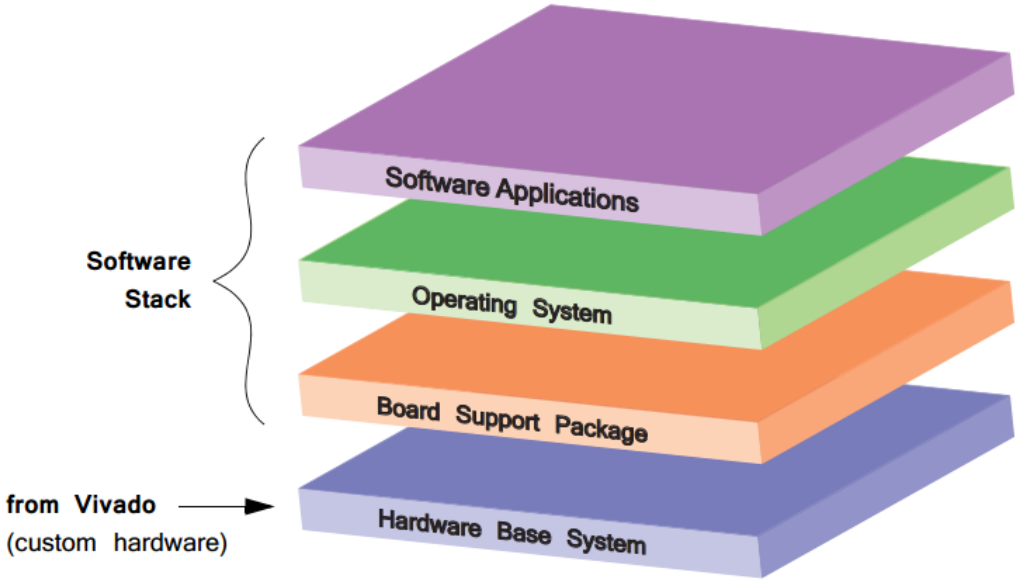
Zynq Boot Process

Zynq Design Flow





Revisiting Previous Lab



- ***Board Support Package (BSP)*:** Set of low-level drivers and functions that are used by the next layer up (the *Operating System or baremetal*) to communicate with the hardware
- **Software Applications** run on top of the Operating System — these collectively represent the uppermost layer in the software stack

- SDK provides the environment for creating BSPs, and developing and testing software for deployment in the upper layers.
- BSP (includes hardware parameters, device drivers, and low-level OS functions) should be refreshed if changes are made to the hardware base system.
- The purpose of ELF files is to program the PS, while BIT files are used to program the PL.
- Not all designs require both an Executable Linkable Format (ELF, *.elf) file and a BIT (*.bit) file to configure the device.
- If only one part of the Zynq device is used (PS or PL), then only the corresponding file type is needed.