[SoC Design]
Target Platform/Board

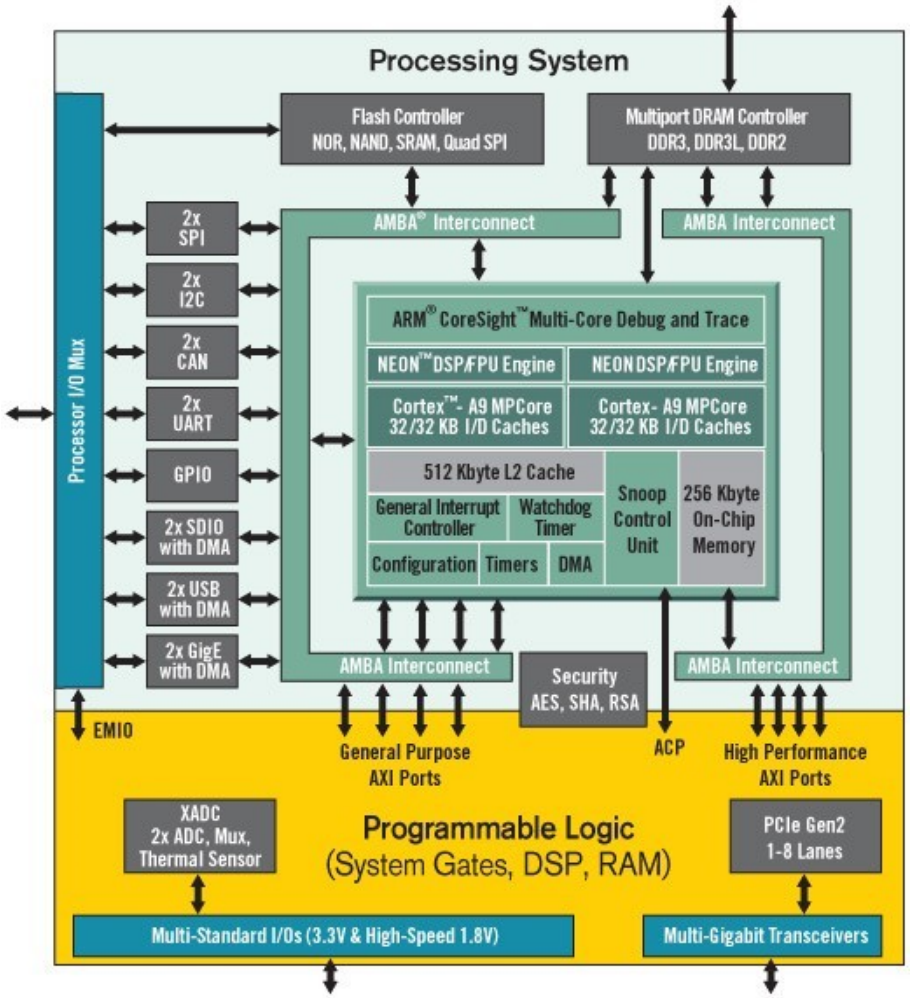
Chester Sungchung Park
SoC Design Lab, Konkuk University
Webpage: http://soclab.konkuk.ac.kr

Outline

- Target platform/board
  - Xilinx ZYNQ
  - Zybo Z7
- Lab0: board test

Target Platform

- Xilinx ZYNQ



Xilinx ZYNQ

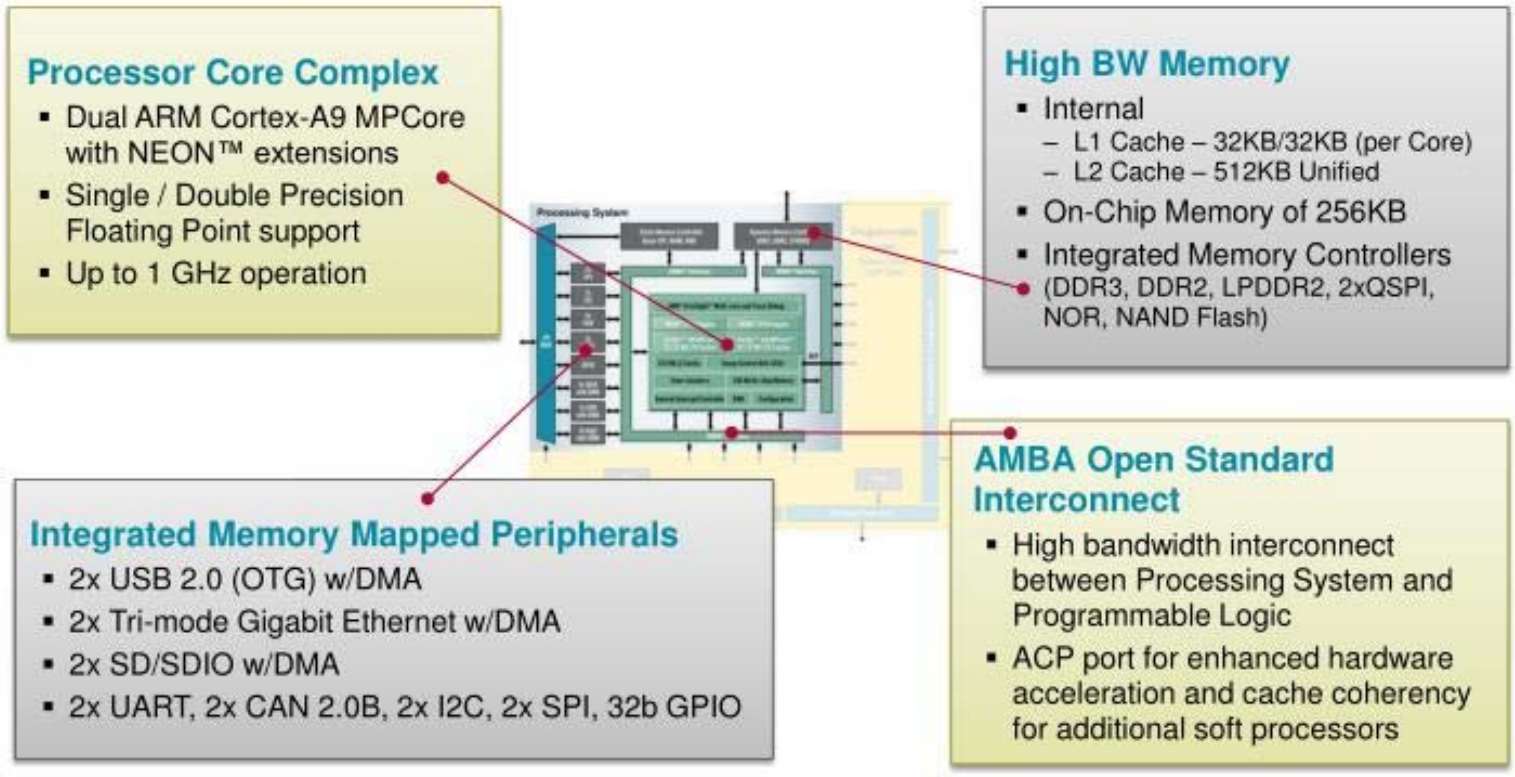
- Applications
  - Automotive driver assistance, driver information and infotainment
  - Broadcast camera
  - Industrial motor control, industrial networking and machine vision
  - IP and smart camera
  - LTE radio and baseband
  - Medical diagnostics and imaging
  - Multifunction printers
  - Video and night vision equipment

Xilinx ZYNQ

- System overview
  - Complete ARM-based processing system (PS)
    - ✓ Application processor unit
      - Dual ARM Cortex-A9 processors
      - Caches and support blocks
    - ✓ Fully integrated memory controllers
    - ✓ I/O peripherals
  - Tightly integrated programmable logic (PL)
    - ✓ Hardware acceleration
  - Flexible array of I/O
    - ✓ Wide range of external multi-standard I/O
    - ✓ High-performance integrated serial transceiver
    - ✓ Analog-to-digital converter inputs

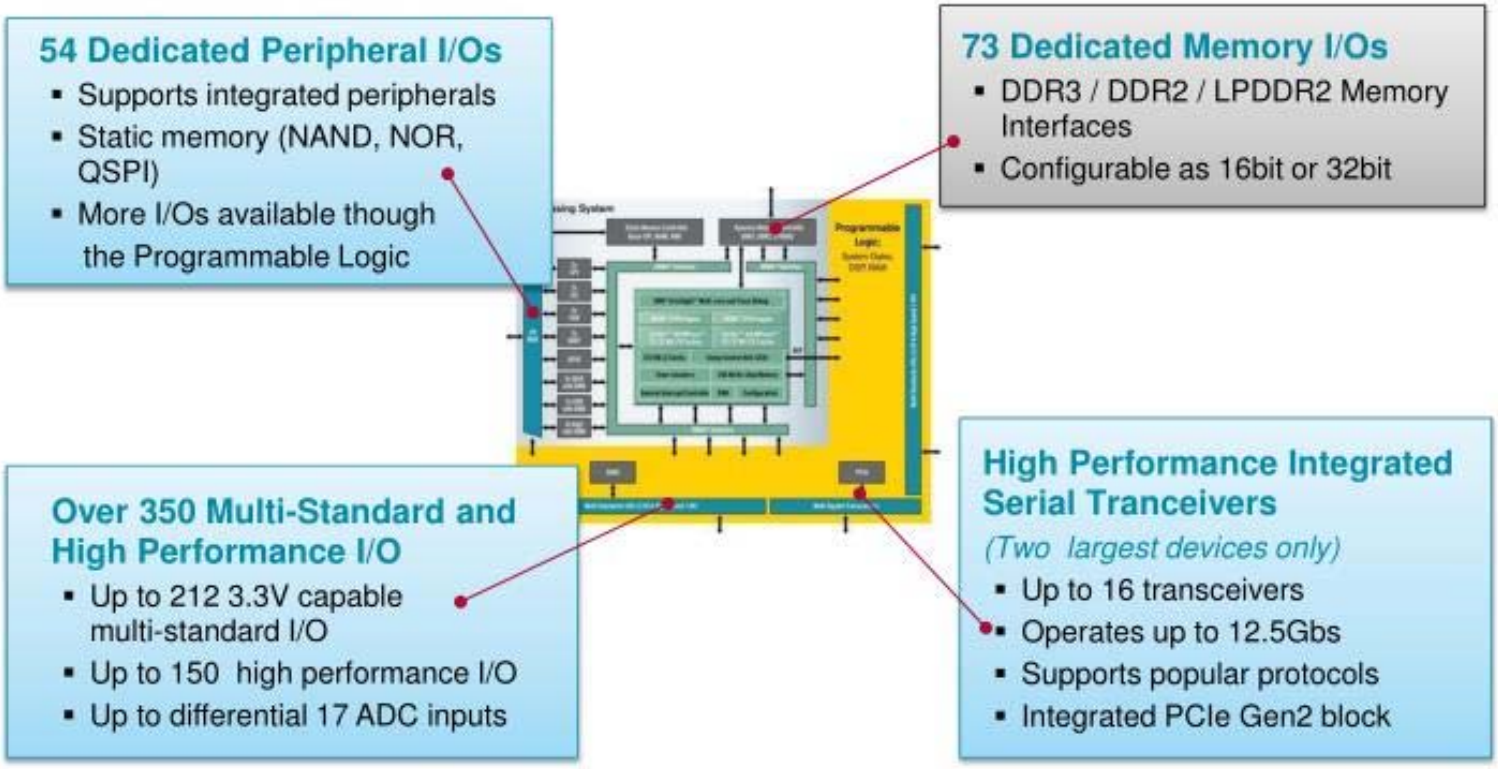
Xilinx ZYNQ

- Processing System (PS)

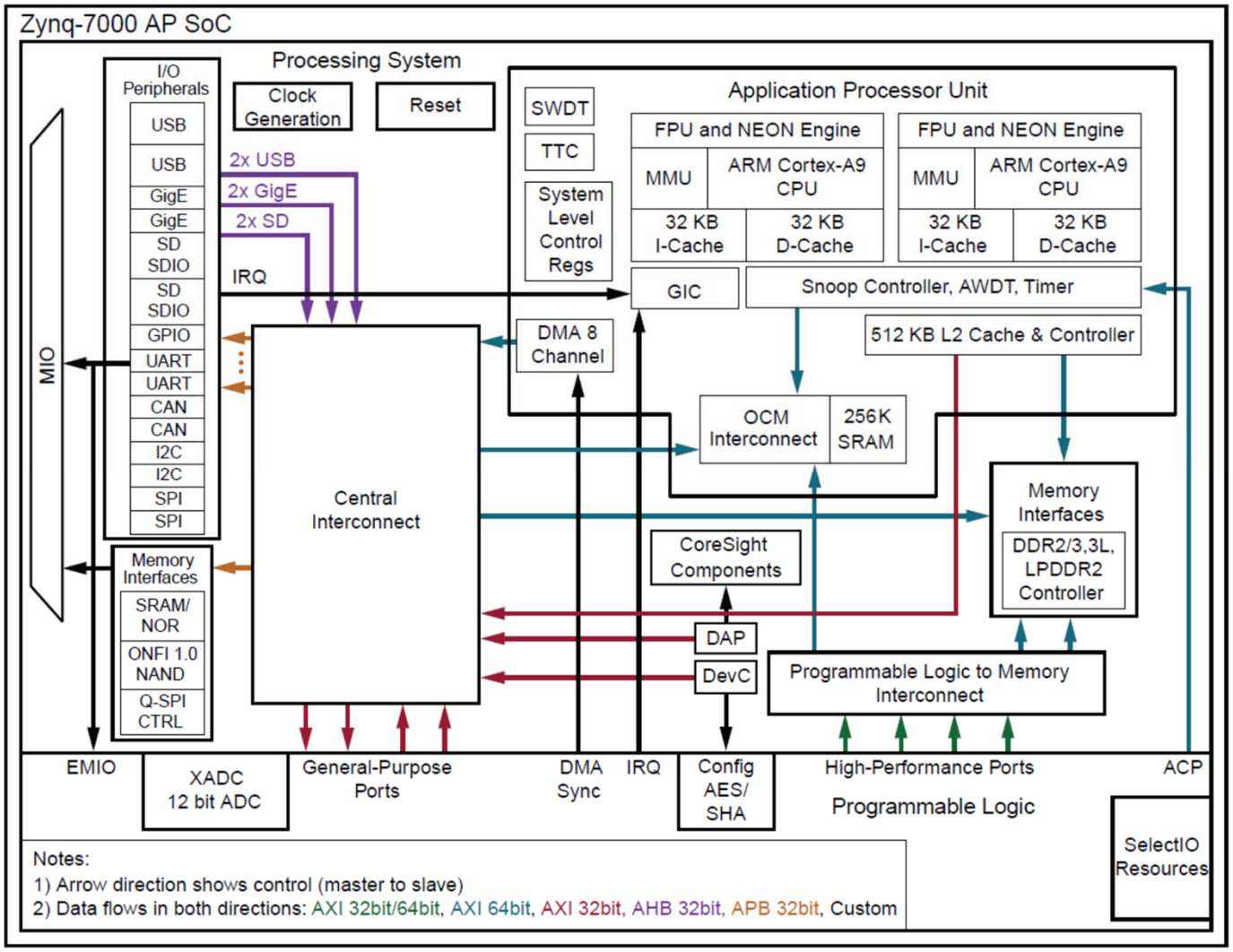


Xilinx ZYNQ

- External I/Os



Xilinx ZYNQ


Zynq-7000 AP SoC
Processing System
I/O Peripherals
USB
USB
GigE
GigE
SD SDIO
SD SDIO
GPIO
UART
UART
CAN
CAN
I2C
I2C
SPI
SPI
MIO
Clock Generation
Reset
2x USB
2x GigE
2x SD
IRQ
SWDT
TTC
System Level Control Regs
Application Processor Unit
FPU and NEON Engine
MMU
ARM Cortex-A9 CPU
32 KB I-Cache
32 KB D-Cache
FPU and NEON Engine
MMU
ARM Cortex-A9 CPU
32 KB I-Cache
32 KB D-Cache
GIC
Snoop Controller, AWDT, Timer
512 KB L2 Cache & Controller
DMA 8 Channel
OCM Interconnect
256K SRAM
Central Interconnect
Memory Interfaces
DDR2/3,3L, LPDDR2 Controller
CoreSight Components
DAP
DevC
Programmable Logic to Memory Interconnect
Memory Interfaces
SRAM/ NOR
ONFI 1.0 NAND
Q-SPI CTRL
EMIO
XADC 12 bit ADC
General-Purpose Ports
DMA Sync
IRQ
Config AES/ SHA
High-Performance Ports
ACP
Programmable Logic
SelectIO Resources
Notes:
1) Arrow direction shows control (master to slave)
2) Data flows in both directions: AXI 32bit/64bit, AXI 64bit, AXI 32bit, AHB 32bit, APB 32bit, Custom

Xilinx ZYNQ

❑ Specifications

		Cost-Optimized Devices						Mid-Range Devices			
	Device Name	Z-7007S	Z-7012S	Z-7014S	Z-7010	Z-7015	Z-7020	Z-7030	Z-7035	Z-7045	Z-7100
	Part Number	XC7Z007S	XC7Z012S	XC7Z014S	XC7Z010	XC7Z015	XC7Z020	XC7Z030	XC7Z035	XC7Z045	XC7Z100
Processing System (PS)	Processor Core	Single-Core ARM® Cortex™-A9 MPCore™ Up to 766MHz			Dual-Core ARM Cortex-A9 MPCore Up to 866MHz			Dual-Core ARM Cortex-A9 MPCore Up to 1GHz[1]			
	Processor Extensions	NEON™ SIMD Engine and Single/Double Precision Floating Point Unit per processor									
	L1 Cache	32KB Instruction, 32KB Data per processor									
	L2 Cache	512KB									
	On-Chip Memory	256KB									
	External Memory Support[2]	DDR3, DDR3L, DDR2, LPDDR2									
	External Static Memory Support[2]	2x Quad-SPI, NAND, NOR									
	DMA Channels	8 (4 dedicated to PL)									
	Peripherals	2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 4x 32b GPIO									
	Peripherals w/ built-in DMA[2]	2x USB 2.0 (OTG), 2x Tri-mode Gigabit Ethernet, 2x SD/SDIO									
	Security[3]	RSA Authentication of First Stage Boot Loader, AES and SHA 256b Decryption and Authentication for Secure Boot									
	Processing System to Programmable Logic Interface Ports (Primary Interfaces & Interrupts Only)	2x AXI 32b Master, 2x AXI 32b Slave 4x AXI 64b/32b Memory AXI 64b ACP 16 Interrupts									
Programmable Logic (PL)	7 Series PL Equivalent	Artix®-7	Artix-7	Artix-7	Artix-7	Artix-7	Artix-7	Kintex®-7	Kintex-7	Kintex-7	Kintex-7
	Logic Cells	23K	55K	65K	28K	74K	85K	125K	275K	350K	444K
	Look-Up Tables (LUTs)	14,400	34,400	40,600	17,600	46,200	53,200	78,600	171,900	218,600	277,400
	Flip-Flops	28,800	68,800	81,200	35,200	92,400	106,400	157,200	343,800	437,200	554,800
	Total Block RAM	1.8Mb	2.5Mb	3.8Mb	2.1Mb	3.3Mb	4.9Mb	9.3Mb	17.6Mb	19.2Mb	26.5Mb
	(# 36Kb Blocks)	(50)	(72)	(107)	(60)	(95)	(140)	(265)	(500)	(545)	(755)
	DSP Slices	66	120	170	80	160	220	400	900	900	2,020
	PCI Express®	—	Gen2 x4	—	—	Gen2 x4	—	Gen2 x4	Gen2 x8	Gen2 x8	Gen2 x8
	Analog Mixed Signal (AMS) / XADC[2]	2x 12 bit, MSPS ADCs with up to 17 Differential Inputs									
	Security[3]	AES & SHA 256b Decryption & Authentication for Secure Programmable Logic Config									
	Speed Grades: Commercial	-1			-1			-1			-1
	Speed Grades: Extended	-2			-2,-3			-2,-3			-2
	Speed Grades: Industrial	-1, -2			-1, -2, -1L			-1, -2, -2L			-1, -2, -2L

Xilinx ZYNQ

- Development tools
  - **HW: Vivado Design Suites**
    - ✓PS configuration wizard
    - ✓IP integrator
    - ✓RTL synthesis, implementation, generate bit stream
  - **SW: Eclipse IDE-based Software Development Kit (SDK)**
    - ✓Project workspace
    - ✓C/C++ compiler for the ARM Cortex-A9 processor
    - ✓Debugger for the ARM Cortex-A9 processor
    - ✓Instruction Set Simulation

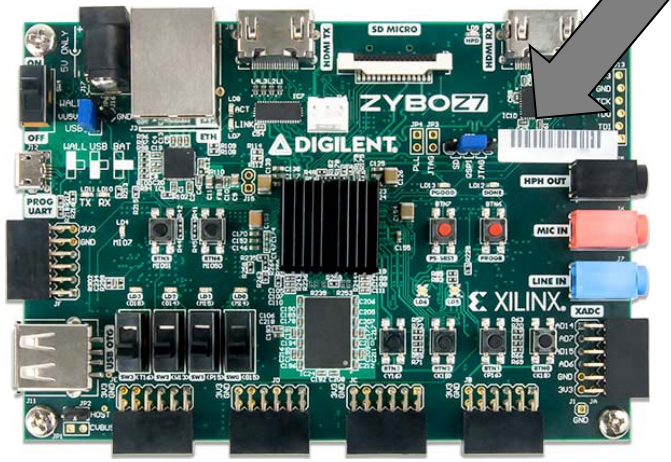
Xilinx ZYNQ

- Design flow

Vivado

SDK

ZYNQ/Zybo



Xilinx ZYNQ

❑ Design flow (cont'd)

Vivado

1. Launch Vivado
2. Create IP block [IP integrator]
3. Configuration PS settings
4. Add IP
5. Add Top-Level HDL
6. Add Constraints file
7. Add Generate Bitstream
8. Export hardware to SDK

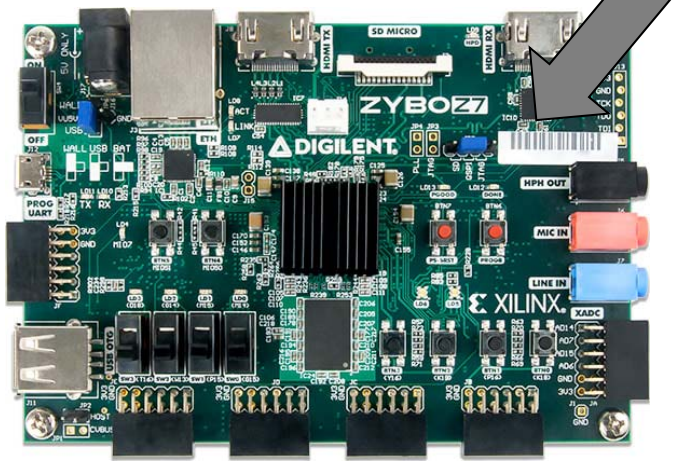
SDK

9. Specify hardware built from Vivado
10. Add software project & build
11. Program bitstream

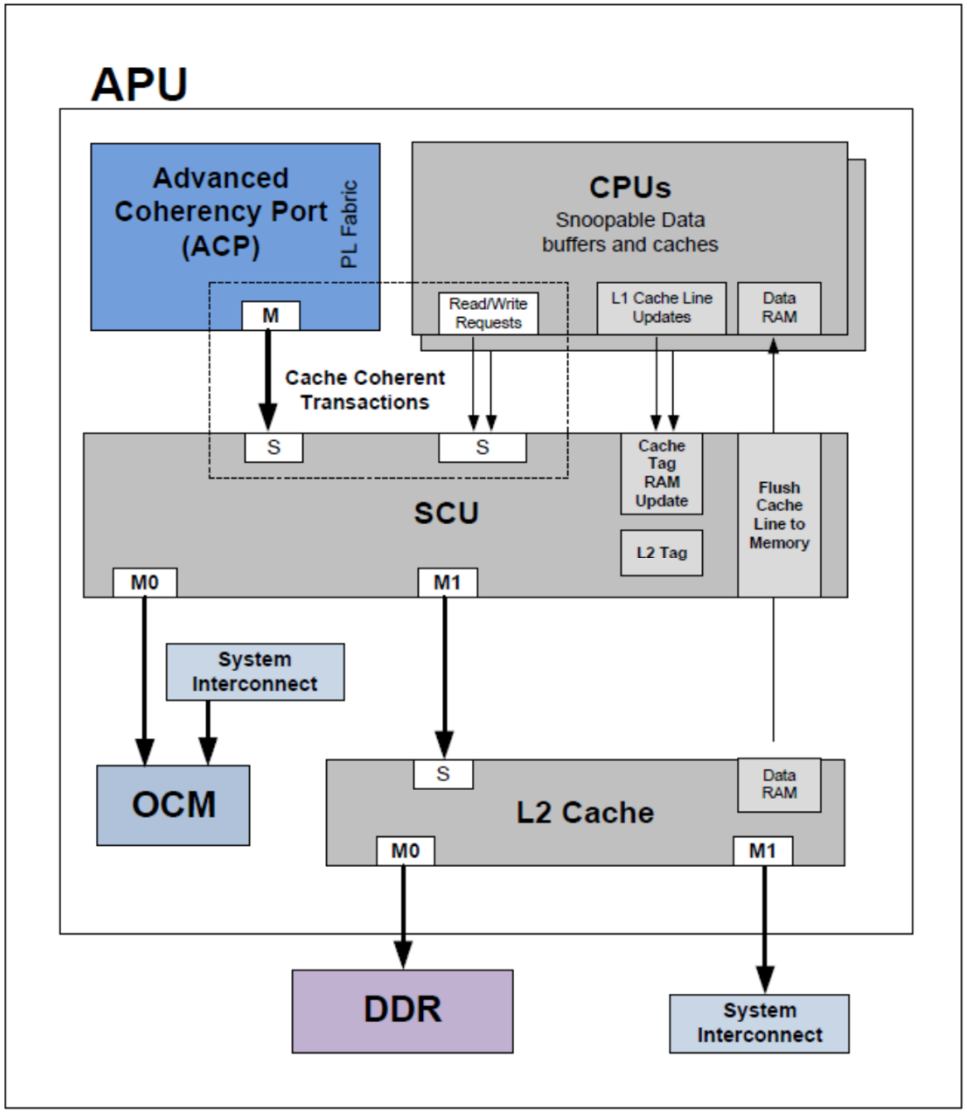
SDK
Xilinx SDK

This board test will run only the software part (Step 10)

ZYNQ/Zybo

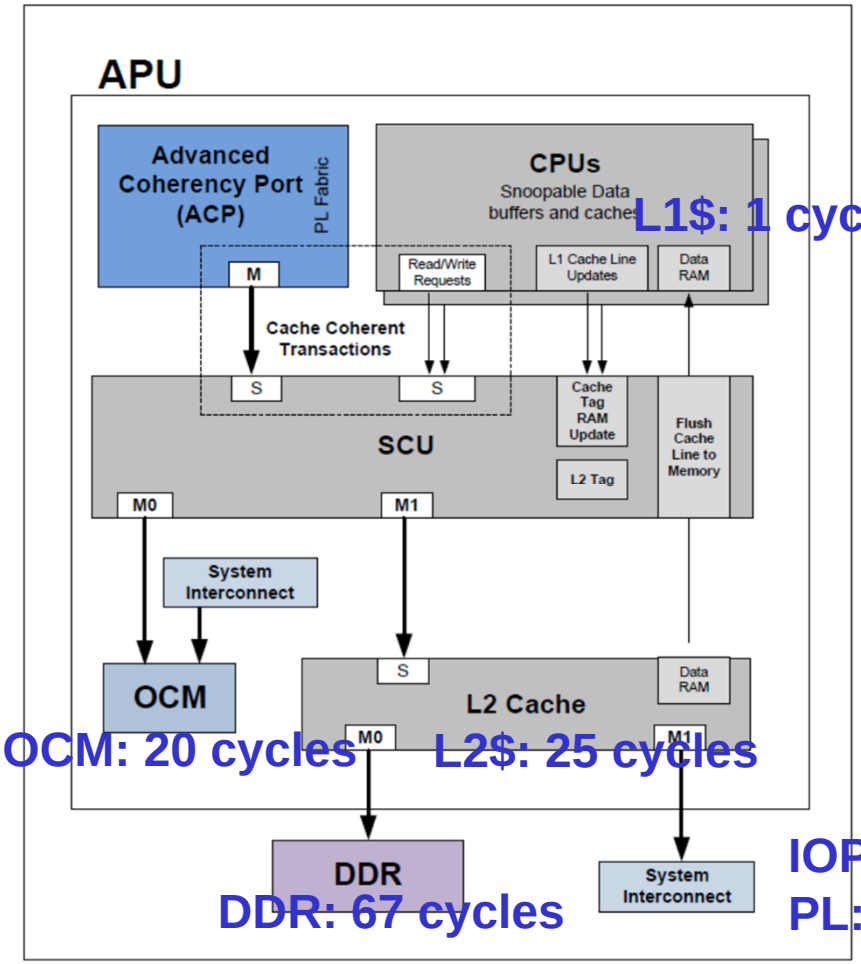


Memory Hierarchy



Memory Hierarchy

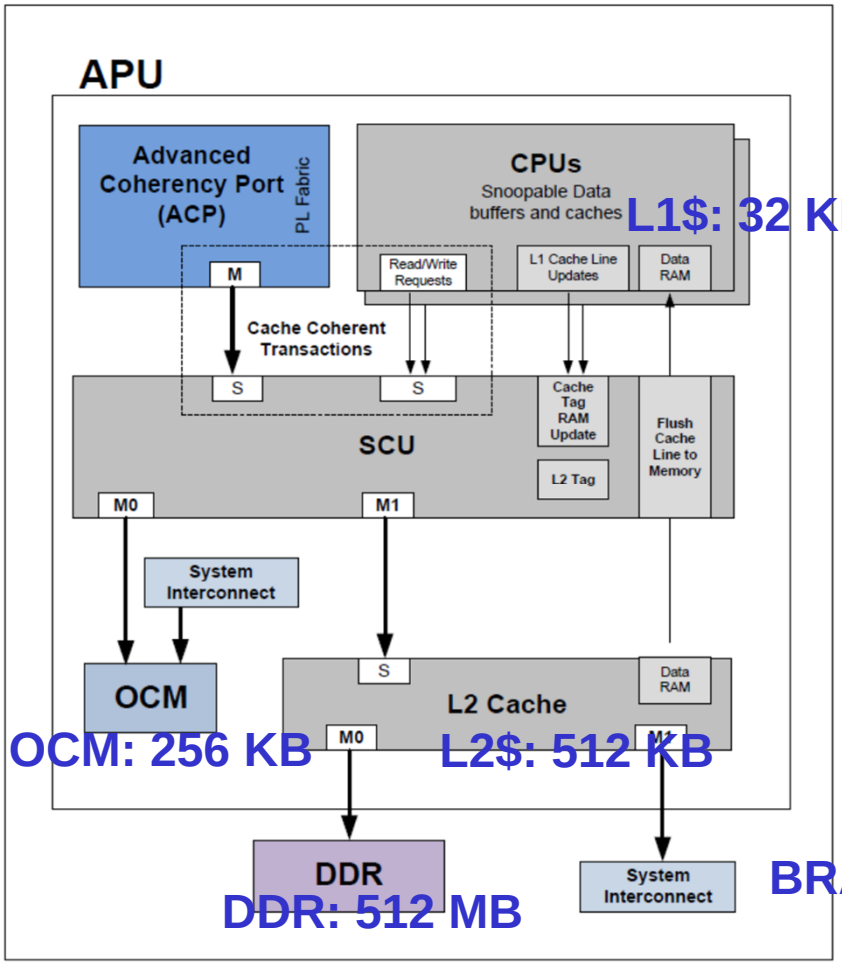
- Access latency (CPU cycle)



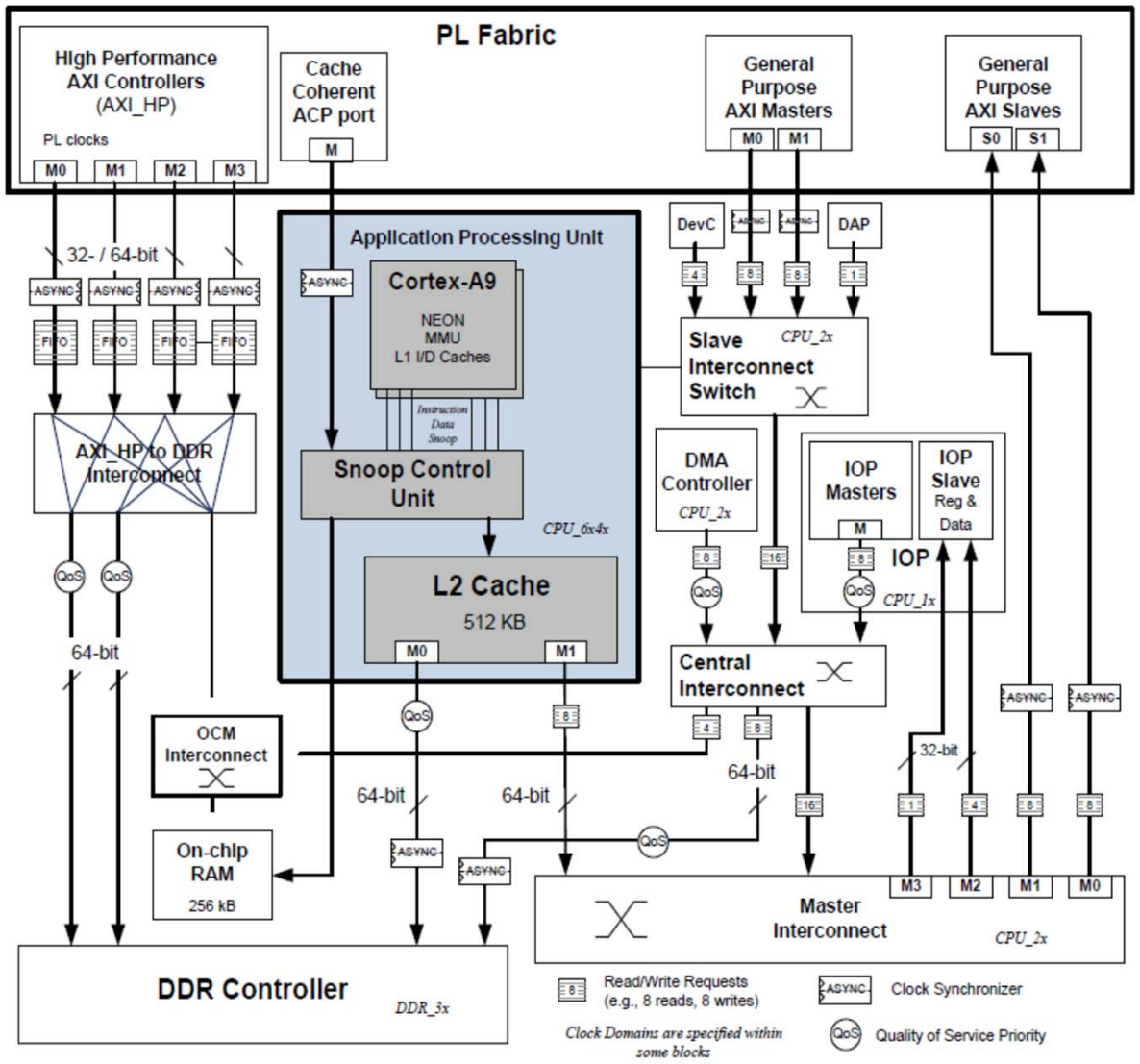
- CPU @ 667 MHz
- DDR @ 533 MHz
- PL @150 MHz

Memory Hierarchy

- Capacity

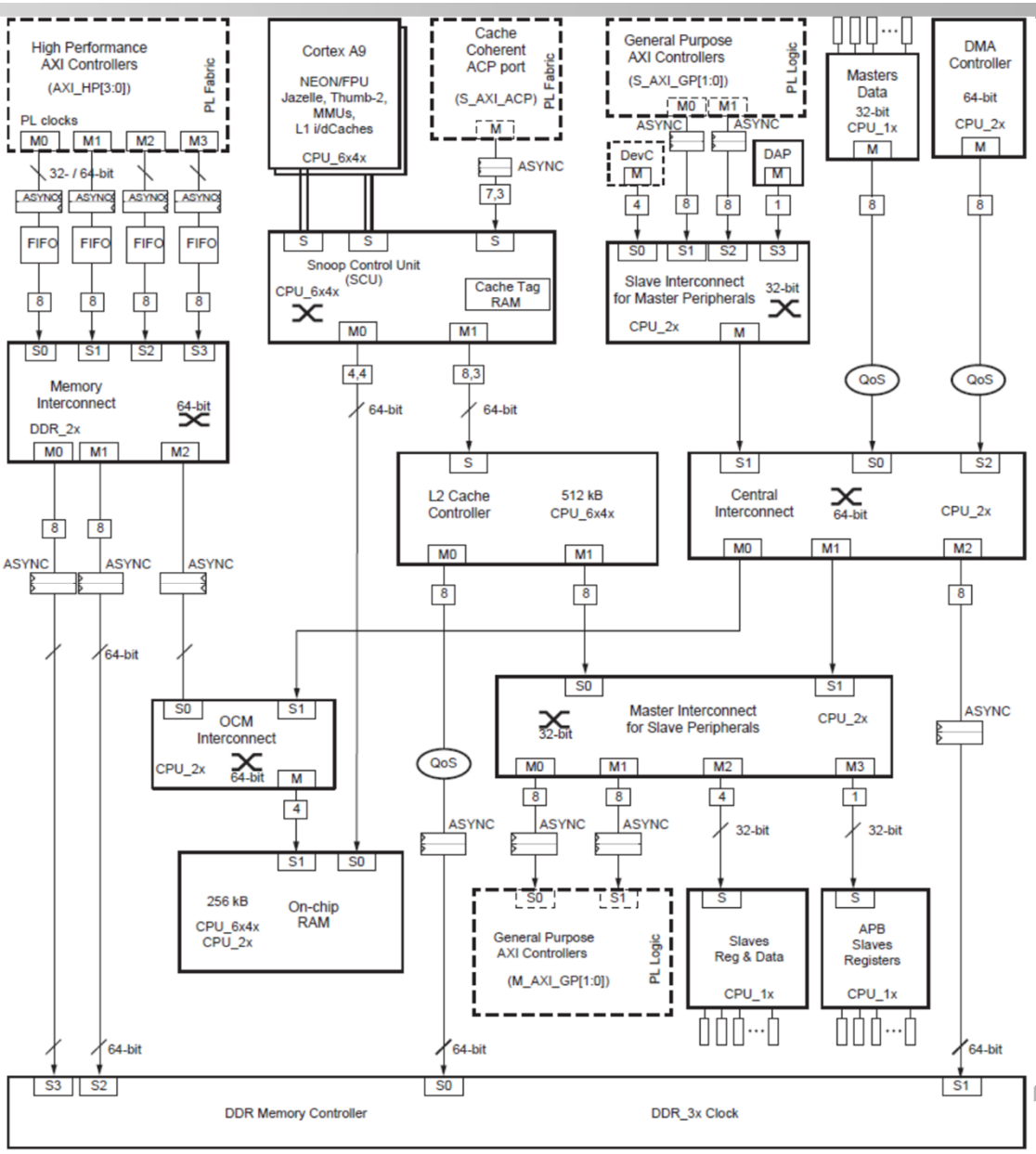


On-Chip Interconnect



On-Chip Interconnect

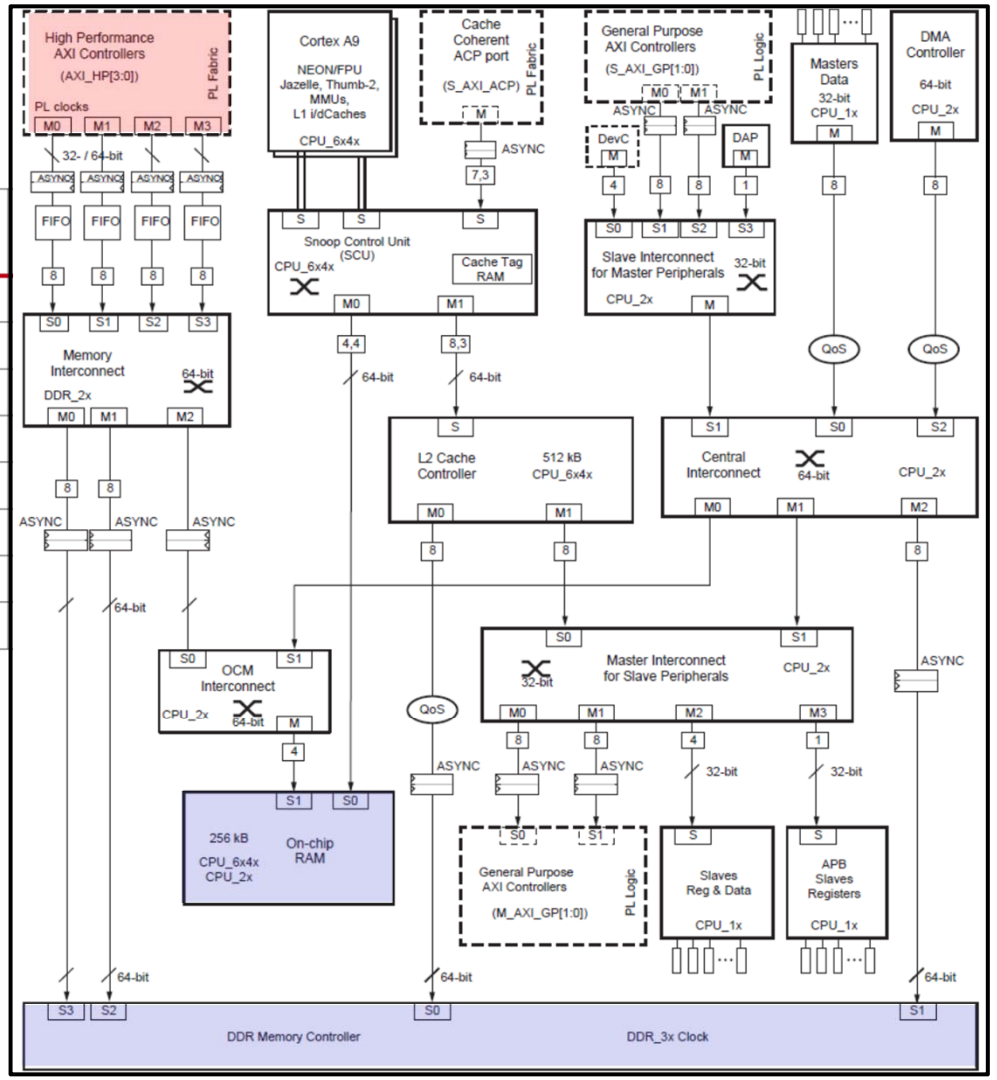
- Detailed…



On-Chip Interconnect

❑ Connections

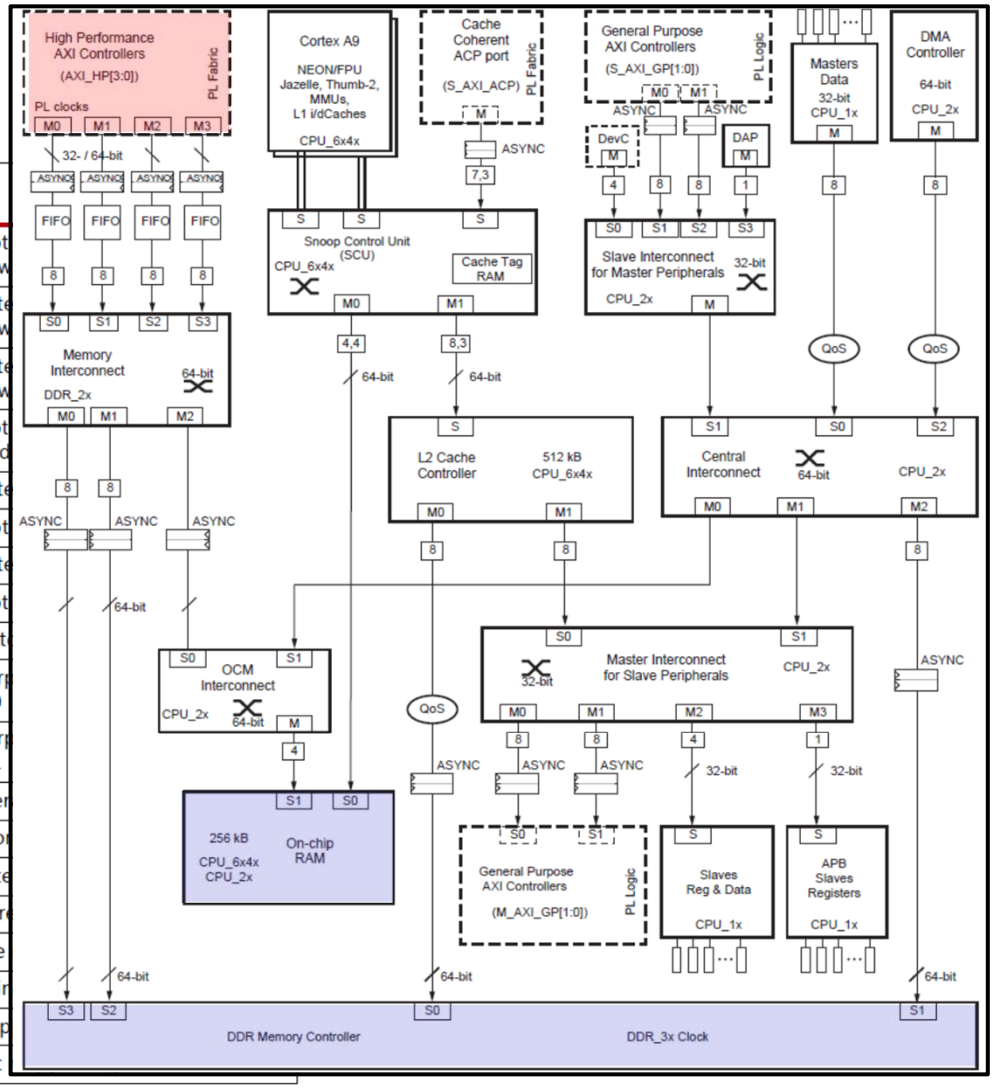
Master \ Slave	On-chip RAM	DDR Port 0	DDR Port 1	DDR Port 2	DDR Port 3	M_AXI _GP
CPUs	X	X				X
AXI_ACP	X	X				X
AXI_HP{0,1}	X				X	
AXI_HP{2,3}	X			X		
S_AXI_GP{0,1}	X		X			X
DMA Controller	X		X			X
AHB Masters	X		X			X
DevC, DAP	X		X			X



On-Chip Interconnect

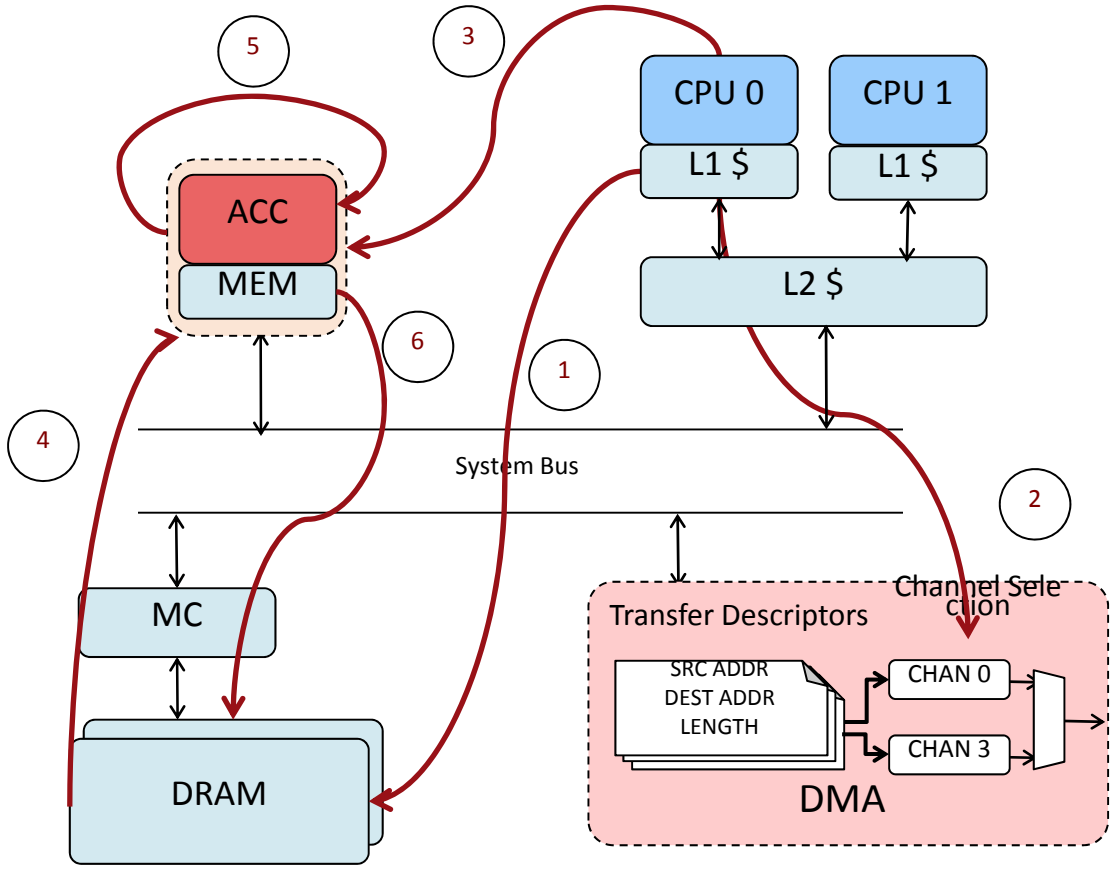
❑ Address range

Address Range	CPUs and ACP	AXI_HP	Other Bus Masters(1)	
0000_0000 to 0003_FFFF(2)	OCM	OCM	OCM	Address not mapped low
	DDR	OCM	OCM	Address filte mapped low
	DDR			Address filte mapped low
				Address not not mapped
0004_0000 to 0007_FFFF	DDR			Address filte
				Address not
0008_0000 to 000F_FFFF	DDR	DDR	DDR	Address filte
		DDR	DDR	Address not
0010_0000 to 3FFF_FFFF	DDR	DDR	DDR	Accessible t
4000_0000 to 7FFF_FFFF	PL		PL	General Purp M_AXI_GP0
8000_0000 to BFFF_FFFF	PL		PL	General Purp M_AXI_GP1
E000_0000 to E02F_FFFF	IOP		IOP	I/O Peripher
E100_0000 to E5FF_FFFF	SMC		SMC	SMC Memor
F800_0000 to F800_0BFF	SLCR		SLCR	SLCR registe
F800_1000 to F880_FFFF	PS		PS	PS System r
F890_0000 to F8F0_2FFF	CPU			CPU Private
FC00_0000 to FDFF_FFFF(4)	Quad-SPI		Quad-SPI	Quad-SPI lir
FFFC_0000 to FFFF_FFFF(2)	OCM	OCM	OCM	OCM is map
				OCM is not



Accelerator on ZYNQ

- DMA-controlled accelerator

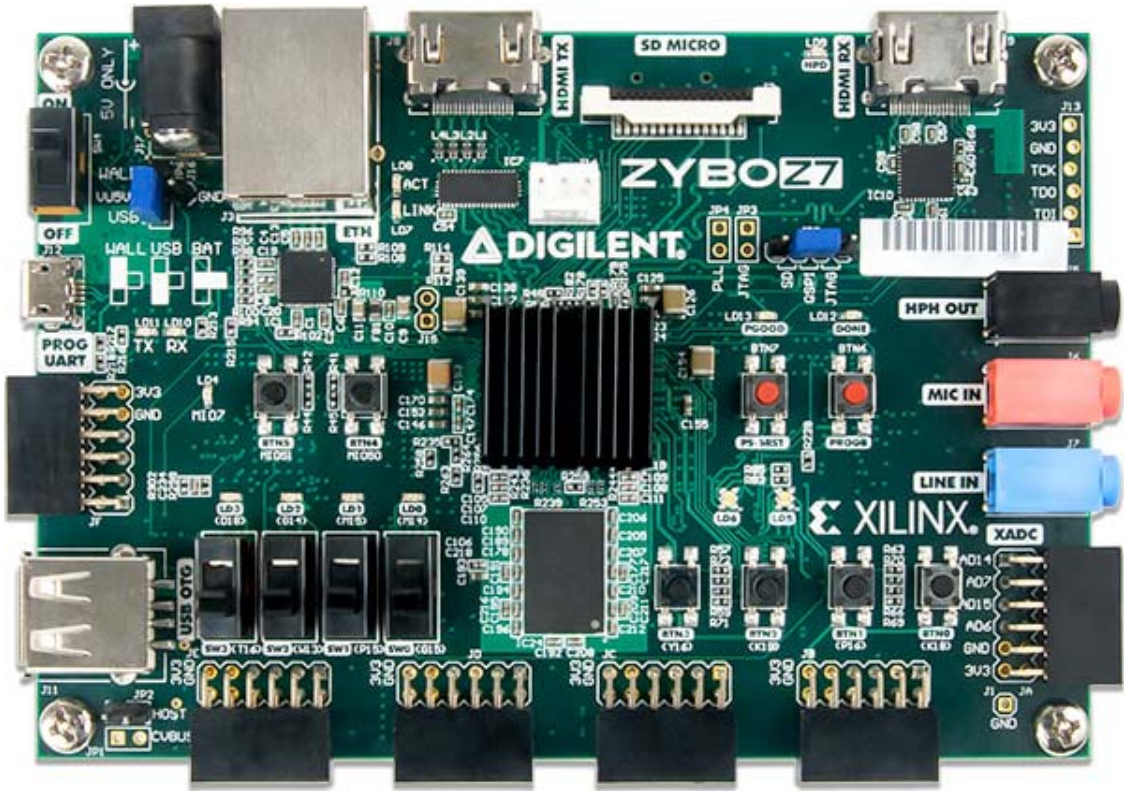


Accelerator on ZYNQ

- Design examples (SoCDesignLab@KU)
  - WiFi baseband modem (SDR II Evaluation Kit)
    - ✓ https://www.sites.google.com/site/kusocdesignlab/demos/graduationworks2015
  - Convolutional neural network for WiFi
    - ✓ https://www.sites.google.com/site/kusocdesignlab/demos/socforcnn-basedlinkadaptation
  - Convolutional neural network for image recognition
    - ✓ https://www.sites.google.com/site/kusocdesignlab/demos/cnnacceleratoronzynq

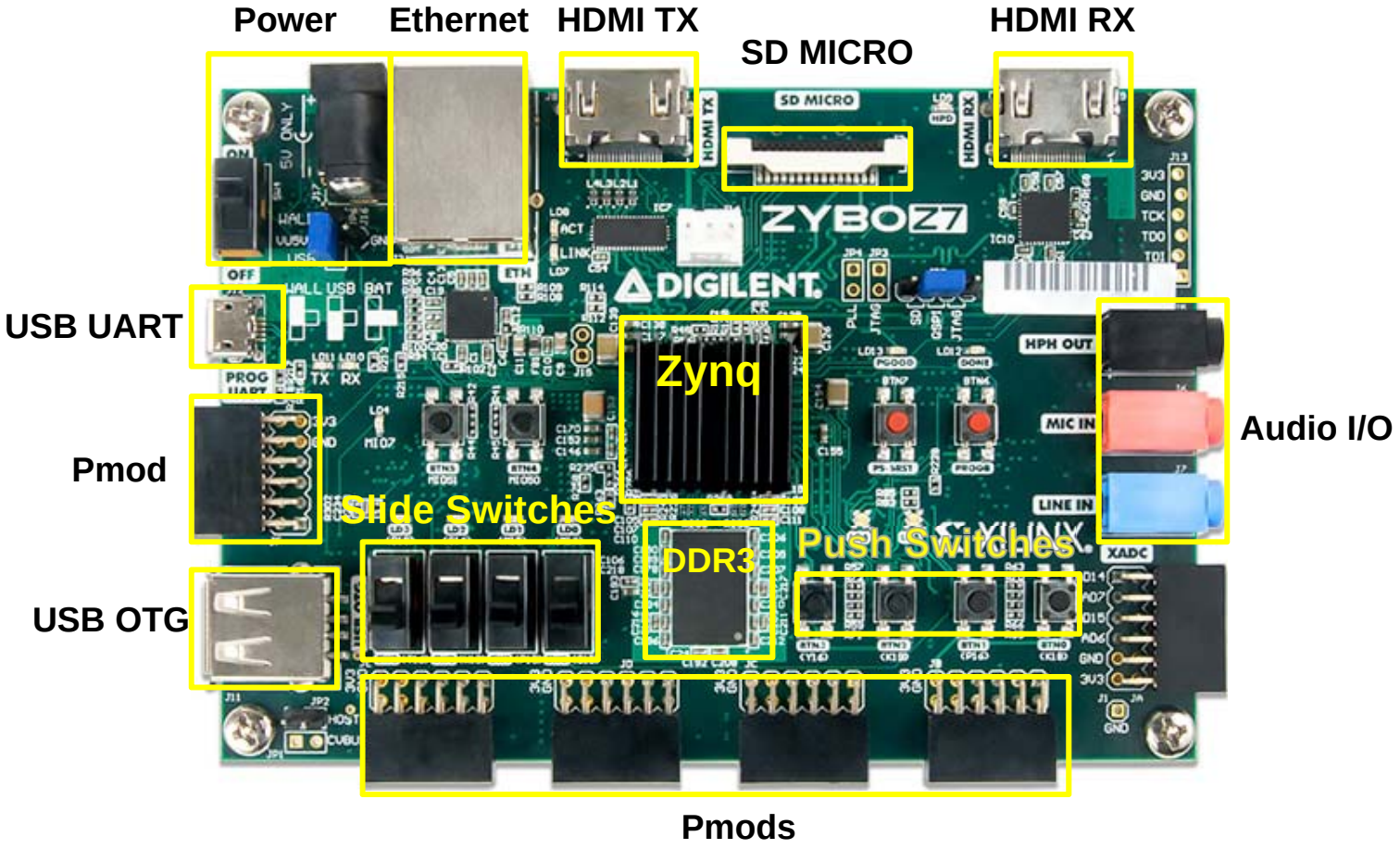
Target Board

- Zybo Z7



Zybo Z7

- Block diagram

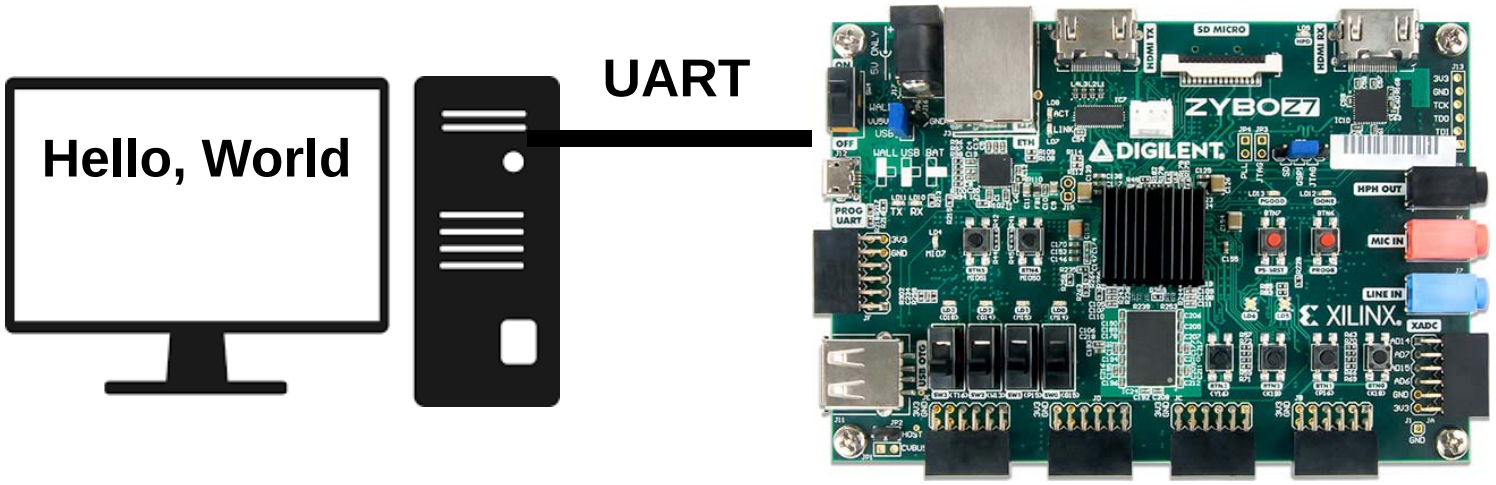


Lab 0: Board Test

- Objectives
  - Creating a project using SDK
  - Running a C application using SDK
  - Communicating with the board using UART

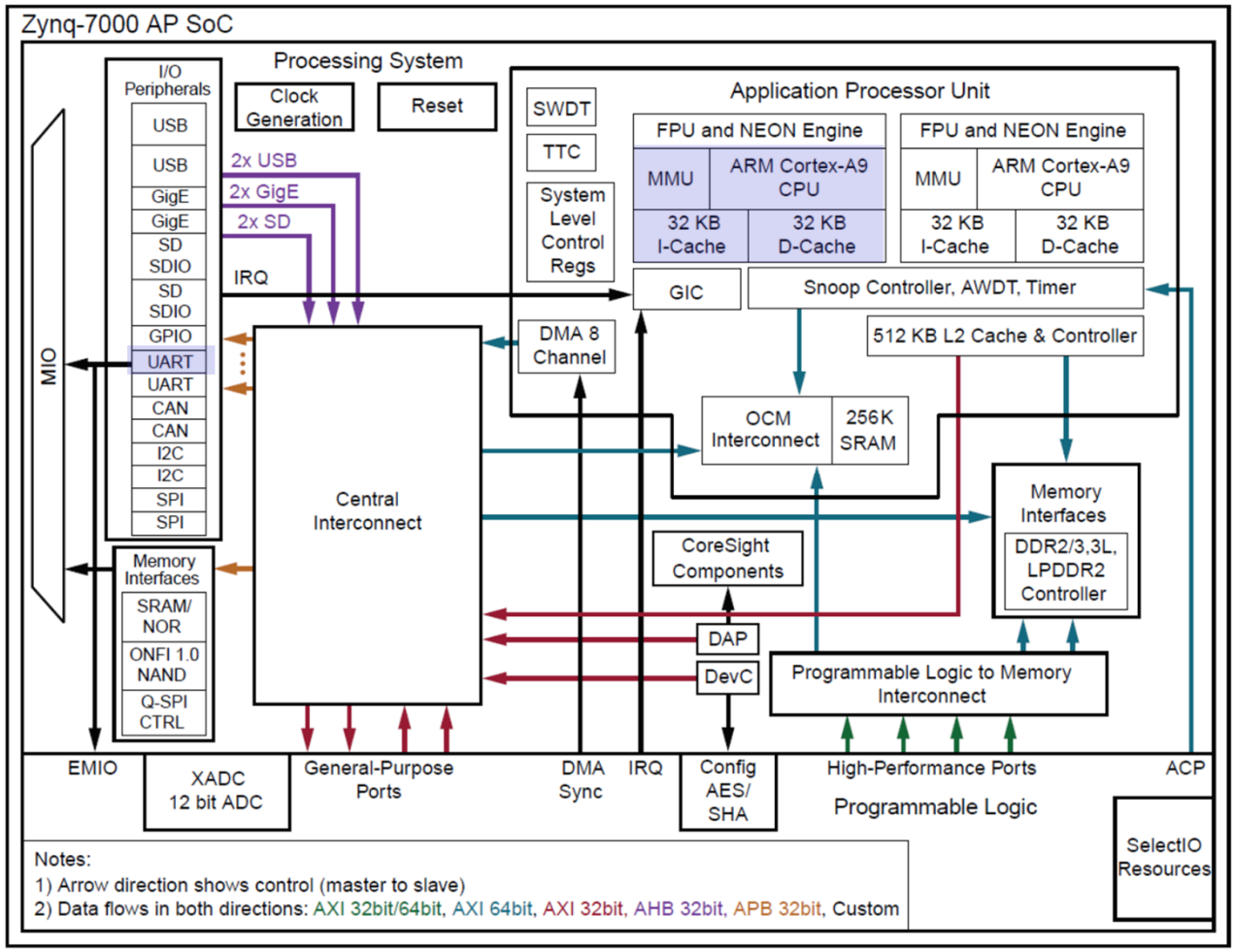
Lab 0: Board Test

- Design description
  - Showing how to create a simple software design



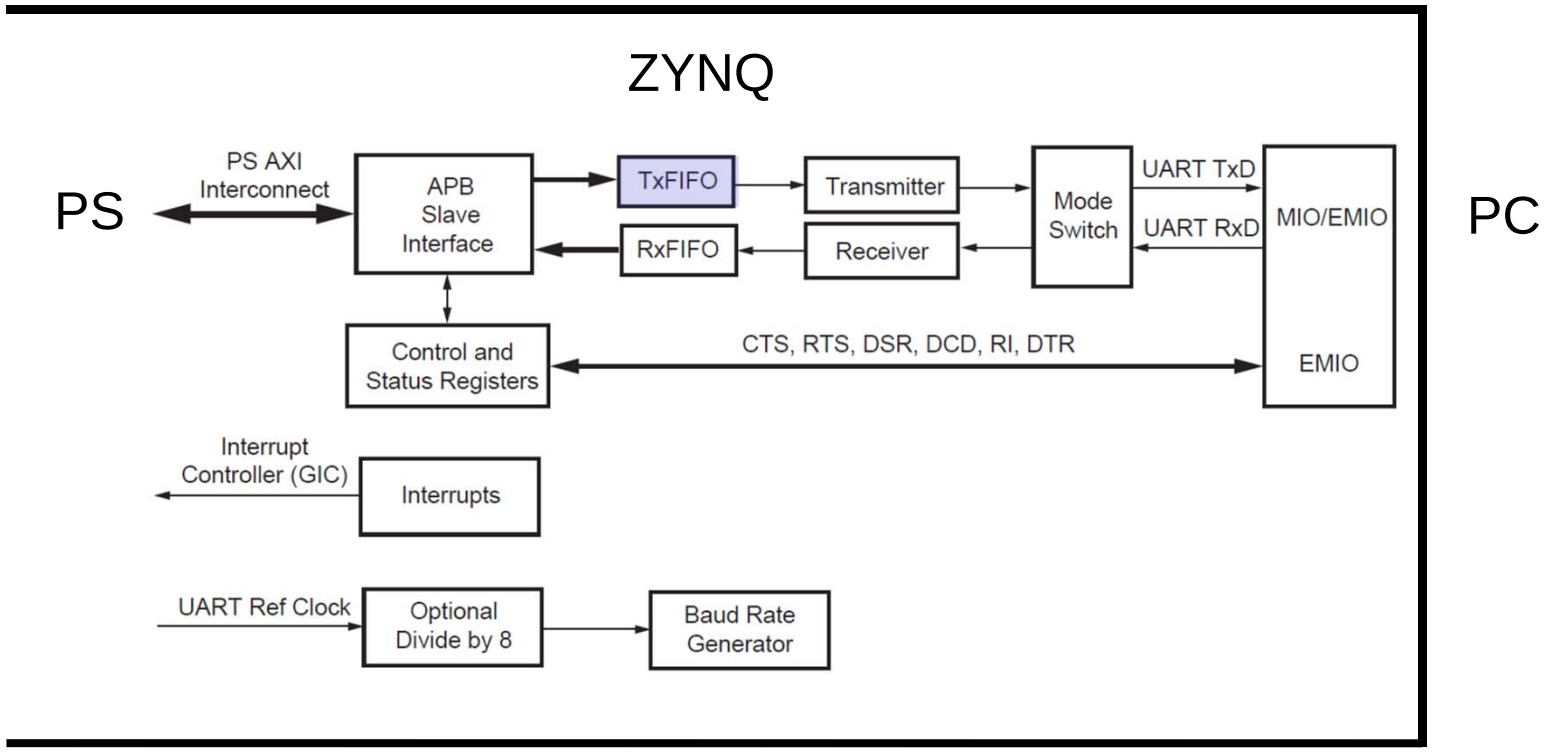
Lab 0: Board Test

- Block diagram: ZYNQ



Lab 0: Board Test

- Block diagram: UART



Lab 0: Board Test

- Address map

system.hdf

design_1_wrapper_hw_platform_0 Hardware Platform Specification

Design Information

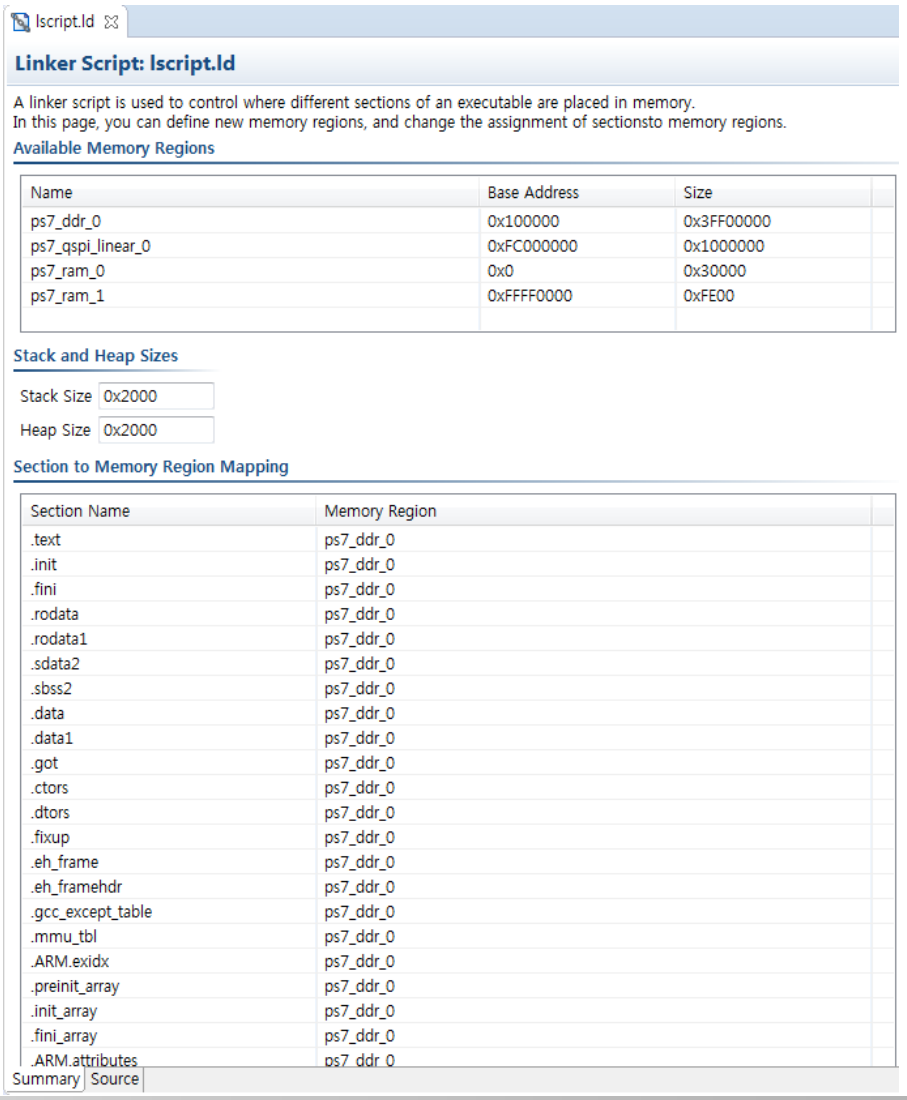
Target FPGA Device:	7z020
Part:	xc7z020clg400-1
Created With:	Vivado 2017.4.1
Created On:	Fri Nov 1 19:27:43 2019

Address Map for processor ps7_cortexa9_[0-1]

Cell	Base Addr	High Addr	Slave I/f	Mem/Reg
ps7_intc_dist_0	0xf8f01000	0xf8f01fff		REGISTER
ps7_gpio_0	0xe000a000	0xe000afff		REGISTER
ps7_scutimer_0	0xf8f00600	0xf8f0061f		REGISTER
ps7_slcr_0	0xf8000000	0xf8000fff		REGISTER
ps7_scuwdt_0	0xf8f00620	0xf8f006ff		REGISTER
ps7_l2cachec_0	0xf8f02000	0xf8f02fff		REGISTER
ps7_scuc_0	0xf8f00000	0xf8f000fc		REGISTER
ps7_qspi_linear_0	0xfc000000	0xfcffffff		FLASH
ps7_pmu_0	0xf8893000	0xf8893fff		REGISTER
ps7_afi_1	0xf8009000	0xf8009fff		REGISTER
ps7_afi_0	0xf8008000	0xf8008fff		REGISTER
ps7_qspi_0	0xe000d000	0xe000dfff		REGISTER
ps7_usb_0	0xe0002000	0xe0002fff		REGISTER
ps7_afi_3	0xf800b000	0xf800bfff		REGISTER
ps7_afi_2	0xf800a000	0xf800afff		REGISTER
ps7_globaltimer_0	0xf8f00200	0xf8f002ff		REGISTER
ps7_dma_s	0xf8003000	0xf8003fff		REGISTER
ps7_iop_bus_config_0	0xe0200000	0xe0200fff		REGISTER
ps7_xadc_0	0xf8007100	0xf8007120		REGISTER
ps7_ddr_0	0x00100000	0x3fffffff		MEMORY
ps7_ddrc_0	0xf8006000	0xf8006fff		REGISTER
ps7_ocmc_0	0xf800c000	0xf800cfff		REGISTER
ps7_pl310_0	0xf8f02000	0xf8f02fff		REGISTER
ps7_uart_1	0xe0001000	0xe0001fff		REGISTER
ps7_coresight_comp_0	0xf8800000	0xf88fffff		REGISTER
ps7_scugic_0	0xf8f00100	0xf8f001ff		REGISTER
ps7_ethernet_0	0xe000b000	0xe000bfff		REGISTER

Lab 0: Board Test

- Section map



lscript.ld

Linker Script: lscript.ld

A linker script is used to control where different sections of an executable are placed in memory.
In this page, you can define new memory regions, and change the assignment of sectionsto memory regions.

Available Memory Regions

Name	Base Address	Size
ps7_ddr_0	0x100000	0x3FF00000
ps7_qspi_linear_0	0xFC000000	0x1000000
ps7_ram_0	0x0	0x30000
ps7_ram_1	0xFFFF0000	0xFE00

Stack and Heap Sizes

Stack Size 0x2000

Heap Size 0x2000

Section to Memory Region Mapping

Section Name	Memory Region
.text	ps7_ddr_0
.init	ps7_ddr_0
.fini	ps7_ddr_0
.rodata	ps7_ddr_0
.rodata1	ps7_ddr_0
.sdata2	ps7_ddr_0
.sbss2	ps7_ddr_0
.data	ps7_ddr_0
.data1	ps7_ddr_0
.got	ps7_ddr_0
.ctors	ps7_ddr_0
.dtors	ps7_ddr_0
.fixup	ps7_ddr_0
.eh_frame	ps7_ddr_0
.eh_framehdr	ps7_ddr_0
.gcc_except_table	ps7_ddr_0
.mmu_tbl	ps7_ddr_0
.ARM.exidx	ps7_ddr_0
.preinit_array	ps7_ddr_0
.init_array	ps7_ddr_0
.fini_array	ps7_ddr_0
.ARM.attributes	ps7_ddr_0

Summary | Source

Lab 0: Board Test

- Source code: xparameters.h

xparameters.h

```
#include "xparameters_ps.h"

#define STDIN_BASEADDRESS 0xE0001000
#define STDOUT_BASEADDRESS 0xE0001000
```

xparameters.h

```
/* Definitions for driver UARTPS */
#define XPAR_XUARTPS_NUM_INSTANCES 1

/* Definitions for peripheral PS7_UART_1 */
#define XPAR_PS7_UART_1_DEVICE_ID 0
#define XPAR_PS7_UART_1_BASEADDR 0xE0001000
#define XPAR_PS7_UART_1_HIGHADDR 0xE0001FFF
#define XPAR_PS7_UART_1_UART_CLK_FREQ_HZ 100000000
#define XPAR_PS7_UART_1_HAS_MODEM 0
```

Lab 0: Board Test

- Source code: helloworld.c

helloworld.c

```c
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"


int main()
{
    init_platform();

    print("Hello World\n\r");

    cleanup_platform();
    return 0;
}
```

Lab 0: Board Test

- Source code: platform.c

platform.c

```
void
init_platform()
{
    /*
     * If you want to run this example outside of SDK,
     * uncomment one of the following two lines and also #include "ps7_init.h"
     * or #include "ps7_init.h" at the top, depending on the target.
     * Make sure that the ps7/psu_init.c and ps7/psu_init.h files are included
     * along with this example source files for compilation.
     */
    /* ps7_init();*/
    /* psu_init();*/
    enable_caches();
    init_uart();
}
```

Lab 0: Board Test

- Source code: print.c

system.xml | *helloworld.c | *platform.c | print.c | outbyte.c | xuartps_hw.c

```
/* print.c -- print a string on the output device.

/*
 * print -- do a raw print of a string
 */
#include "xil_printf.h"

void print(const char *ptr)
{
#ifdef STDOUT_BASEADDRESS
  while (*ptr) {
    outbyte (*ptr++);
  }
#else
(void)ptr;
#endif
}
```

Lab 0: Board Test

- Source code: outbyte.c

system.xml | *helloworld.c | *platform.c | print.c | outbyte.c | xuartps_hw.c

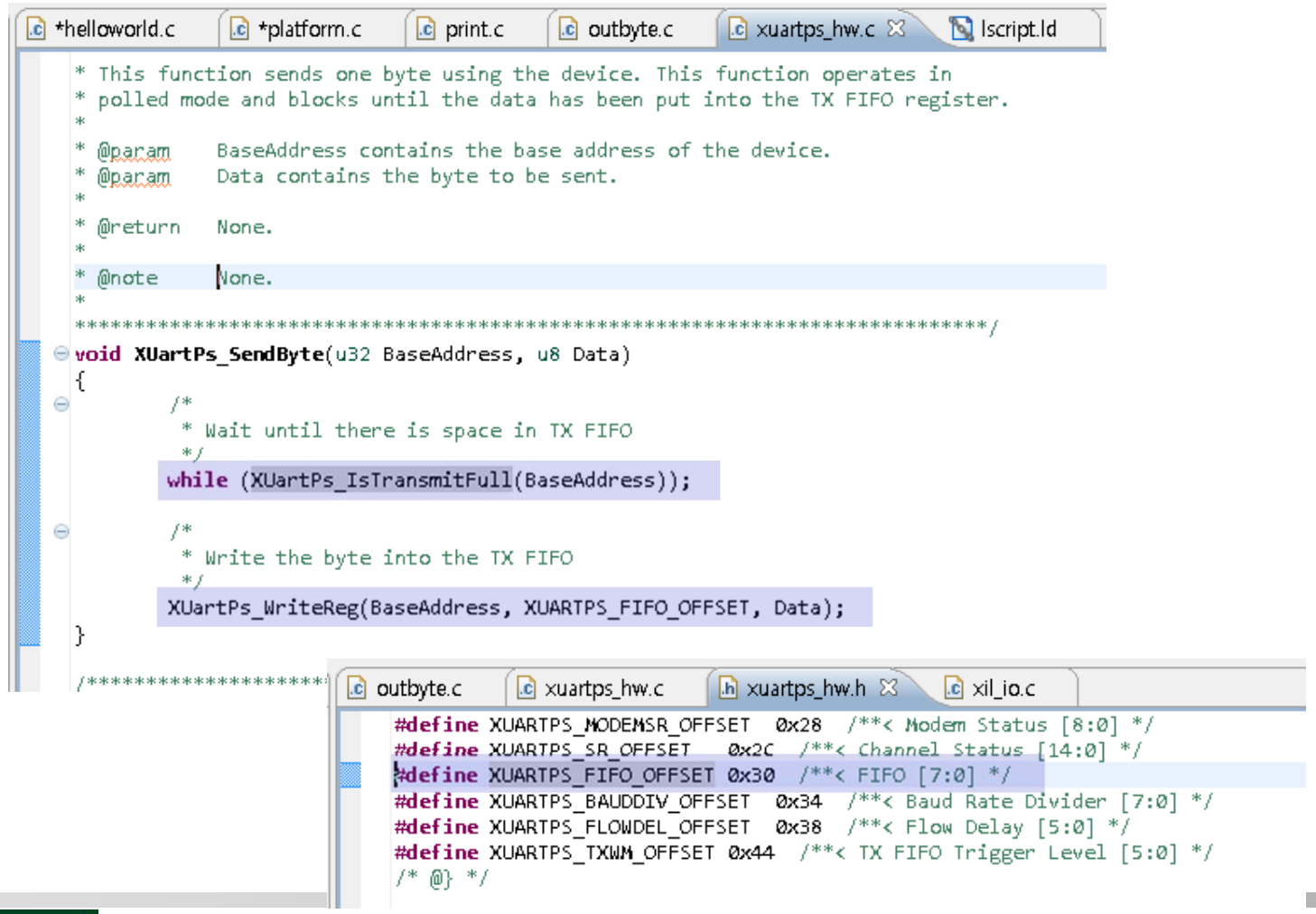
```
#include "xparameters.h"
#include "xuartps_hw.h"

#ifdef __cplusplus
extern "C" {
#endif
void outbyte(char c);

#ifdef __cplusplus
}
#endif

void outbyte(char c) {
    XUartPs_SendByte(STDOUT_BASEADDRESS, c);
}
```

outbyte.c | xuartps_hw.c | xuartps_hw.h | xil_io.c | xparameters.h

```
* CAUTION: This file is automatically generated by libgen.

#include "xparameters_ps.h"

#define STDIN_BASEADDRESS 0xE0001000
#define STDOUT_BASEADDRESS 0xE0001000

/******************************************************************/
```

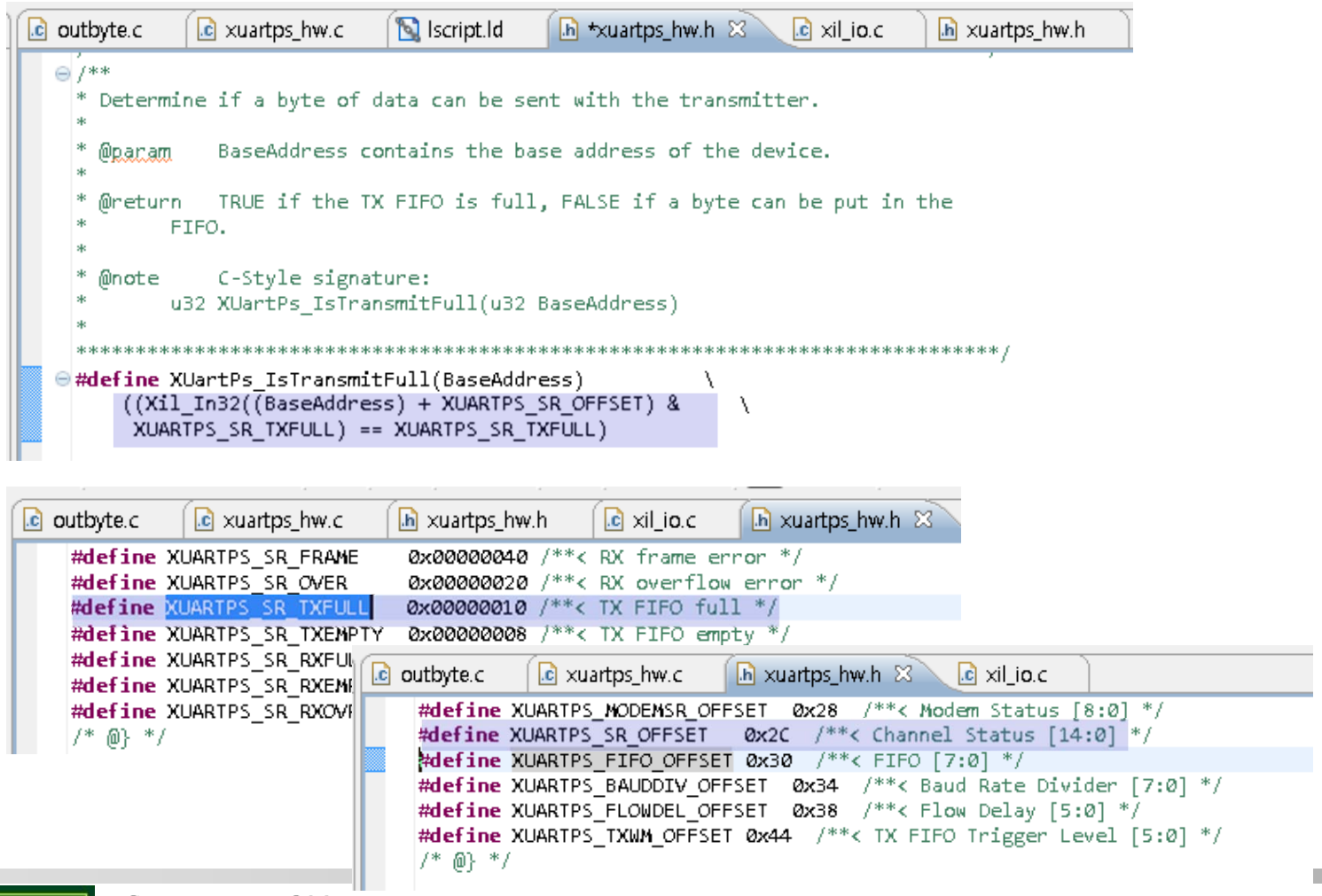
Lab 0: Board Test

- Source code: xuartps_hw.c



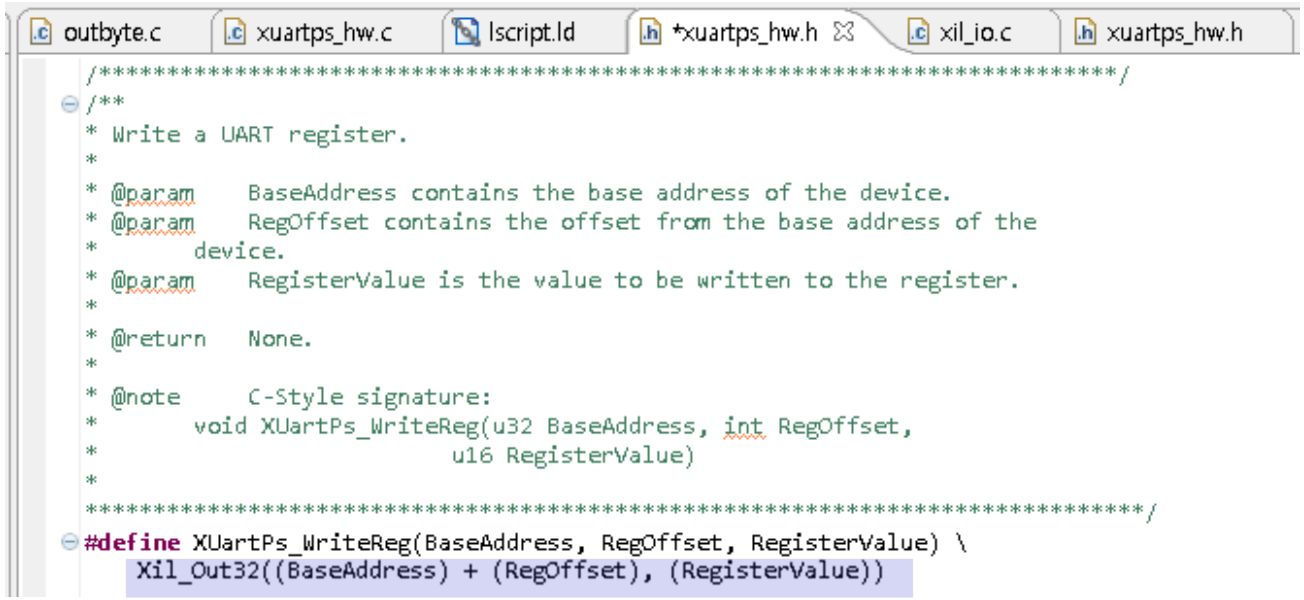
Lab 0: Board Test

- Source code: xuartps_hw.h



Lab 0: Board Test

- Source code: xuartps_hw.h (cont'd)

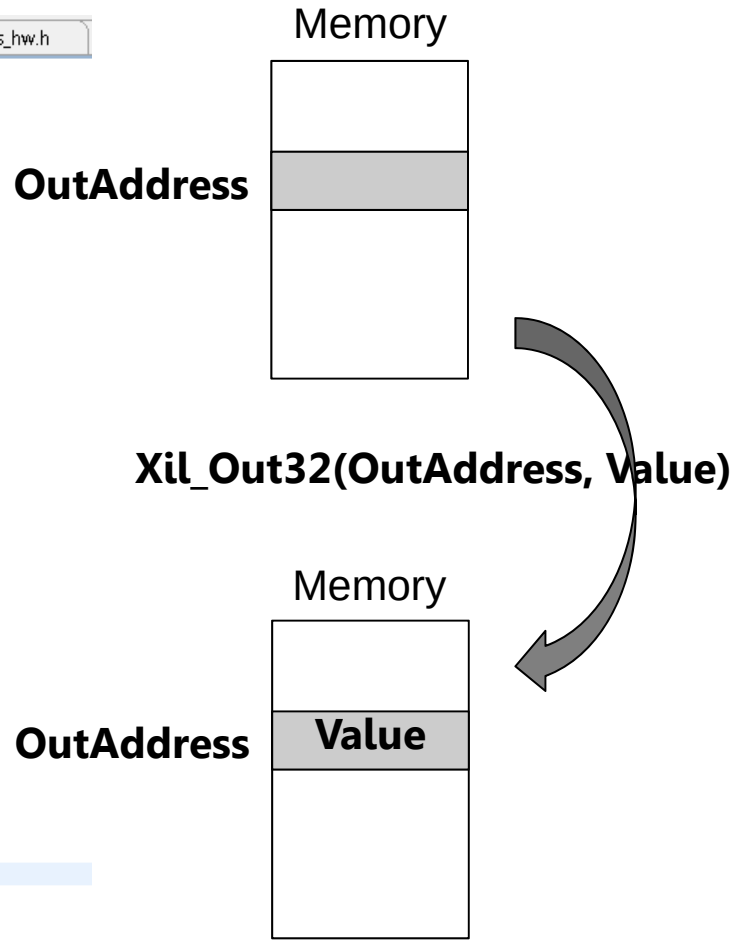


Lab 0: Board Test

- Source code: xil_io.c

outbyte.c | xuartps_hw.c | lscript.ld | *xuartps_hw.h | xil_io.c | xuartps_hw.h

```
/****************************************************************************/
/**
*
* Performs an output operation for a 16-bit memory location by writing the
* specified Value to the the specified address.
*
* @param    OutAddress contains the address to perform the output operation
*           at.
* @param    Value contains the Value to be output at the specified address.
*
* @return   None.
*
* @note     None.
*
****************************************************************************/
void Xil_Out16(u32 OutAddress, u16 Value)
{
    *(volatile u16 *) OutAddress = Value;
}

/****************************************************************************/
/**
*
* Performs an output operation for a 32-bit memory location by writing the
* specified Value to the the specified address.
*
* @param    OutAddress contains the address to perform the output operation
*           at.
* @param    Value contains the Value to be output at the specified address.
*
* @return   None.
*
* @note     None.
*
****************************************************************************/
void Xil_Out32(u32 OutAddress, u32 Value)
{
    *(volatile u32 *) OutAddress = Value;
}

/****************************************************************************/
```

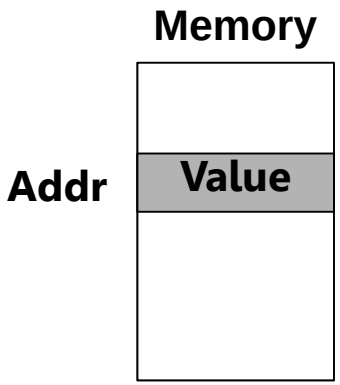


Lab 0: Board Test

- Source code: xil_io.c (cont'd)

outbyte.c | xuartps_hw.c | lscript.ld | *xuartps_hw.h | xil_io.c | xuartps_hw.h

```
/****************************************************************************/
/**
*
* Performs an input operation for a 16-bit memory location by reading from the
* specified address and returning the Value read from that address.
*
* @param    Addr contains the address to perform the input operation
*       at.
*
* @return   The Value read from the specified input address.
*
* @note     None.
*
****************************************************************************/
u16 Xil_In16(u32 Addr)
{
    return *(volatile u16 *) Addr;
}

/****************************************************************************/
/**
*
* Performs an input operation for a 32-bit memory location by reading from the
* specified address and returning the Value read from that address.
*
* @param    Addr contains the address to perform the input operation
*       at.
*
* @return   The Value read from the specified input address.
*
* @note     None.
*
****************************************************************************/
u32 Xil_In32(u32 Addr)
{
    return *(volatile u32 *) Addr;
}
```



Xil_In32(Addr) returns **Value**

References

- Zynq-7000 All Programmable, technical reference manual, Xilinx UG585

Appendix

- UART registers

B.33 UART Controller (UART)

Module Name	UART Controller (UART)
Software Name	XUARTPS
Base Address	0xE0000000 uart0 0xE0001000 uart1
Description	Universal Asynchronous Receiver Transmitter
Vendor Info	Cadence UART

Register Summary

Register Name	Address	Width	Type	Reset Value	Description
Control_reg0	0x00000000	32	mixed	0x00000128	UART Control Register
mode_reg0	0x00000004	32	mixed	0x00000000	UART Mode Register
Intrpt_en_reg0	0x00000008	32	mixed	0x00000000	Interrupt Enable Register
Intrpt_dis_reg0	0x0000000C	32	mixed	0x00000000	Interrupt Disable Register
Intrpt_mask_reg0	0x00000010	32	ro	0x00000000	Interrupt Mask Register
Chnl_int_sts_reg0	0x00000014	32	wtc	0x00000000	Channel Interrupt Status Register
Baud_rate_gen_reg0	0x00000018	32	mixed	0x0000028B	Baud Rate Generator Register.
Rcvr_timeout_reg0	0x0000001C	32	mixed	0x00000000	Receiver Timeout Register
Rcvr_FIFO_trigger_level0	0x00000020	32	mixed	0x00000020	Receiver FIFO Trigger Level Register
Modem_ctrl_reg0	0x00000024	32	mixed	0x00000000	Modem Control Register
Modem_sts_reg0	0x00000028	32	mixed	x	Modem Status Register
Channel_sts_reg0	0x0000002C	32	ro	0x00000000	Channel Status Register
TX_RX_FIFO0	0x00000030	32	mixed	0x00000000	Transmit and Receive FIFO
Baud_rate_divider_reg0	0x00000034	32	mixed	0x0000000F	Baud Rate Divider Register
Flow_delay_reg0	0x00000038	32	mixed	0x00000000	Flow Control Delay Register
Tx_FIFO_trigger_level0	0x00000044	32	mixed	0x00000020	Transmitter FIFO Trigger Level Register

Appendix

- Tx/Rx FIFO register

Field Name	Bits	Type	Reset Value	Description
reserved	31:8	ro	0x0	Reserved, read as zero, ignored on write.
FIFO	7:0	rw	0x0	Operates as Tx FIFO and Rx FIFO.

- Channel status register

reserved	6	ro	0x0	Reserved. Do not modify.
reserved	5	ro	0x0	Reserved. Do not modify.
TFUL (TXFULL)	4	ro	0x0	Transmitter FIFO Full continuous status: 0: Tx FIFO is not full 1: Tx FIFO is full
TEMPTY (TXEMPTY)	3	ro	0x0	Transmitter FIFO Empty continuous status: 0: Tx FIFO is not empty 1: Tx FIFO is empty