SoC-Platforms & Design Methods for SoC Project and Labs

COE838: Systems on Chip Design
http://www.ecb.torontomu.ca/~courses/coe838/
Dr. Gul N. Khan
http://www.ecb.torontomu.ca/~gnkhan
Elect., Computer & Biomedical Engineering
Toronto Metropolitan University

Overview

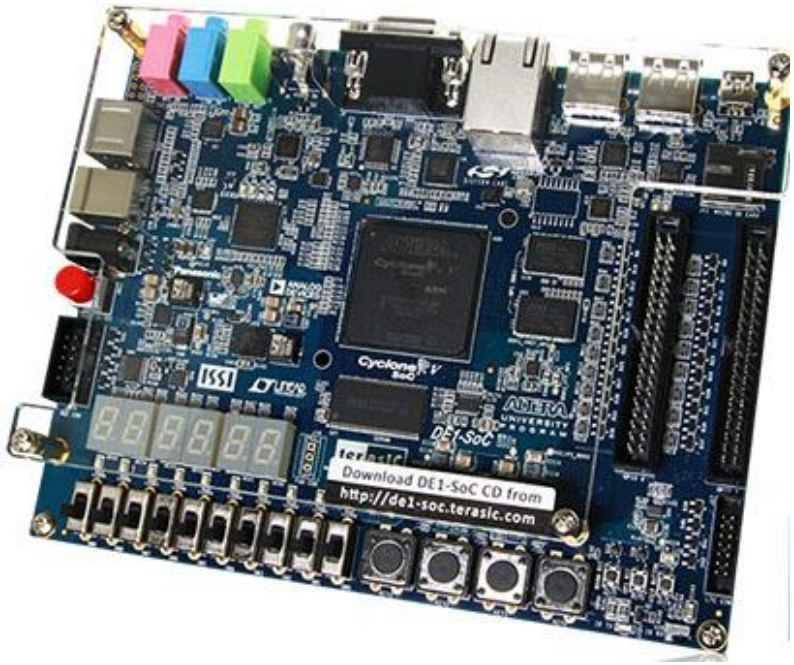
- SoC Platforms --Xilinx Zynq-7000 SoC platform.
- Intel (Altera) Cyclone-V SoC platforms, DE1-SoC.
- DE1-SoC Bootup Process and Embedded OS.
- Bootup Process and Device Tree.
- Root File System and Yocto Pre-built Packages.

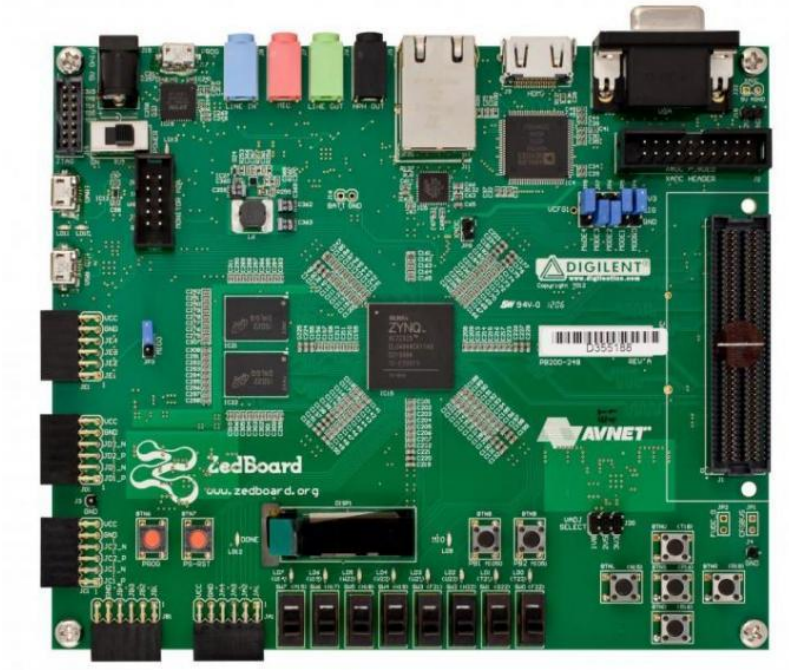
SoC Platforms, DE1-SoC Manuals and Material from Chapter 3 of the Text by M.J. Flynn

SoC Platforms

- **SoC Platforms can be of two categories:**
  - SoC FPGA
  - Commercial and Specific SoC platforms employed by Apple, Samsung, Siemens, etc.
- **SoC FPGA Platforms:**
  - Zynq-7000 by Xilinx
  - Cyclone-V and Arria-V platforms by Intel (Altera)

Main HPS/FPGA Systems





Intel (Altera) DE1-SoC

Cyclone V

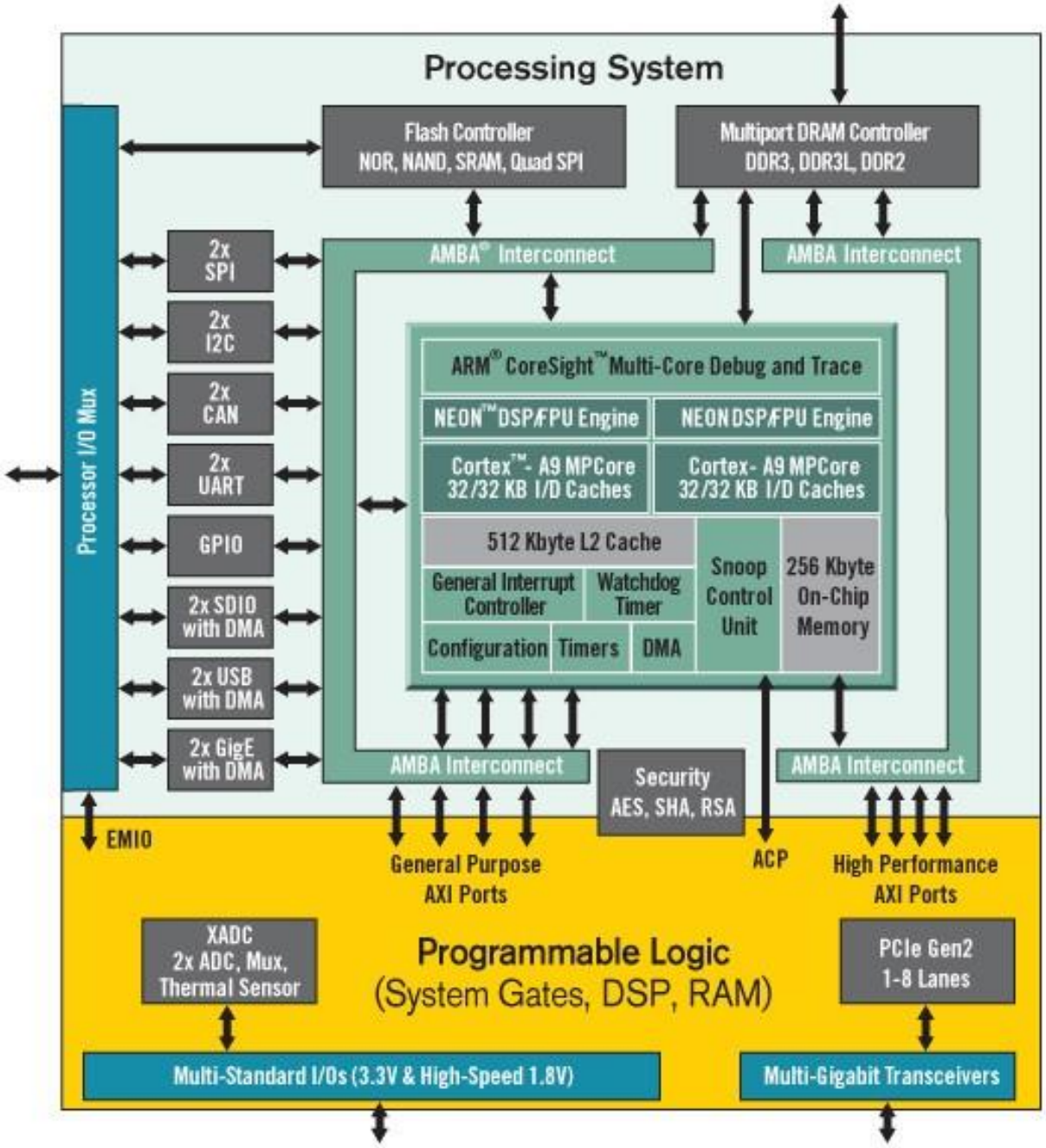
Arria V, etc

Xilinx Zedboard

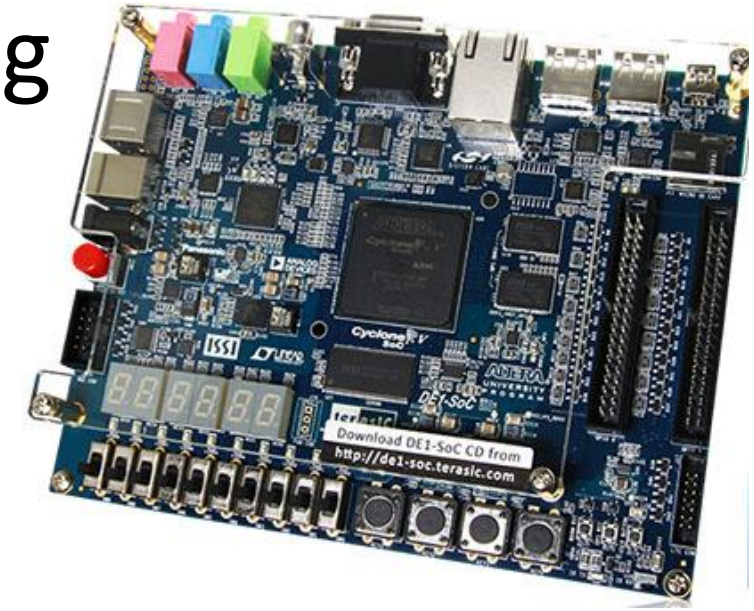
Zynq 7000

Zynq 7XXX etc

Zynq SoC


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

What we'll be covering



- HPS/FPGA Systems
  - Intel Altera DE1-SoC
  - ARM Cortex-A9 and Cyclone V FPGA
  - Interfacing, emulating, simulating real HW/SW systems
  - Integrating IPs in your system as accelerators
  - Creating software to execute on your CPU and offloading to accelerators

Intel (Altera) SoC Family FPGA

SoC Family

Family	KLE	Block Memory Bits (Mb)	Var. Prec. Multiplier Blocks	Max. FPGA User I/Os	HPS Dedicated I/Os	Max. Transceivers (GP)	Per-Transceiver Max. Data Rate (Gbps)	SoC Hard Memory Controller	FPGA Hard Memory Controllers
Cyclone V SoC	25	1.4	36	145	181	6	3	1	1
	40	2.7	58	145	181	6	3	1	1
	85	4.0	87	288	181	9	5	1	1
	110	5.6	112	288	181	9	5	1	1
Arria V SoC	350	17.3	809	528	208	30 / 16	6 / 10	1	3
	460	22.8	1,068	528	208	30 / 16	6 / 10	1	3

		Non-Transceiver Devices (FPGA User I/Os)			Transceiver Devices (FPGA User I/Os, Transceivers)			
Family	KLE	U484-WB 19x19	U672-WB 23x23	F896-WB 31x31	U672-WB 23x23 (I/O, 3G/5G)	F896-WB 31x31 (I/O, 3G/5G)	F896-FC 31x31 (I/O, 6G, 10G)	F1152-FC 35x35 (I/O, 6G, 10G)
Cyclone V SoC	25	66	145	–	145, 6	–	–	–
	40	66	145	–	145, 6	–	–	–
	85	66	145	288	145, 6	288, 9	–	–
	110	66	145	288	145, 6	288, 9	–	–
Arria V SoC	350	–	–	–	–	–	170, 12, 4	350, 18, 8
	460	–	–	–	–	–	170, 12, 4	350, 18, 8
HPS I/O		161	181	181	181	181	208	208

Cyclone-V SoC FPGA

Cyclone-V FPGA family is based on 1.1-V, 28nm technology
High performance designs and SoC prototyping

	5CSXC2	5CSXC4	5CSXC5	5CSXC6
ALMs	9,434	15,094	32,075	41,509
LEs (K)	25	40	85	110
Registers	37,736	60,376	128,300	166,036
M10K memory blocks	140	224	397	514
M10K memory (Kb)	1,400	2,240	3,972	5,140
MLAB memory (Kb)	138	220	480	621
Variable-precision DSP blocks	36	58	87	112
18 x 18 multipliers	72	116	174	224
Processor cores (ARM Cortex-A9)	Dual	Dual	Dual	Dual
Global clock networks	16			
PLLs[2] (FPGA)	4	5	6	6
PLLs[2] (HPS)	3	3	3	3
Transceiver count (3.125 Gbps)	6	6	9	9
PCIe hard IP blocks (Gen1 x4)	2	2	2	2
GPIOs (FPGA)	145	145	288	288
GPIOs (HPS)	188	188	188	188
Hard memory controllers[4] (FPGA)	1	1	1	1
Hard memory controllers[4] (HPS)	1	1	1	1

Arria-V SX SoC FPGA

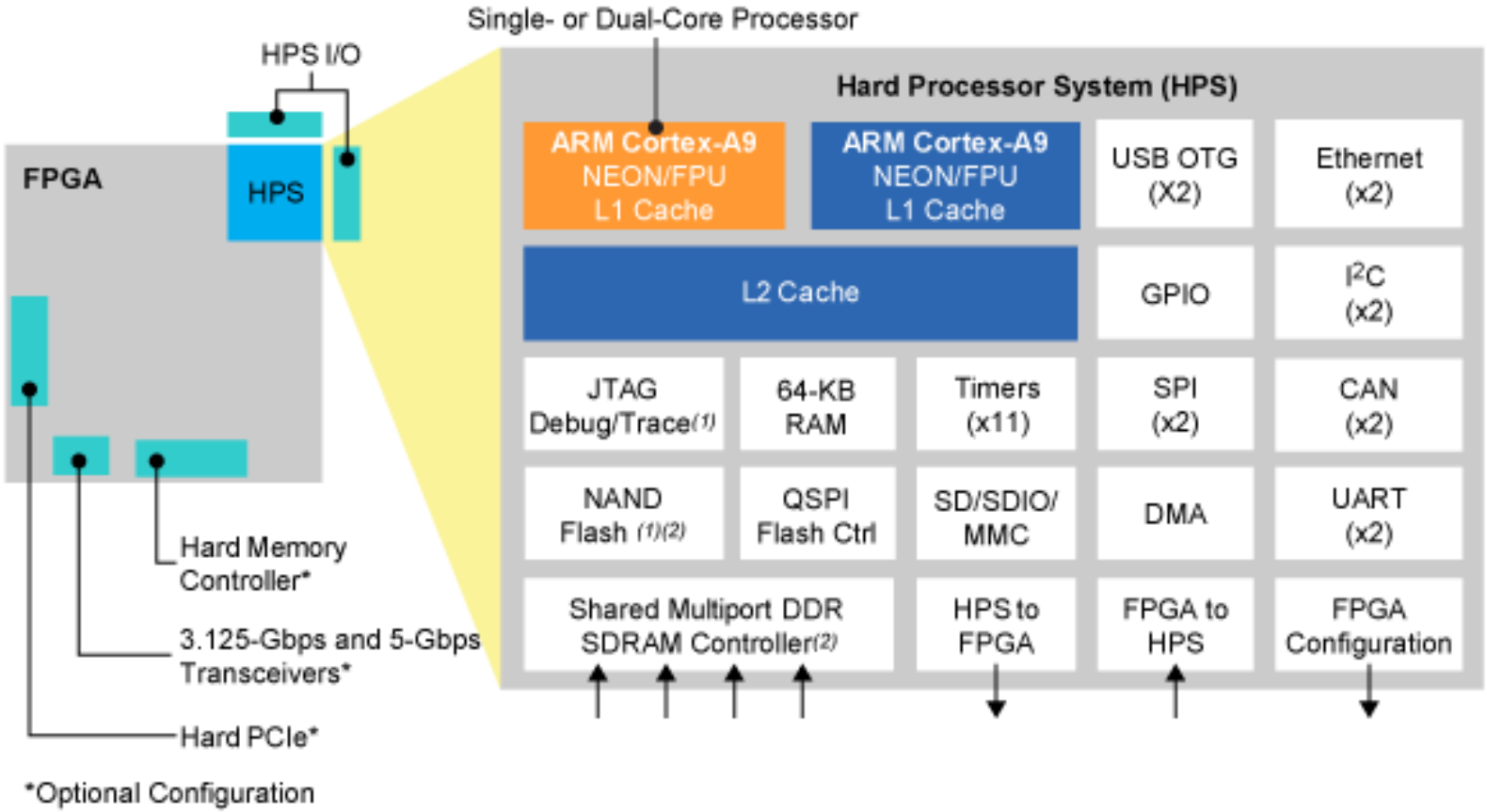
Arria-V FPGA family is based on 1.1V, 28nm

It can prototype/implement ARM Cortex A9 CPU based SoCs

	5ASXB3	5ASXB5
ALMs	132,075	174,340
LEs (K)	350	462
Registers	528,300	697,360
M10K memory blocks	1,729	2,282
M10K memory (Kb)	17,288	22,820
MLAB memory (Kb)	2,014	2,658
Variable-precision DSP blocks	809	1,068
18 x 18 multipliers	1,618	2,186
Processor cores (ARM Cortex-A9)	Dual	Dual
Global clock networks	16	
PLLs[2] (FPGA)	10	14
PLLs[2] (HPS)	3	3

Cyclone-V SoC Platform

Cyclone-V SoCs reduce system power, cost, and board size while increasing system performance by integrating discrete processor, FPGA, and DSP functions into a single, user customizable ARM-based (SoC)

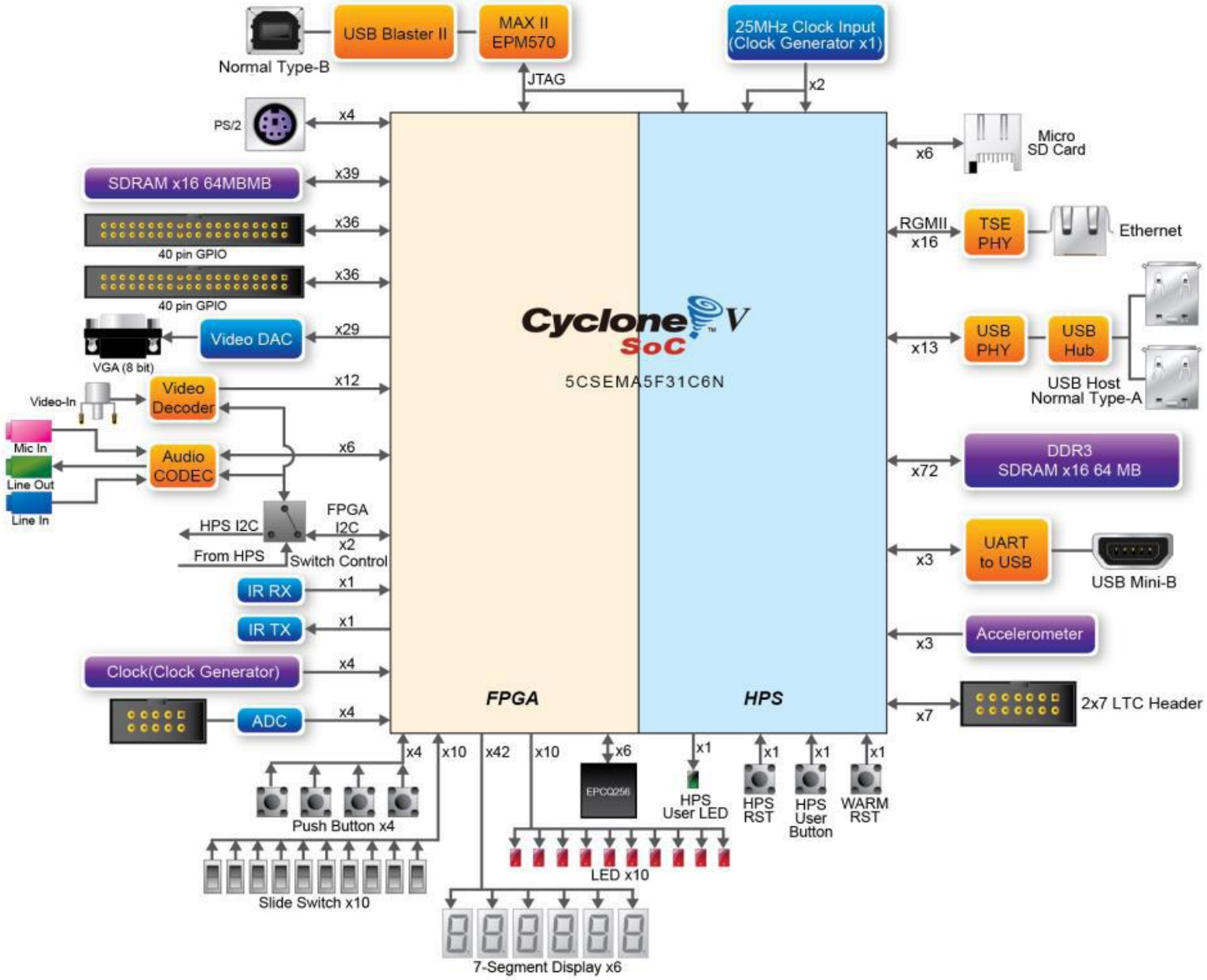


https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=836#contents

DE1-SoC

- Cyclone-V SoC 5CSEMA5F31 Device
- Dual-core ARM Cortex-A9 (HPS)
- 85K programmable logic elements
- 4,450Kbits embedded memory
- 6 fractional PLLs
- 2 hard memory controllers
- Quad serial configuration device –EPCQ256
- On-board USB-Blaster II
- 64MB (32Mx16) SDRAM on FPGA
- 1GB (2x256Mx16) DDR3 SDRAM on HPS
- Micro SD card socket on HPS

DE1-SoC

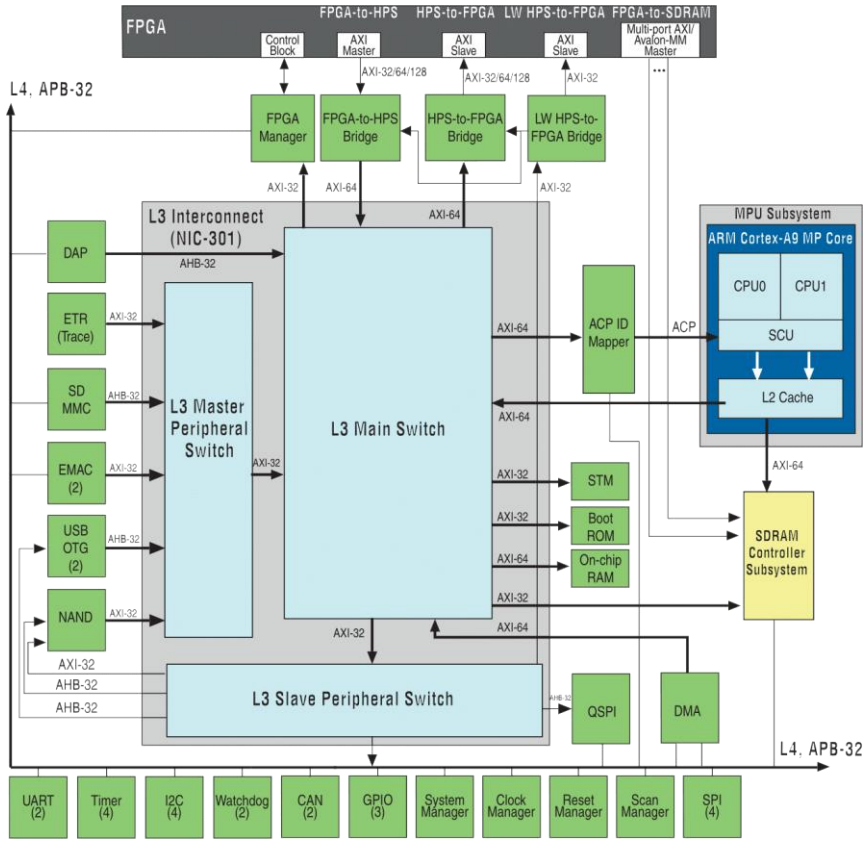

USB Blaster II
MAX II EPM570
Normal Type-B
JTAG
25MHz Clock Input (Clock Generator x1)
x2
PS/2
x4
SDRAM x16 64MBMB
x39
40 pin GPIO
x36
40 pin GPIO
x36
VGA (8 bit)
Video DAC
x29
Video-In
Video Decoder
x12
Mic In
Line Out
Line In
Audio CODEC
x6
HPS I2C
FPGA I2C
x2
From HPS
Switch Control
IR RX
x1
IR TX
x1
Clock(Clock Generator)
x4
ADC
x4
Cyclone V SoC
5CSEMA5F31C6N
FPGA
HPS
x6
Micro SD Card
RGMII
x16
TSE PHY
Ethernet
x13
USB PHY
USB Hub
USB Host Normal Type-A
x72
DDR3 SDRAM x16 64 MB
x3
UART to USB
USB Mini-B
x3
Accelerometer
x7
2x7 LTC Header
x4
x10
x42
x10
x6
x1
x1
x1
x1
Push Button x4
Slide Switch x10
LED x10
HPS User LED
HPS RST
HPS User Button
WARM RST
7-Segment Display x6

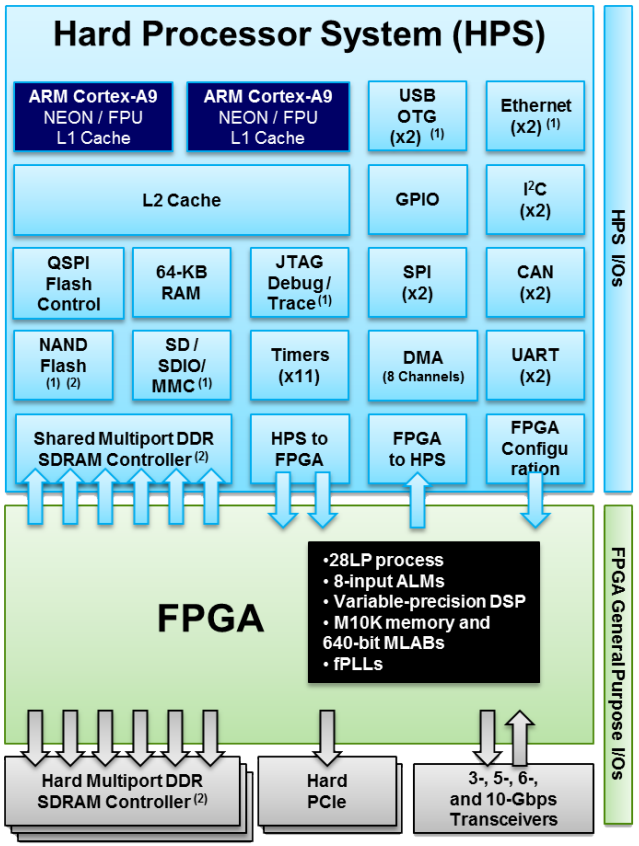
Cyclone-V HPS

800MHz/1.05 GHz, **dual-core ARM Cortex-A9 processor** including L1 Inst. & data caches (32 KB each), FPU and NEON media engine.

- 512 KB of shared L2 cache with **error correction code** (ECC) support
- 64 KB of scratch RAM with ECC support
- Multiport SDRAM controller with support for DDR2, DDR3, etc.
- QSPI and NAND flash controller with DMA
- SD/SDIO/MMC controller with DMA
- 2x 10/100/1000 Ethernet media access control (MAC) with DMA
- 2x USB On-The-Go (OTG) controller with DMA
- 4x I^2C controller and 2x UART
- 2x (SPI) master peripherals, 2x SPI slave peripherals
- Up to 134 general-purpose I/O (GPIO)
- 7x general-purpose timers and 4 watchdog timers
- Warm and Cold reset

Inside the SoC





HPS-to-FPGA Interconnect

HPS and the FPGA can operate independently.

Tightly coupled via AMBA bridges:

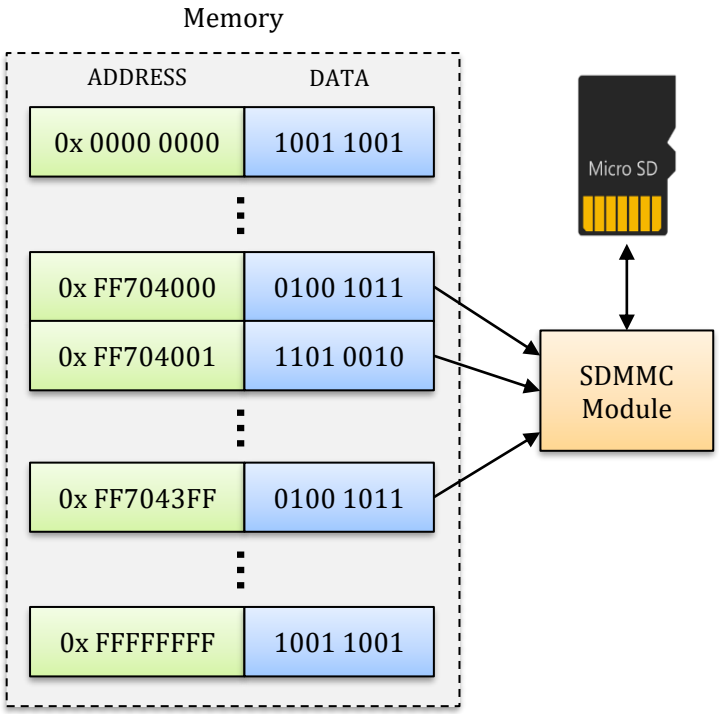
- IP bus masters in FPGA fabric have access to HPS bus slaves.
- HPS bus masters have access to bus slaves in FPGA fabric.
- Both bridges are AMBA AXI-3 compliant and support simultaneous read and write transactions.

- Additional 32-bit light-weight HPS-to-FPGA bridge with low latency i/f between HPS and peripherals in FPGA fabric.
- Six FPGA masters can share the HPS SDRAM controller with ARM Cortex A9 processor.
- A9 processor can be used to configure FPGA fabric under program control via a dedicated 32-bit port.

HPS-to-FPGA Interconnect

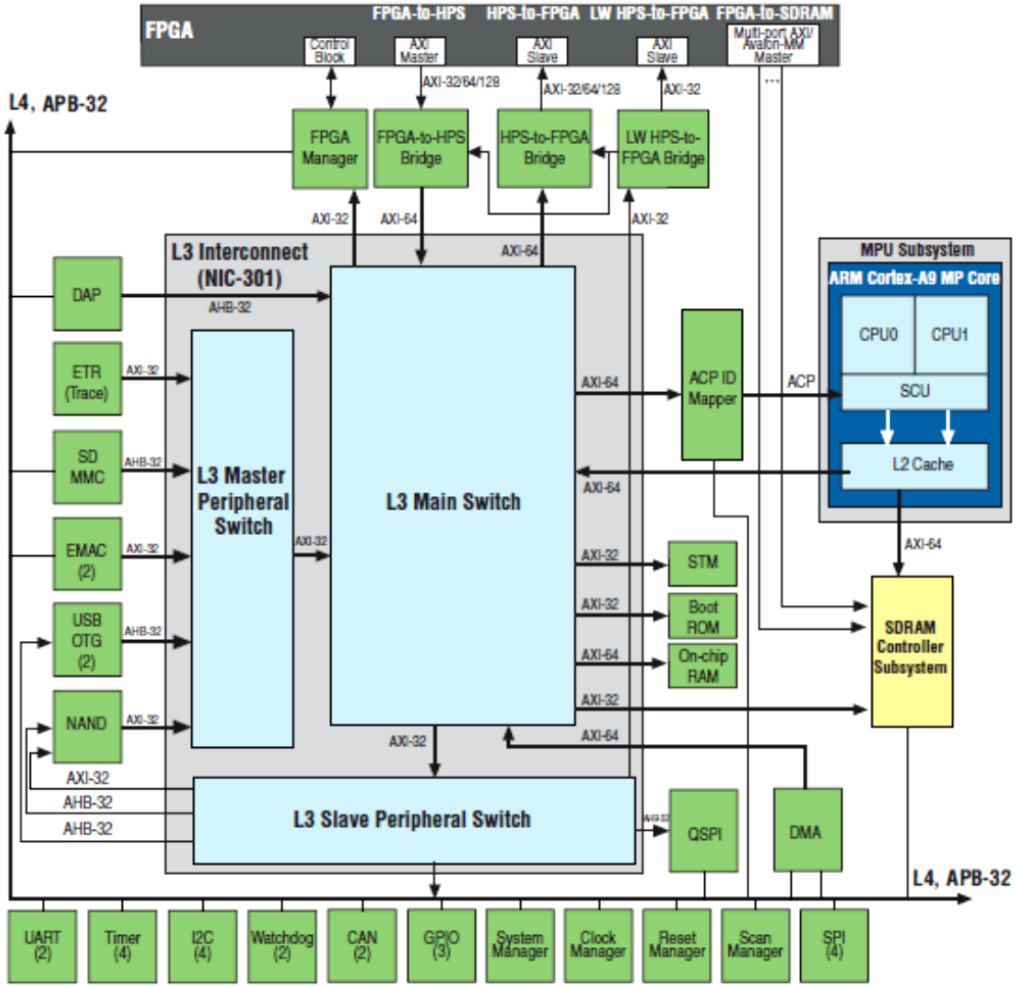
- HPS-to-FPGA: configurable 32-, 64-, or 128-bit AMBA AXI interface optimized for high bandwidth
- FPGA-to-HPS: configurable 32-, 64-, or 128-bit AMBA AXI interface optimized for high bandwidth
- Lightweight HPS-to-FPGA: 32-bit AMBA AXI interface optimized for low latency
- FPGA-to-HPS SDRAM controller: Configurable multi-port interfaces with 6 command ports, 4x 64-bit read data ports and 4x 64-bit write data ports
- 32-bit FPGA configuration manager

Cyclone V HPS Memory Map



0x C0000000 - 0x FBFFFFFF	Slaves via HP AXI Bridge
0x FF200000 - 0x FF3FFFFF	Slaves via LW AXI Bridge
0x FF704000 - 0x FF7043FF	SDMMC Module
0x FF705000 - 0x FF7050FF	QSPI Flash Controller Module
0x FF708000 - 0x FF70A07F	GPIO Module
0x FFB00000 - 0x FFB7FFFF	USB OTG Controller
0x FFFE0000 - 0x FFE01FFF	DMA Module
0x FFFD0000 - 0x FFFDFFFF	Boot ROM Module
0x FFFD0000 - 0x FFFFFFFF	On-Chip RAM Module

AXI / HPS-FPGA Interconnects

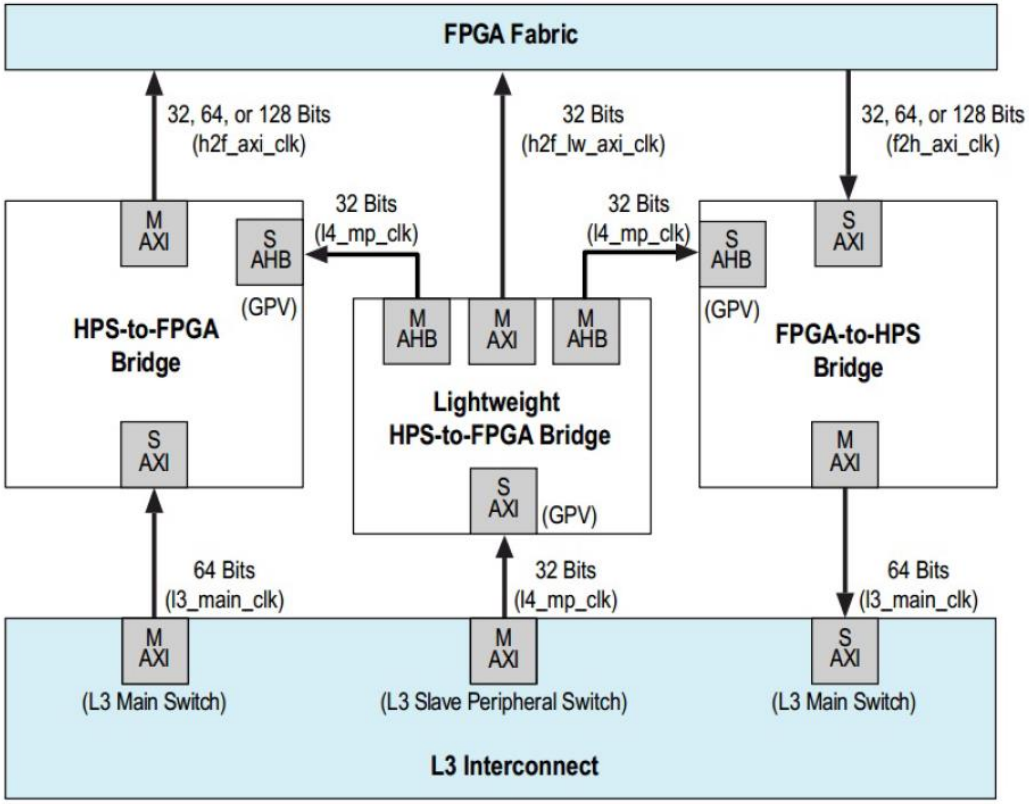


HPS-to-FPGA: Configurable 32, 64 or 128bit AMBA AXI interface optimized for high bandwidth.

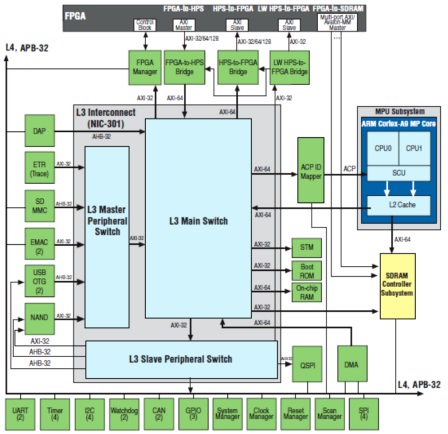
FPGA-to-HPS: --- “ ---- Opposite direction (Mainly used for ACP: Accelerator Coherency Port) which is an AXI 64-bit slave port.

Light weight HPS-to-FPGA: 32-bit AMBA AXI interface optimized for low latency.

HPS – FPGA Bridges

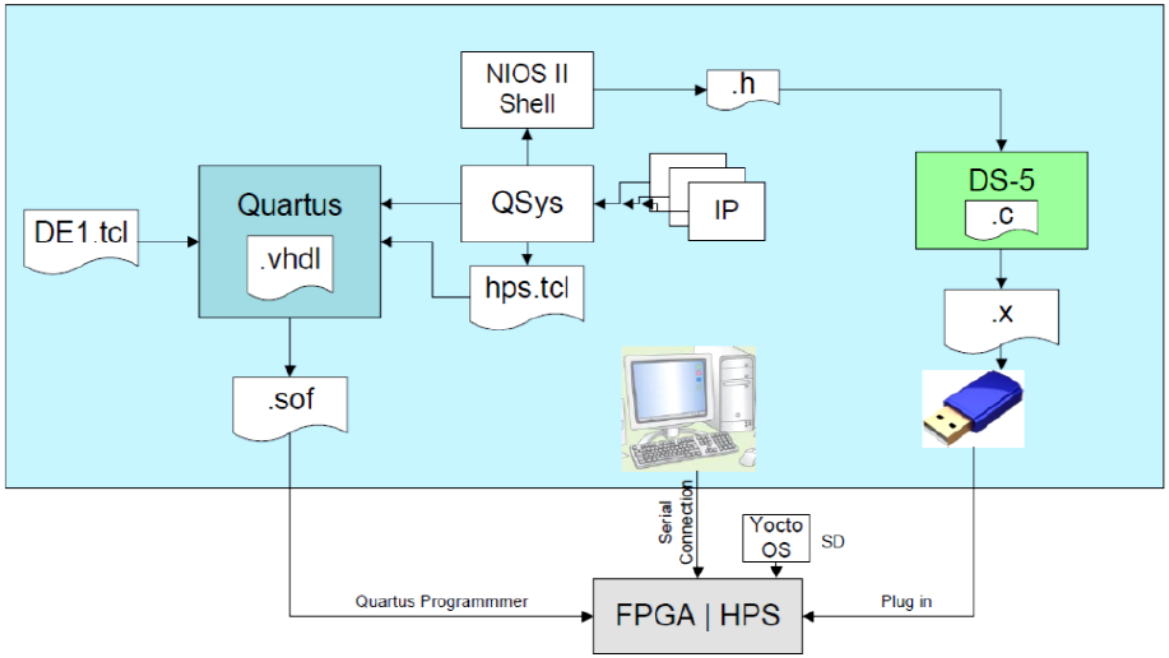


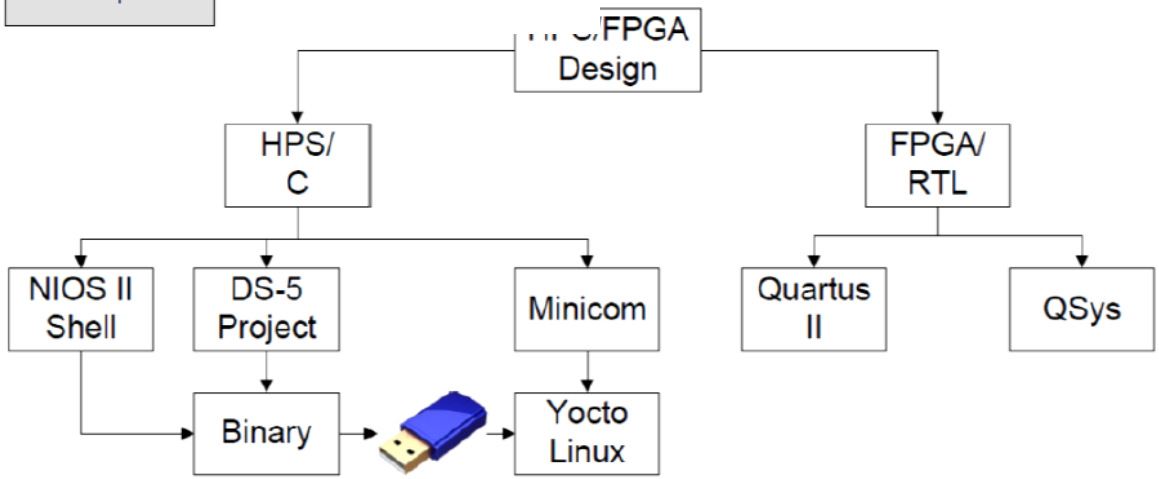
- To modify data and clock signals to support transportation (protocols, clocking, etc.) between components



Region Name	Description	Base Address	Size
FPGA Slaves	For accessing FPGA slaves connected to the h2f bridge	0xC0000000	960MB
HPS Peripherals	Accessing slaves directly connected to the HPS	0xFC000000	64MB
Lightweight FPGA Slaves	Accessing slaves connected to the lwh2f bridge	0xFF200000	2MB

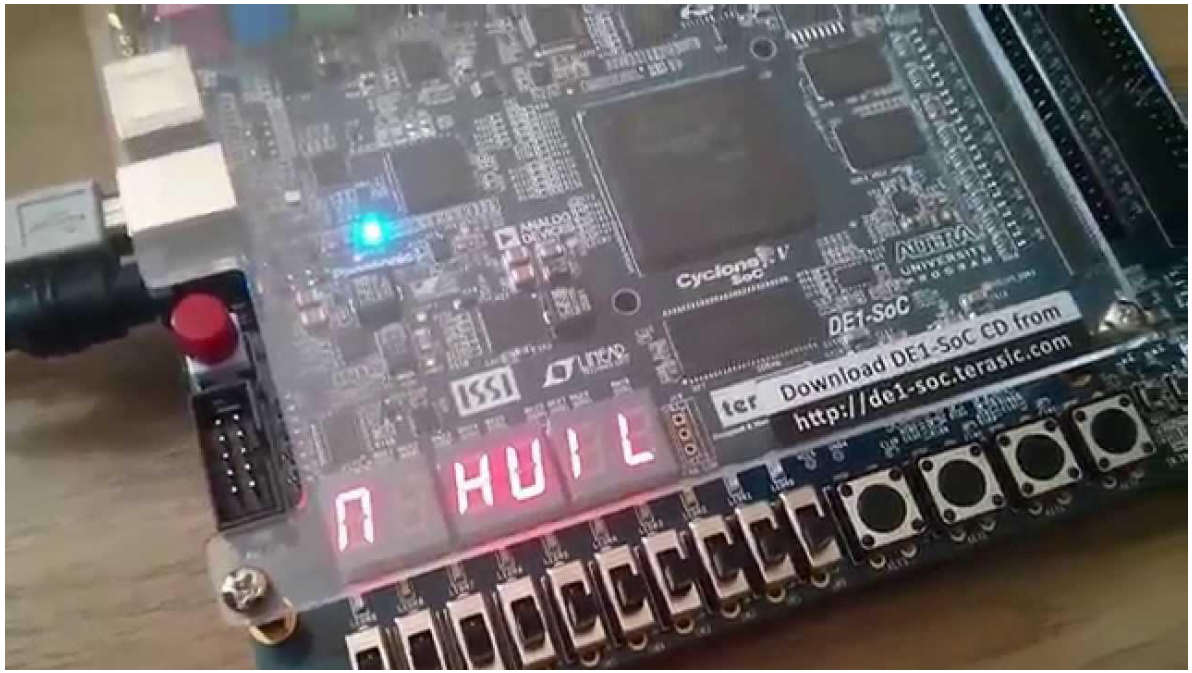
Lab 3: DE1-SoC Tutorial





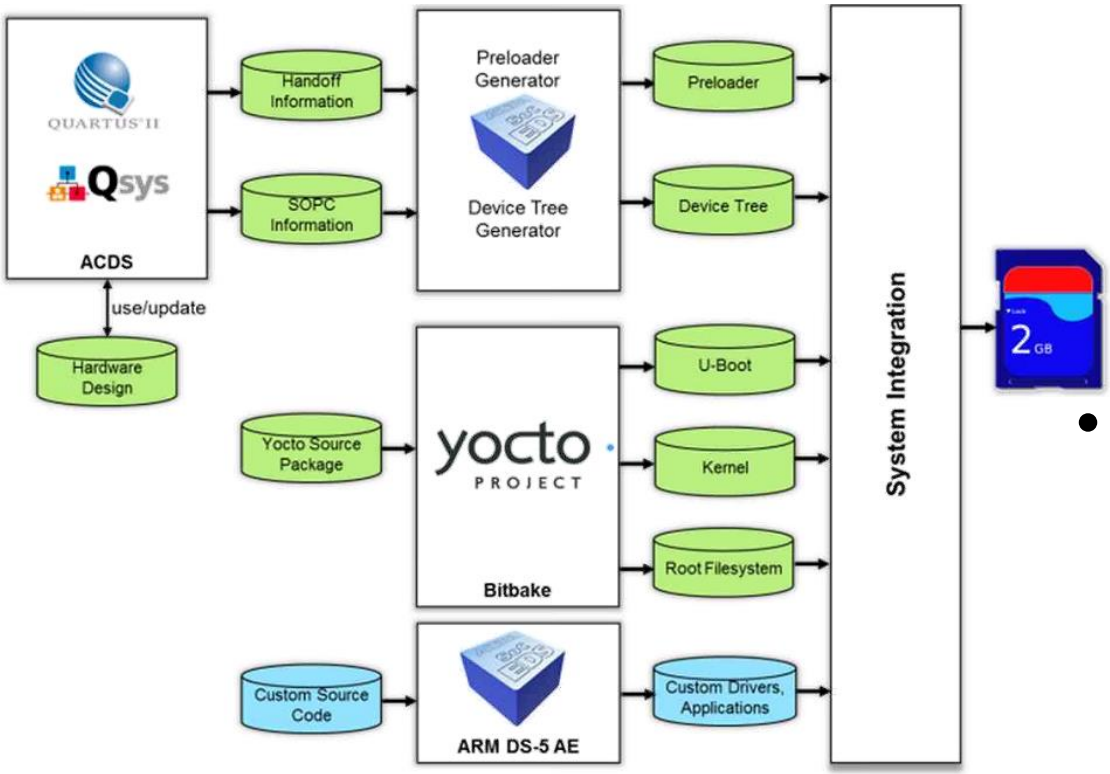
Lab3 – DE1-SoC Tutorial

Creating a HPS/FPGA which controls 7-seg and LEDs on FPGA through software on HPS



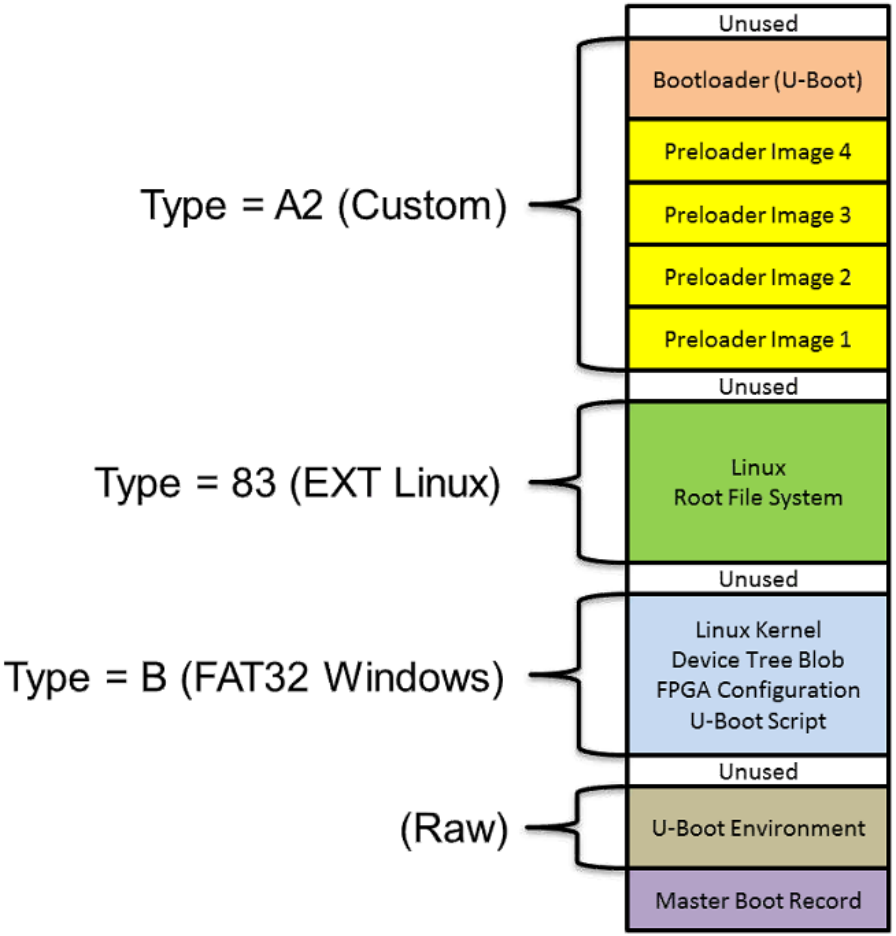
https://www.youtube.com/watch?v=BHg5CjTeftY

Intel (Altera) DE1-SoC & Embedded OS



- Intel (Altera) has their own tools that generate files required for the OS and its boot process
- Caters to the underlying hardware which you generate with Qsys.

SD Card Partitions and Contents



- The SD card contains data pertaining to the operating system and underlying architecture

Type 83 is Linux data partition.

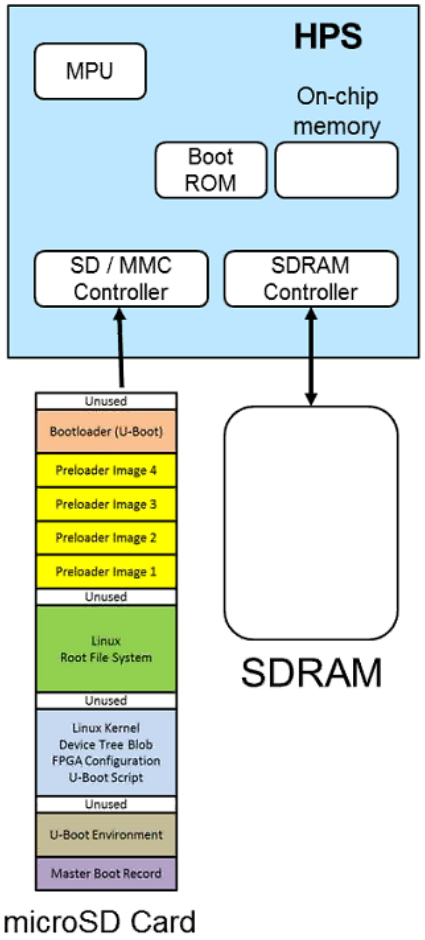
  - If viewing from Windows machine. It can not view many of these files
    - May be FAT32 files only

Boot up Process

5 necessary elements for SoC based Linux (embedded OS)

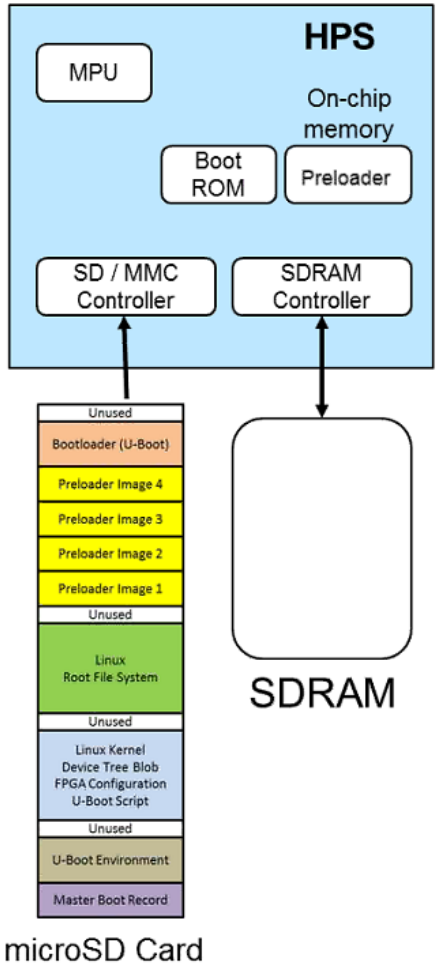
1. Pre-loader
2. Device Tree
3. Boot-loader
4. Kernel
5. Root File System

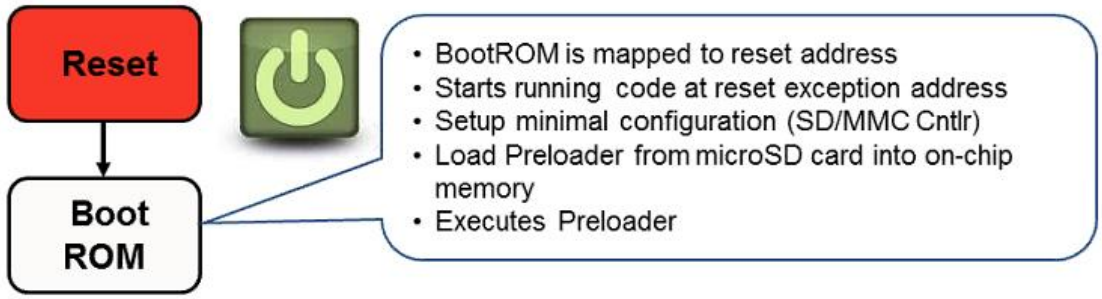
Boot up Process



- Assuming SoC **off**, and microSD contains OS related environment that was flashed to the card.

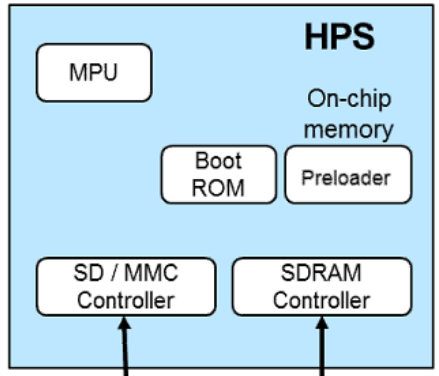
Boot up Process

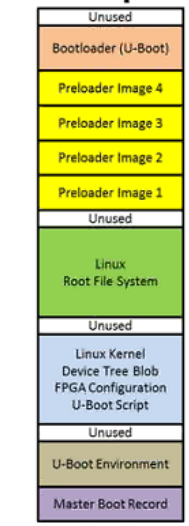




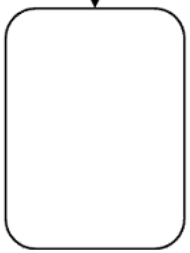
- When powered on, **Boot ROM** (BR) moves pre-loader from SD card to on-chip memory
- BR sets up minimal configuration
- BR is not user accessible
- Specifies where to access pre-loader

Boot up Process

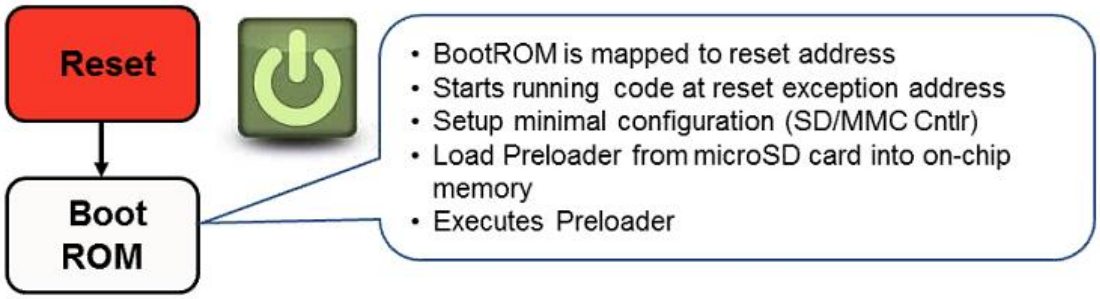




microSD Card

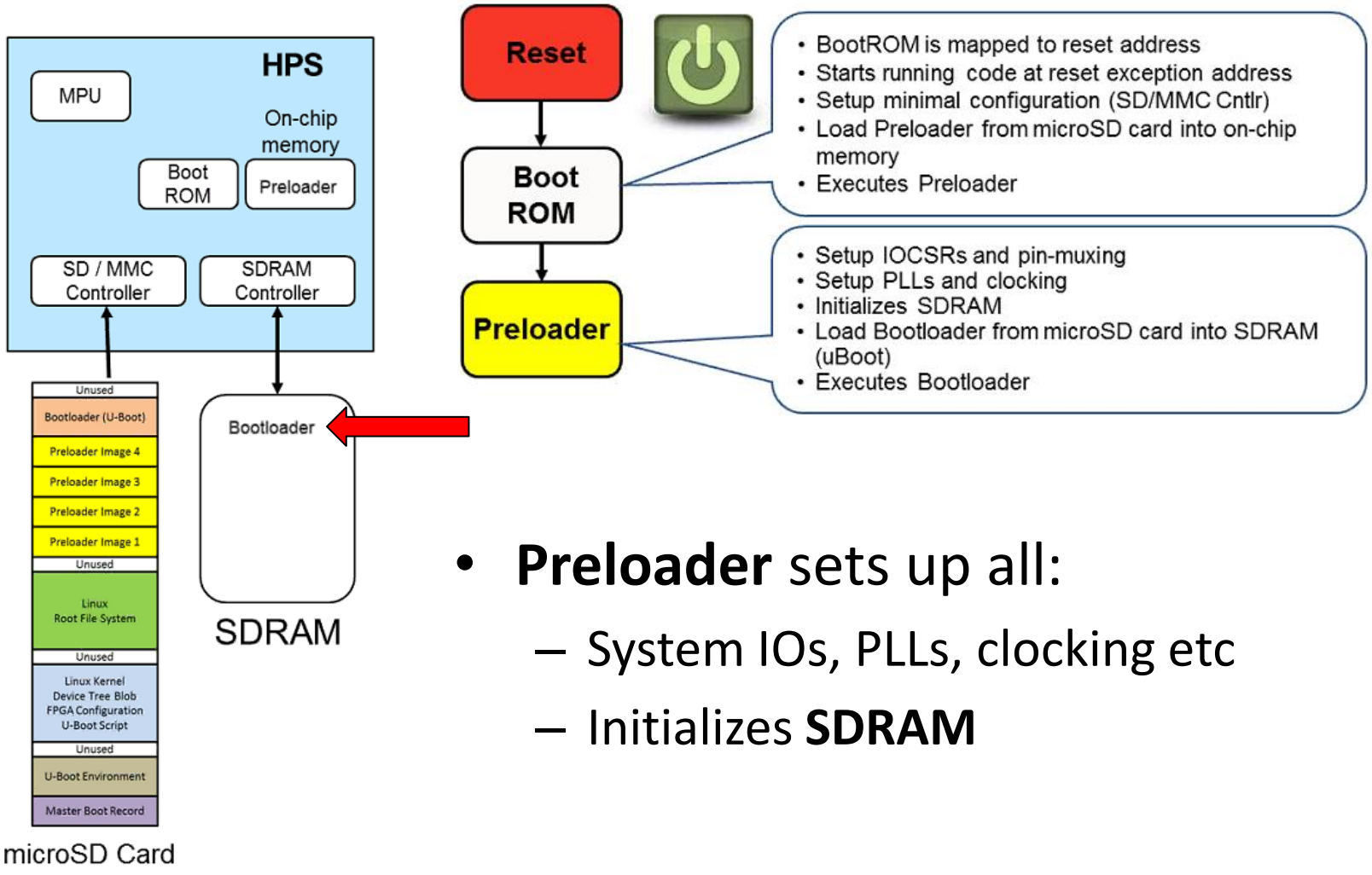


SDRAM



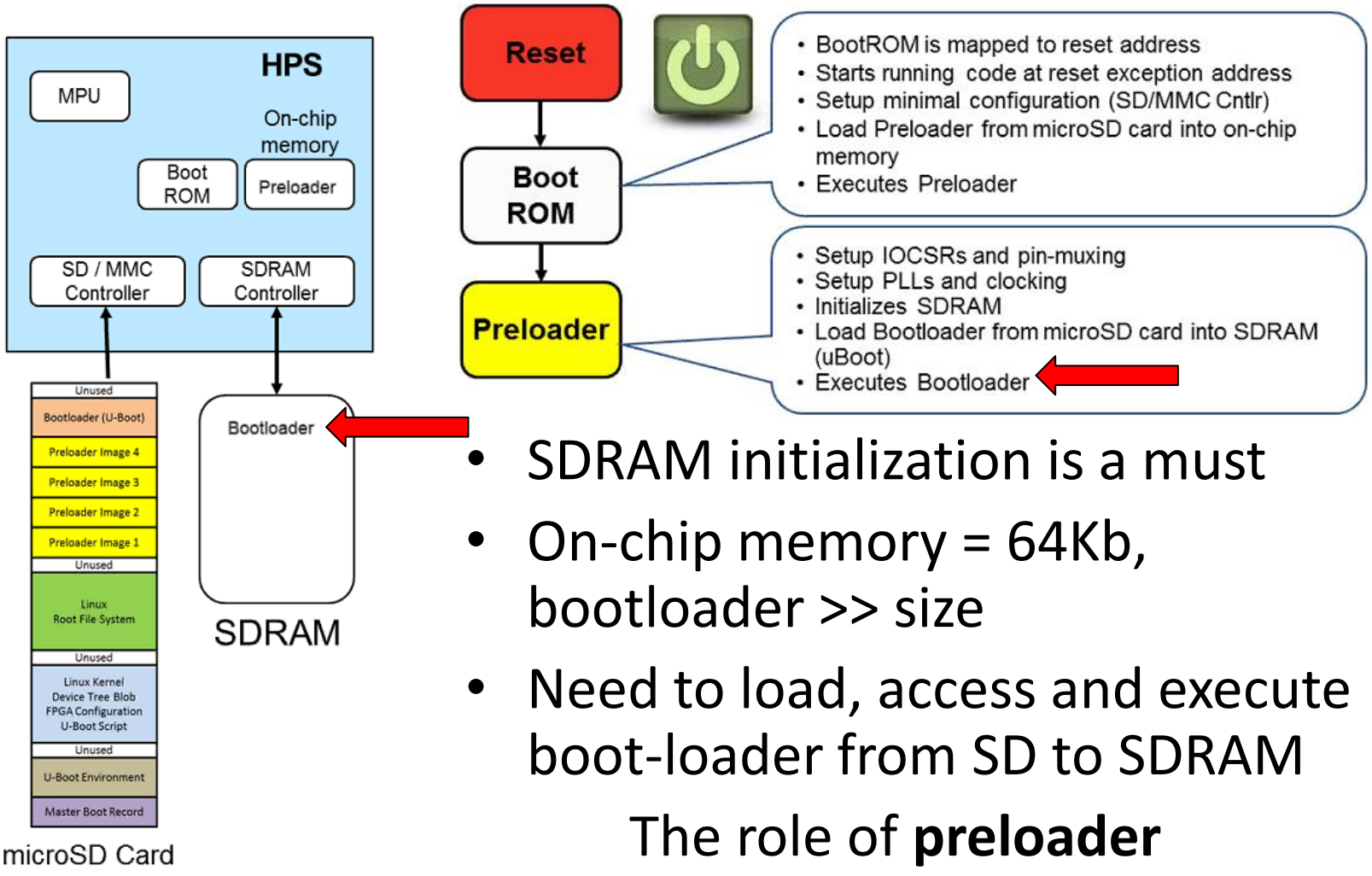
- On-chip mem on Cyclone V = 64Kb
- Once **boot ROM** has finished, turns execution over to **Pre-loader**

Boot up Process



- **Preloader** sets up all:
  - System IOs, PLLs, clocking etc
  - Initializes **SDRAM**

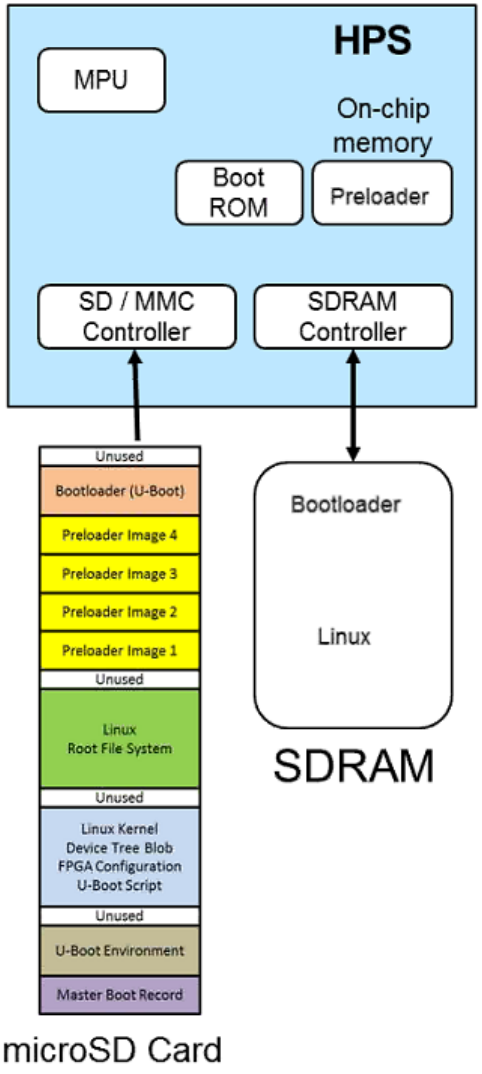
Boot up Process

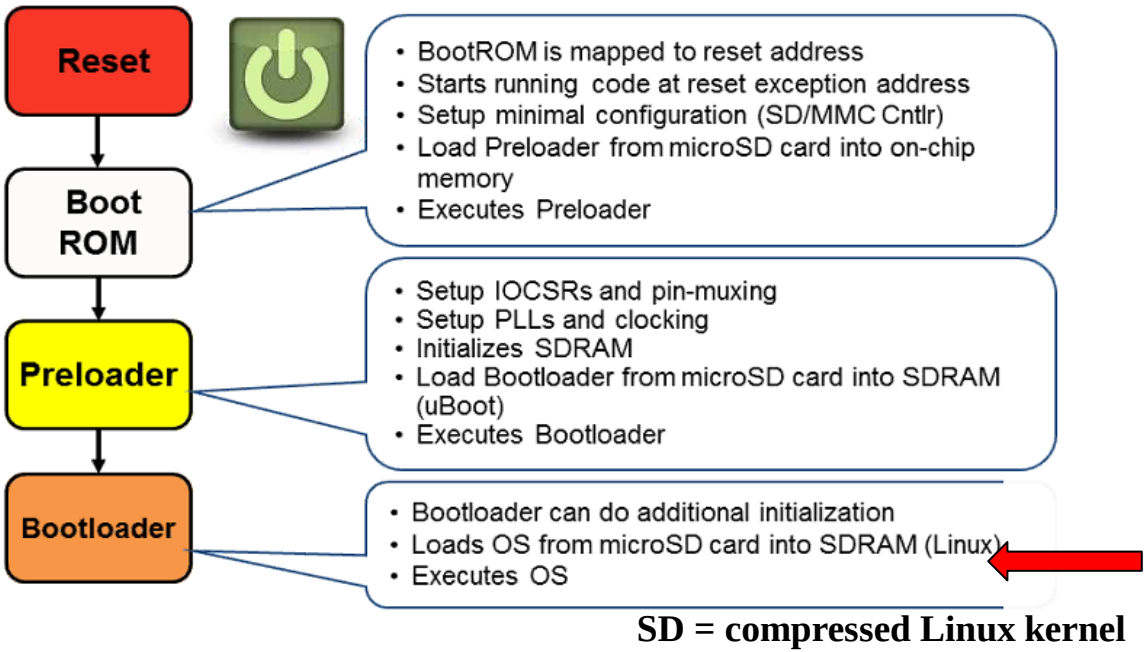


- SDRAM initialization is a must
- On-chip memory = 64Kb, bootloader >> size
- Need to load, access and execute boot-loader from SD to SDRAM

The role of **preloader**

Boot up Process

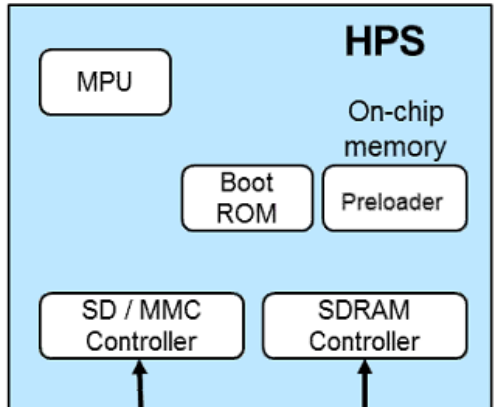


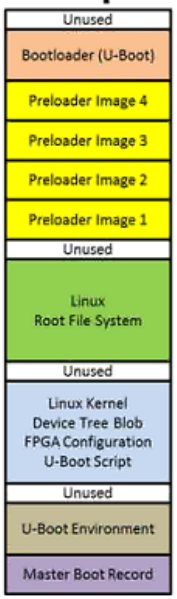


SD = compressed Linux kernel

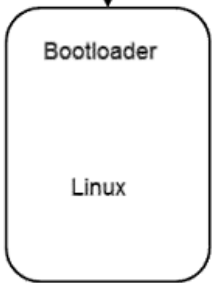
- **Bootloader –** set of instructions used to boot the system's OS kernel, i.e. Loads OS to SDRAM
- We're using **u-boot**

Boot up Process

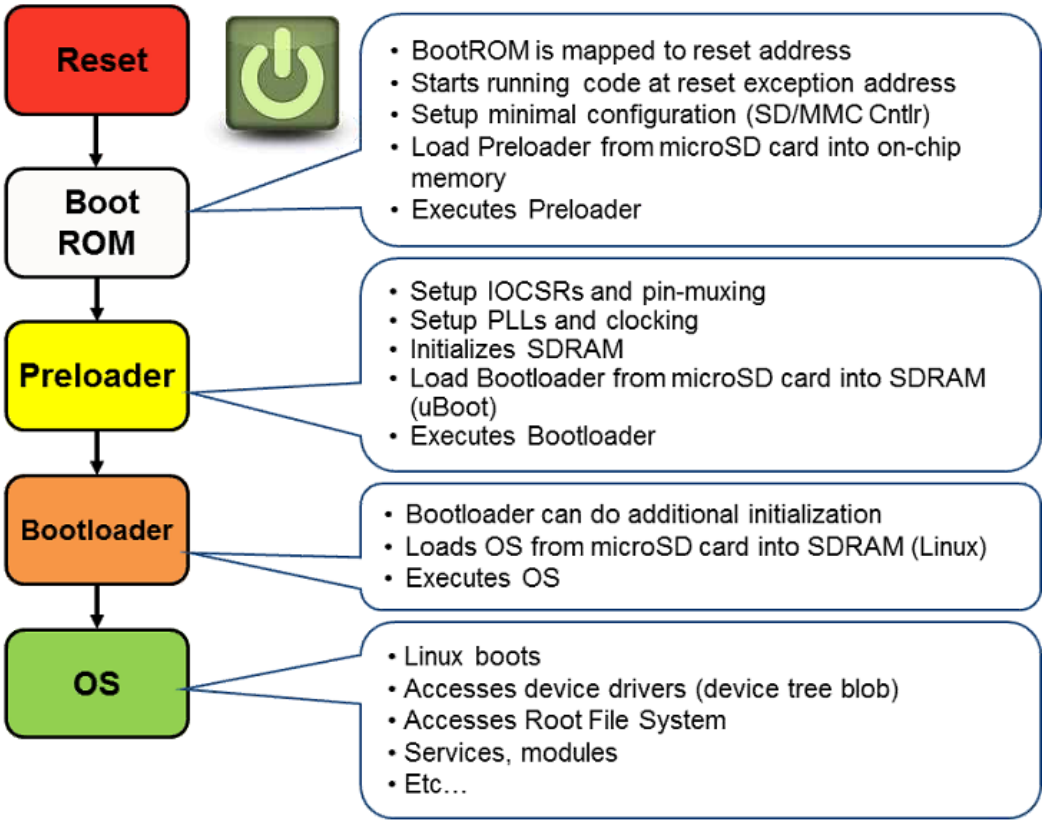




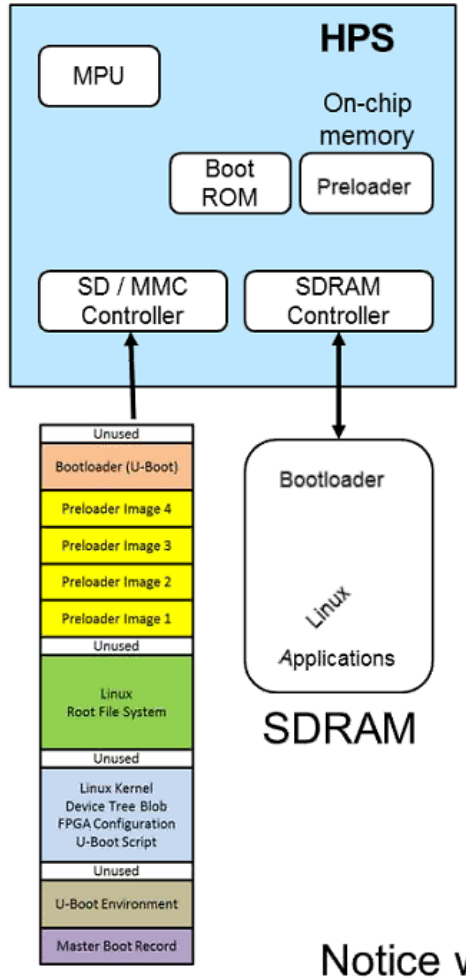
microSD Card

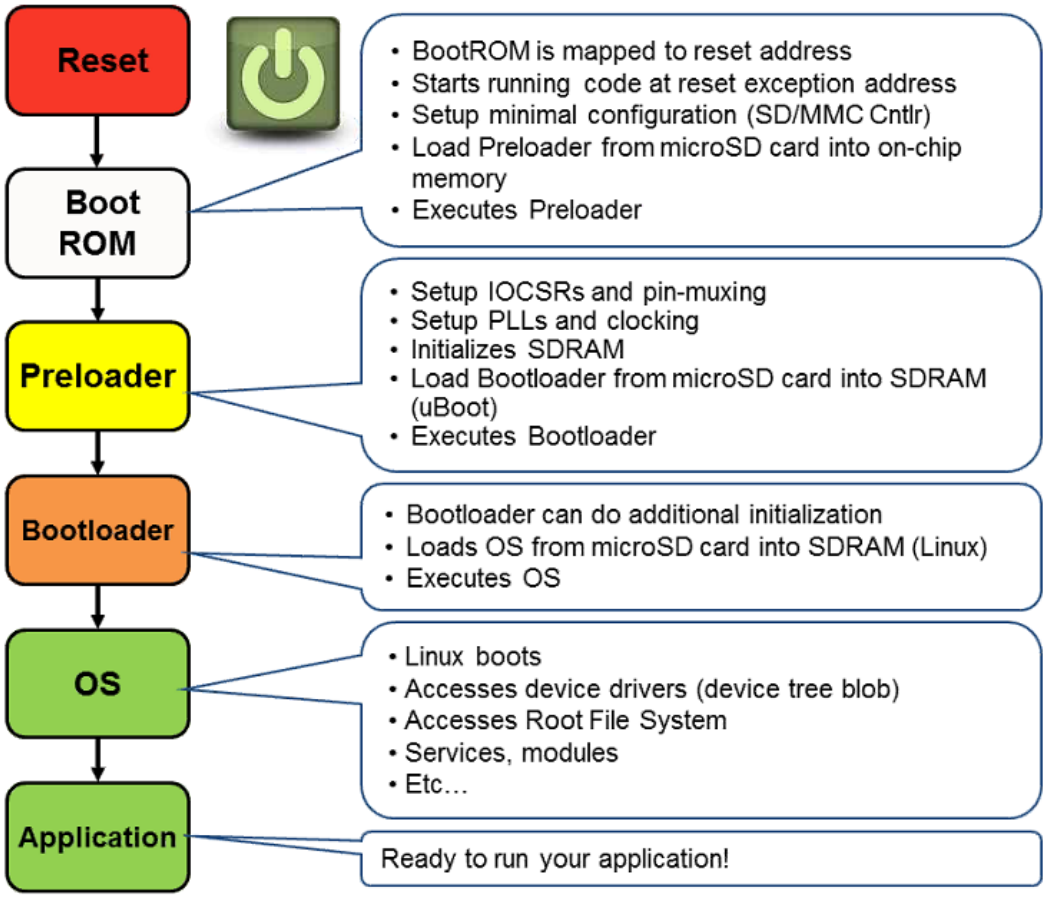


SDRAM



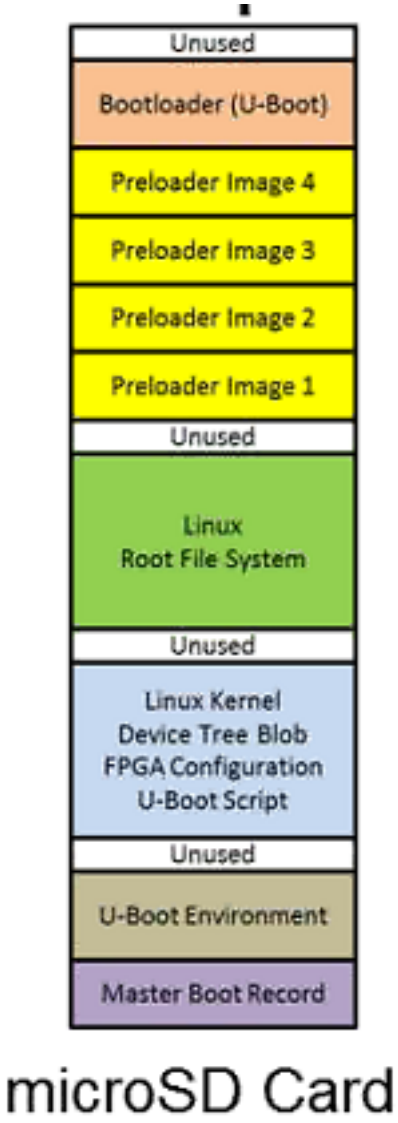
Boot up Process





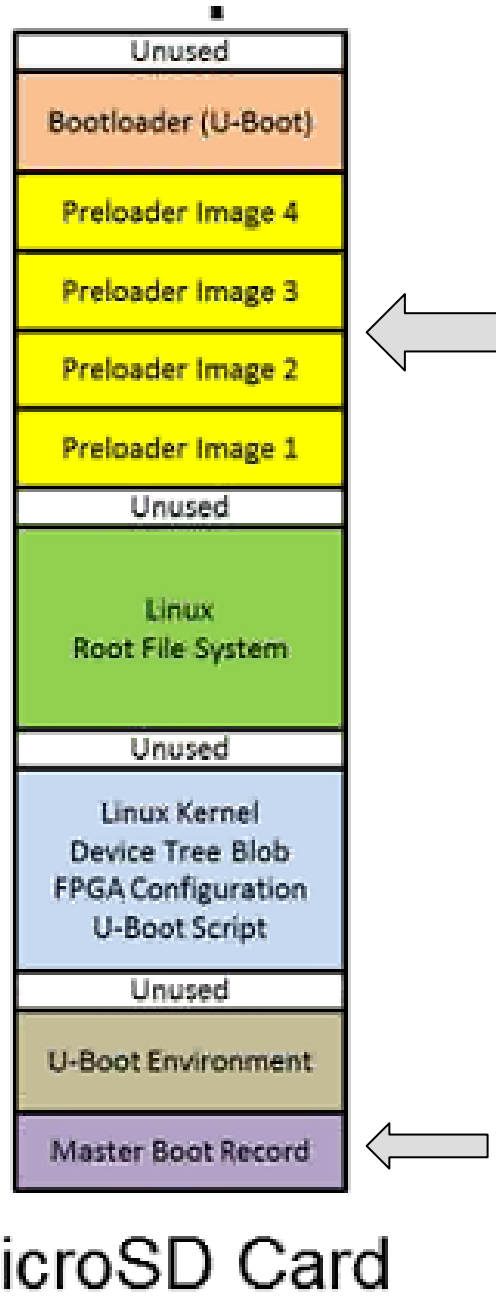
Notice we didn't load the FPGA image!

Boot up Process



- **Look at SD Card layout**
  - **FPGA Configuration**
- **It is possible to get the HPS to configure your FPGA at boot up using the FPGA configuration file**
  - **i.e., the bit-stream on the SD**
- **Use u-boot script to point to necessary locations**

Boot up Process



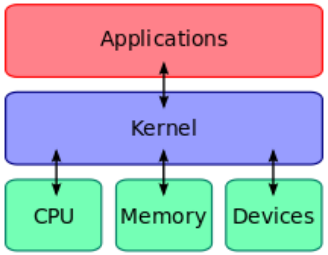
4 replicas of the pre-loader Done since SD cards are known for having bad sectors. We want to guarantee a successful bootup!

Indicates how and where the OS is located on SD, so it can be booted by the system and loaded to its storage

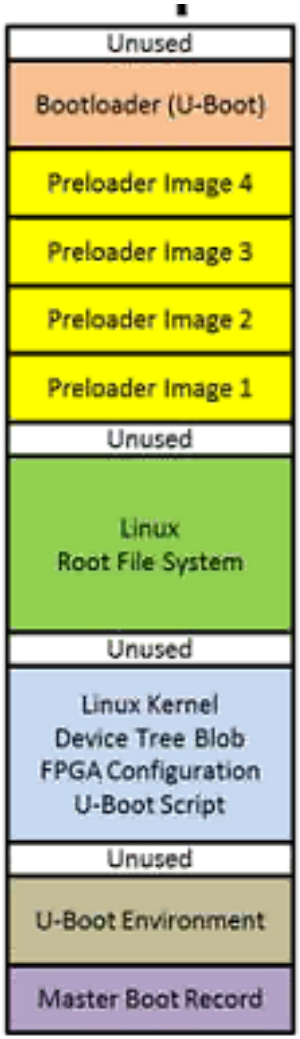
Toolchain

- Set of software development tools “chained” together by specific stages
  - i.e. GCC toolchain
  - Tools include compiler (gcc, g++), libraries (glibc), debugger (dgb) etc
- If we compile from our host computer for HPS:
  - Armv7 or aarch32 toolchain

Boot up Process



Connects application to system HW



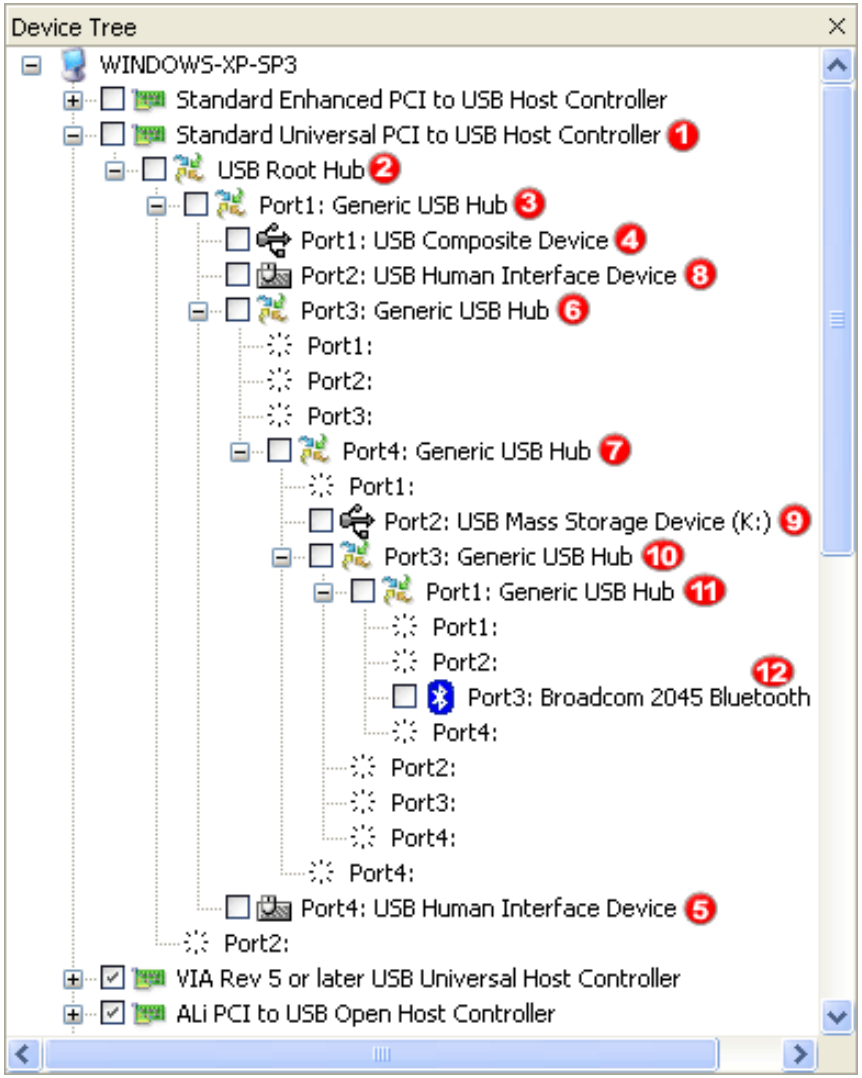
microSD Card

1. **Preloader**
2. **Device Tree**
3. **Bootloader**
4. **Kernel**
5. **Root File System**

Bitbake or use pre-built targets, toolchain selection

Device Tree

- Data structure describing all the hardware devices that the Linux kernel has access to in the system
- At runtime, allows kernel to enable necessary drivers to access the devices

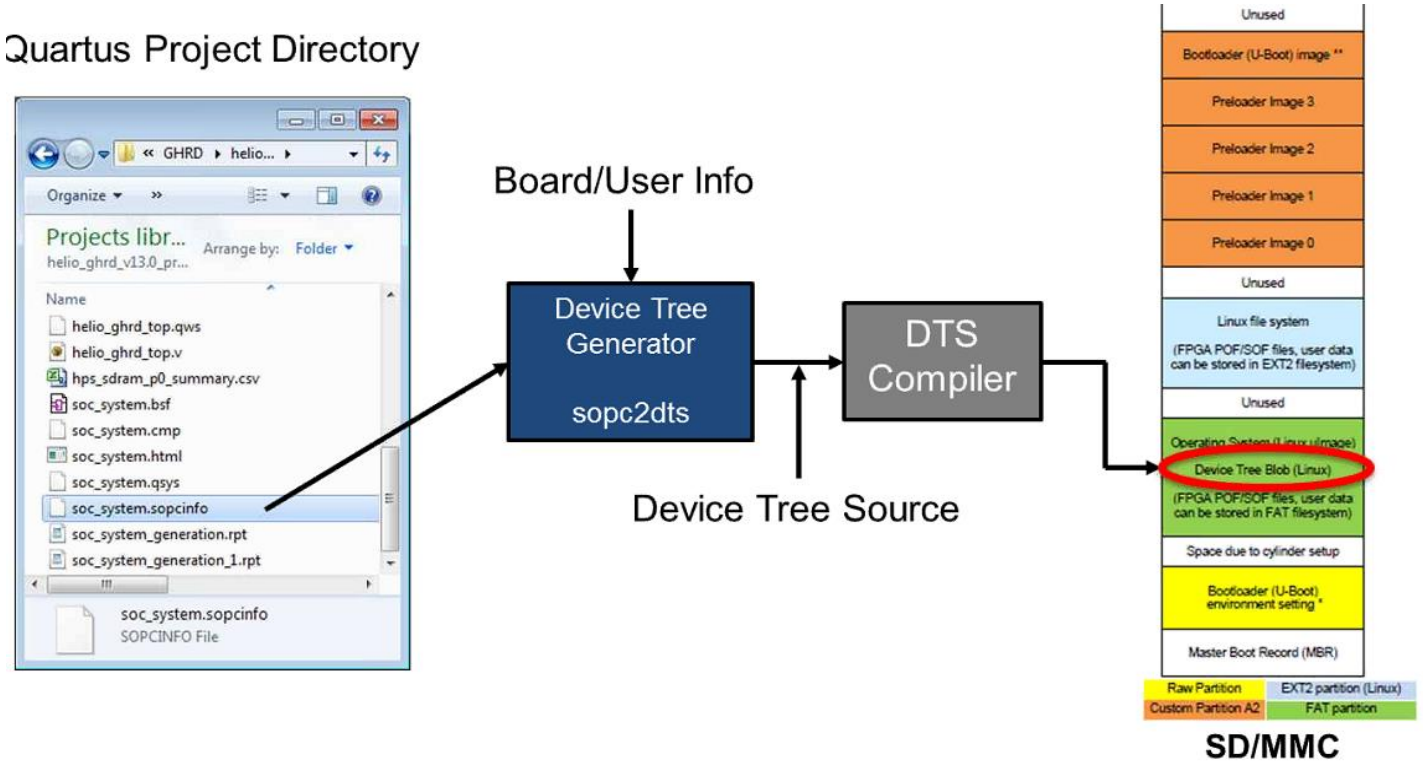


Device Tree

- Allows us to have **one** kernel for a system, don't need to regenerate it every time your HW changes
  - Qsys project – always adding HW, etc.
  - Just recompile new device tree that kernel may read any time a change is made

Device Tree

- Qsys generates .sopcinfo file – that has the info of all devices attached to HPS, used to develop a Device Tree Blob (dtb)



Root File System

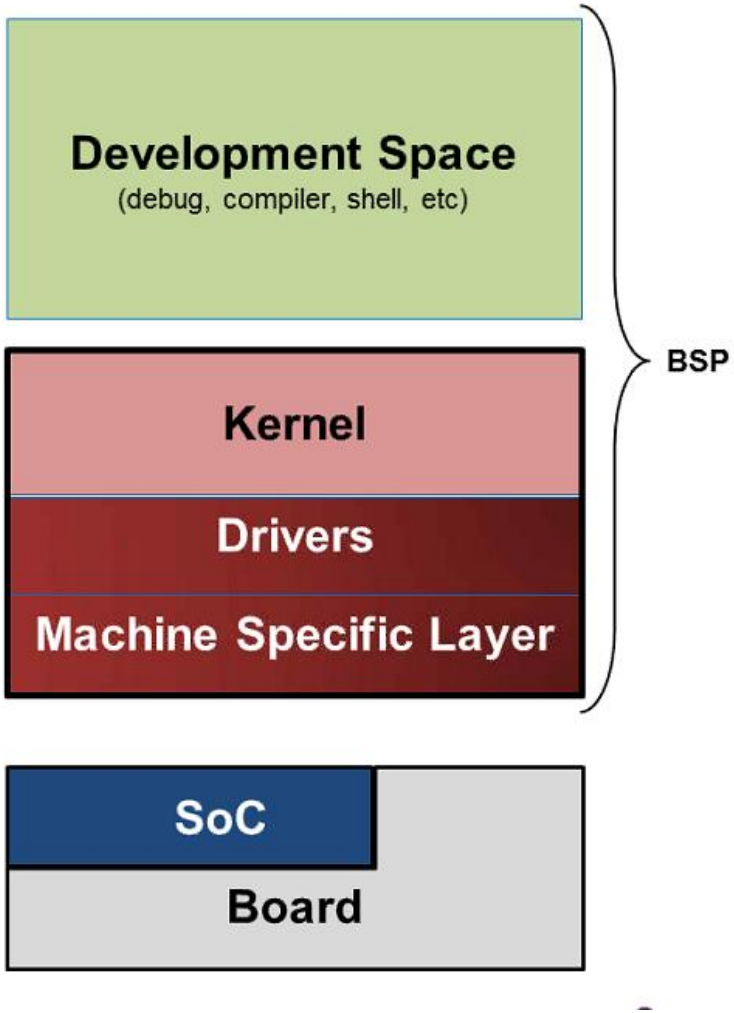
- File system where root directory is located
- File system where all other files systems are mounted
- Built with bitbake with custom parameters (recipes and layers) or use a pre-built RFS

Bitbake Terms (using Yocto)

- **Recipe** – bitbake ingredients and cooking time
  - What version and source codes to use for building (RFS, kernel)
  - Dependencies, targets (“apps”) to be built by bitbake (make, config, ls, etc)
- **Layer** – Set of recipes created by a group for a particular purpose
  - Specifies what core (ARM8) and kernel
  - Altera created meta-altera type for FPGA

Yocto Pre-Built Packages

- Source code package
  - SoC Linux kernel source code
  - SoC U-Boot source code
  - Root file system source code
  - Linaro GCC tool chain
  - The Yocto Project
  - Installation Script
- Binary (pre-built) package
  - kernel
  - Device tree blob
  - Root file system
  - Complete microSD image
- What is not provided?
  - All of the packages known by Yocto (1500+)



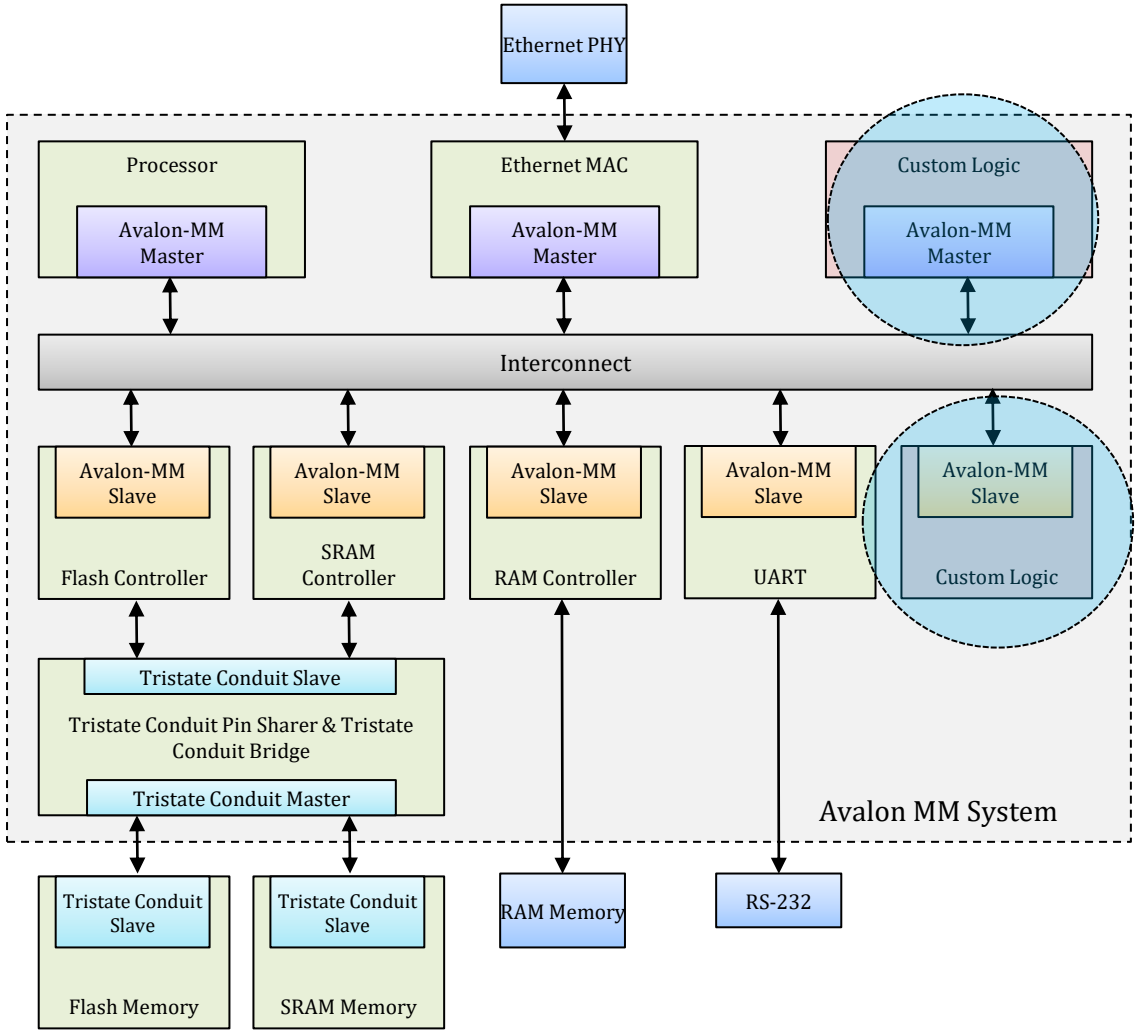
Intel Altera SoC EDS: Embedded Development Suite

SoC EDS =

DS-5 or Higher + Compiler Toolchain +
SOC hw libs + HW/SW interface +
embedded command shell

Custom Logic Bindings

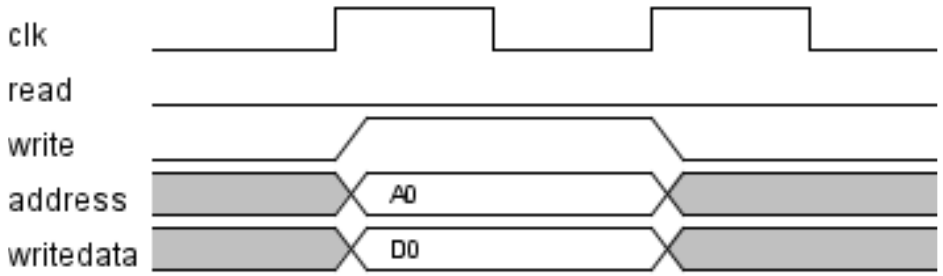
1. Custom Logic can be a master or slave
2. NOIS CPU (Soft IP) is usually the Avalon-MM Master.
3. Any custom I/O can be the Avalon-MM Slave.



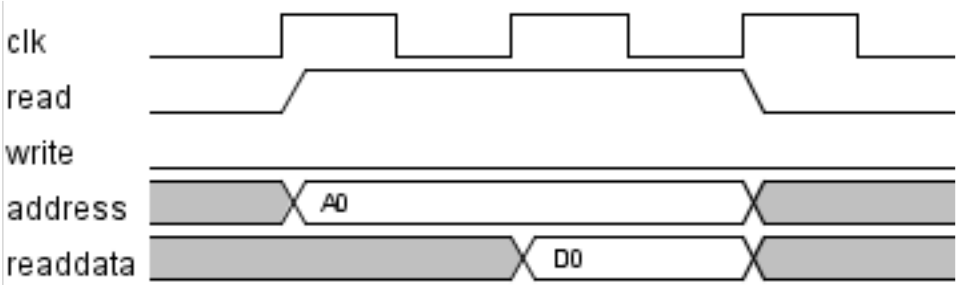
Avalon-MM Slave Interface

Name	Width	Direction	Comments
avs_address	64 bits	Input	Address of slave being accessed
avs_read	1 bit	Input	Read operation requested
avs_write	1 bit	Input	Write operation requested
avs_readdata	8, 16, 32, or 64 bits	Output	Data read from slave
avs_writedata	8, 16, 32, or 64 bits	Input	Data to be written to slave

Write Waveforms:

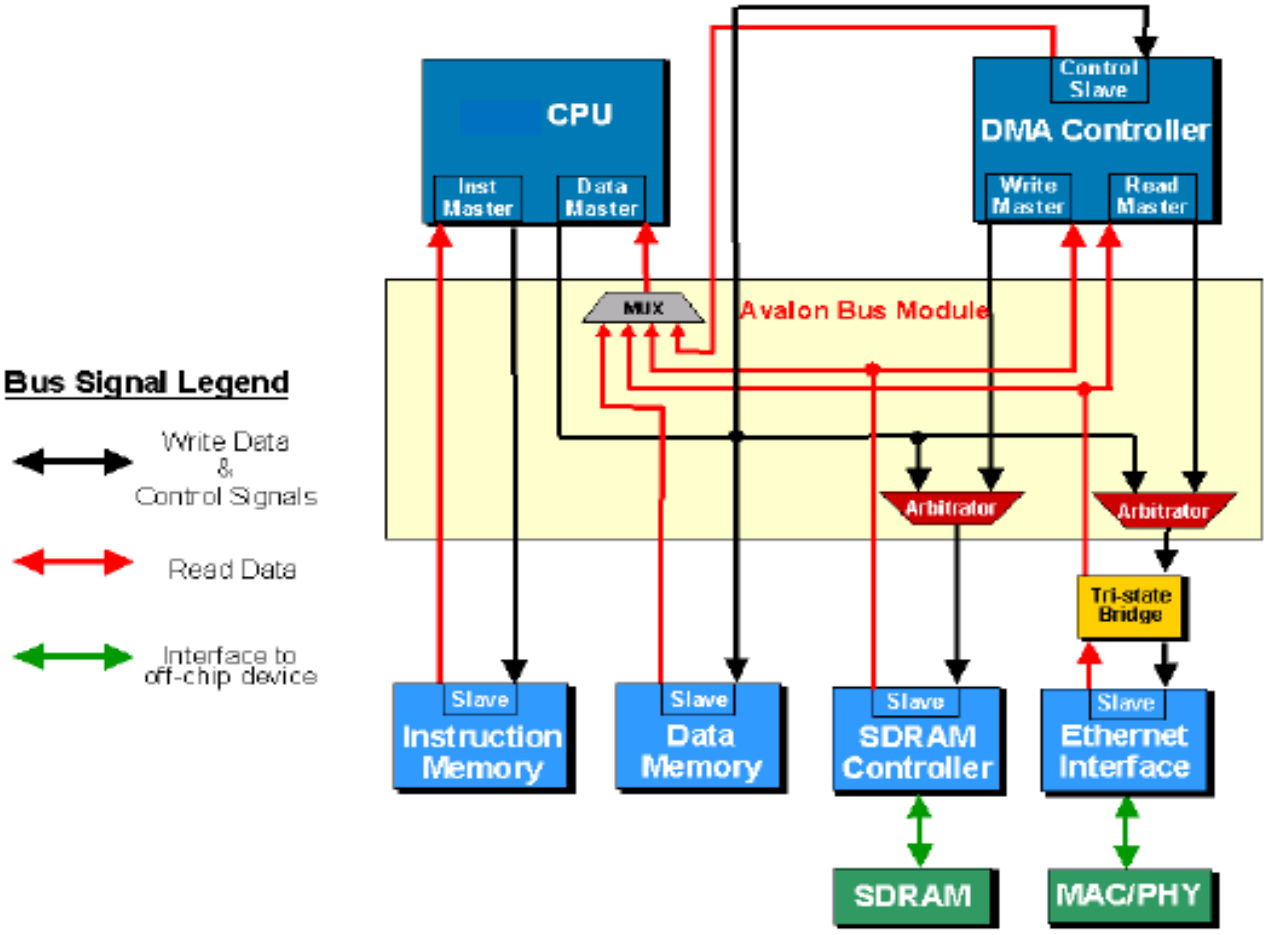


Read Waveforms:



Avalon Bus Module

The Avalon bus module (an Avalon bus) is a unit of active logic, which takes the place of passive, metal bus lines on a physical PCB. Suitable as an on-chip BUS.



Avalon Bus

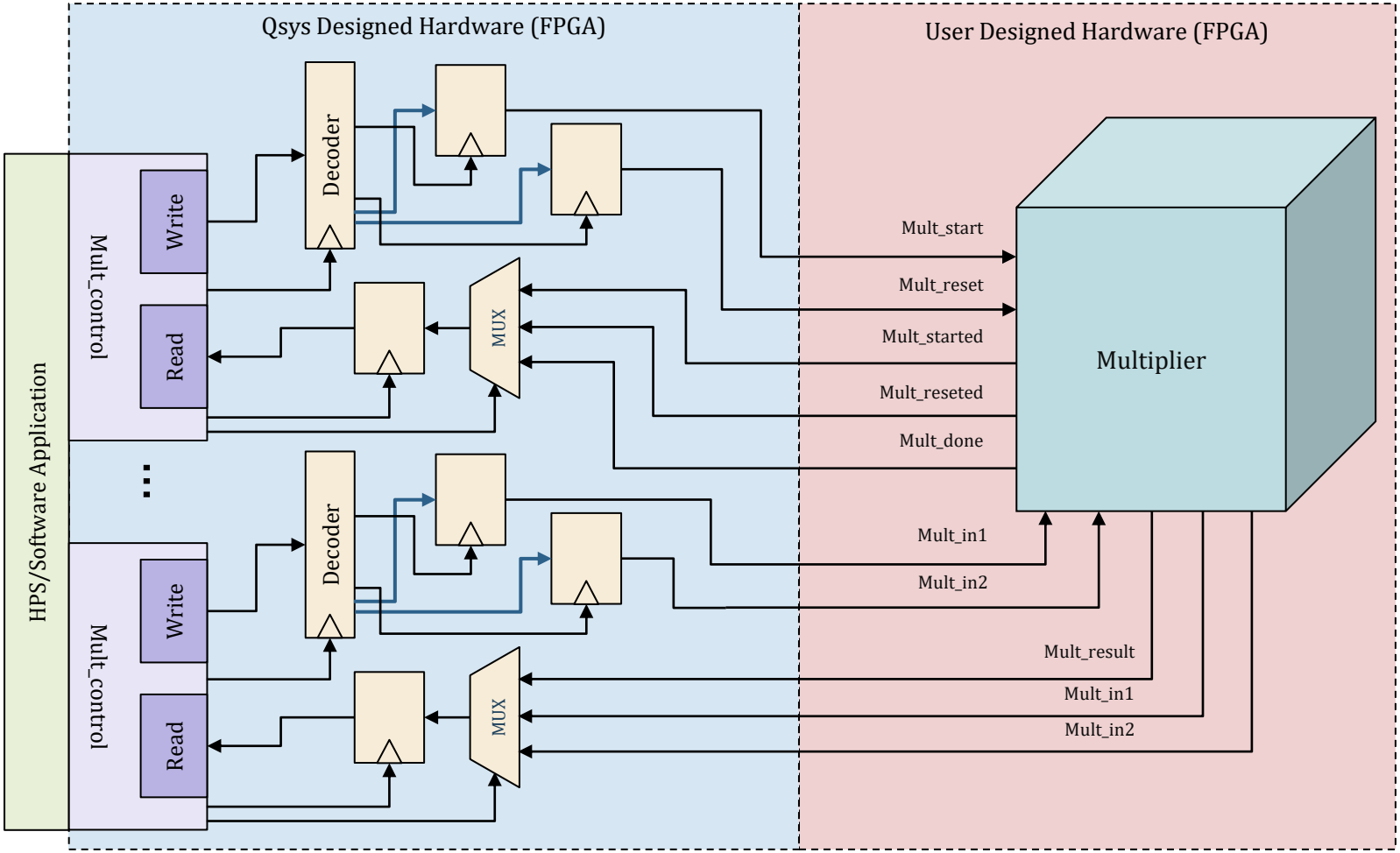
- **Avalon bus is on-chip bus architecture that accommodate the SOC environment.**
- **The interface to peripherals is synchronous with the Avalon clock.**
- **Multiplexers (not tri-state buffers) in the bus determine which signals drive which peripheral. Peripheral is not required to tri-state its output.**
- **Address, data and control signals use separate, dedicated ports. It simplifies the design of peripherals as they don't need to decode address and data bus cycles as well as disable its outputs when it is not selected.**

Slave Arbitrator

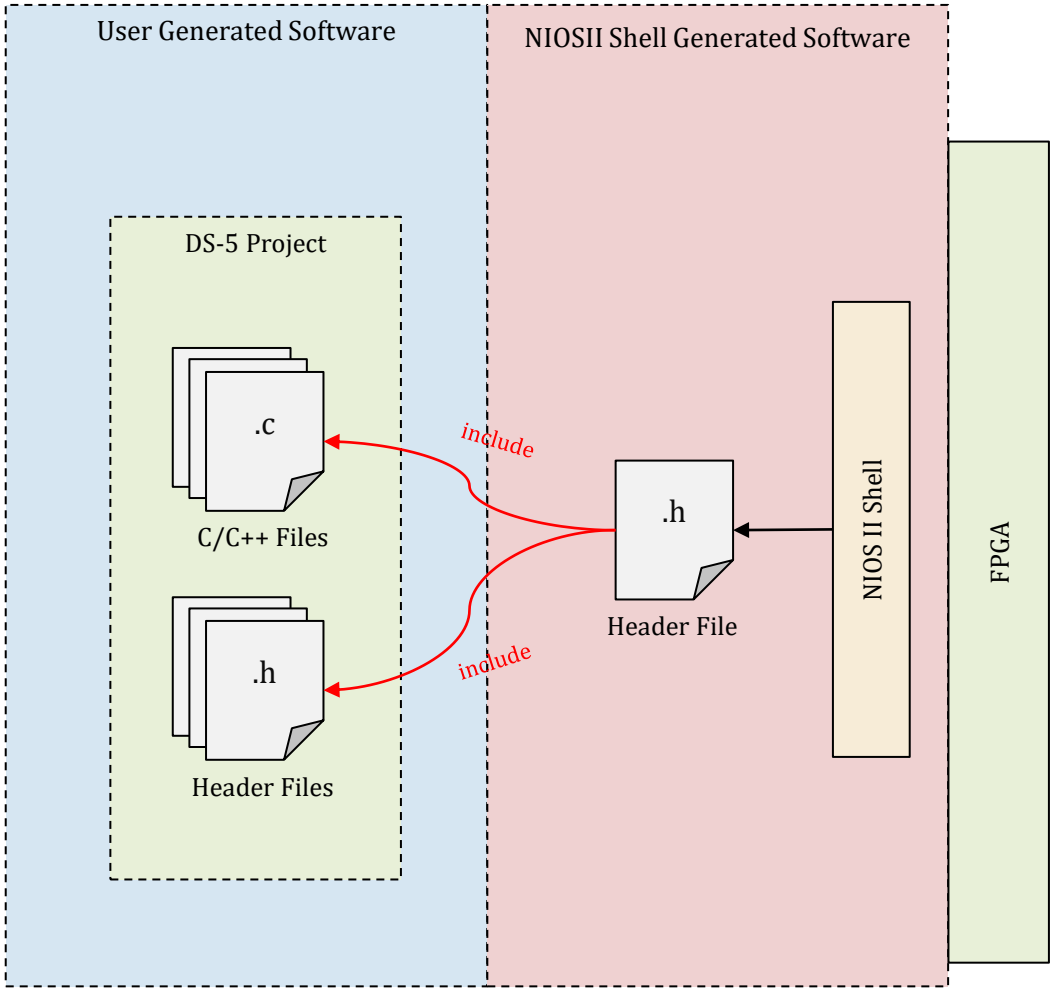
Avalon bus module contains one slave arbitrator for each shared slave port. Slave arbitrator performs the following.

- **Defines control, address, and data paths from multiple master ports to the slave port and specifies the arbitration mechanism to use when multiple masters contend for a slave at the same time.**
- **At any given time, selects which master port has access to the slave port and forces all other contending masters (if any) to wait, based on the arbitration assignments.**
- **Controls the slave port - based on address, data & control signals presented by the currently selected master port.**

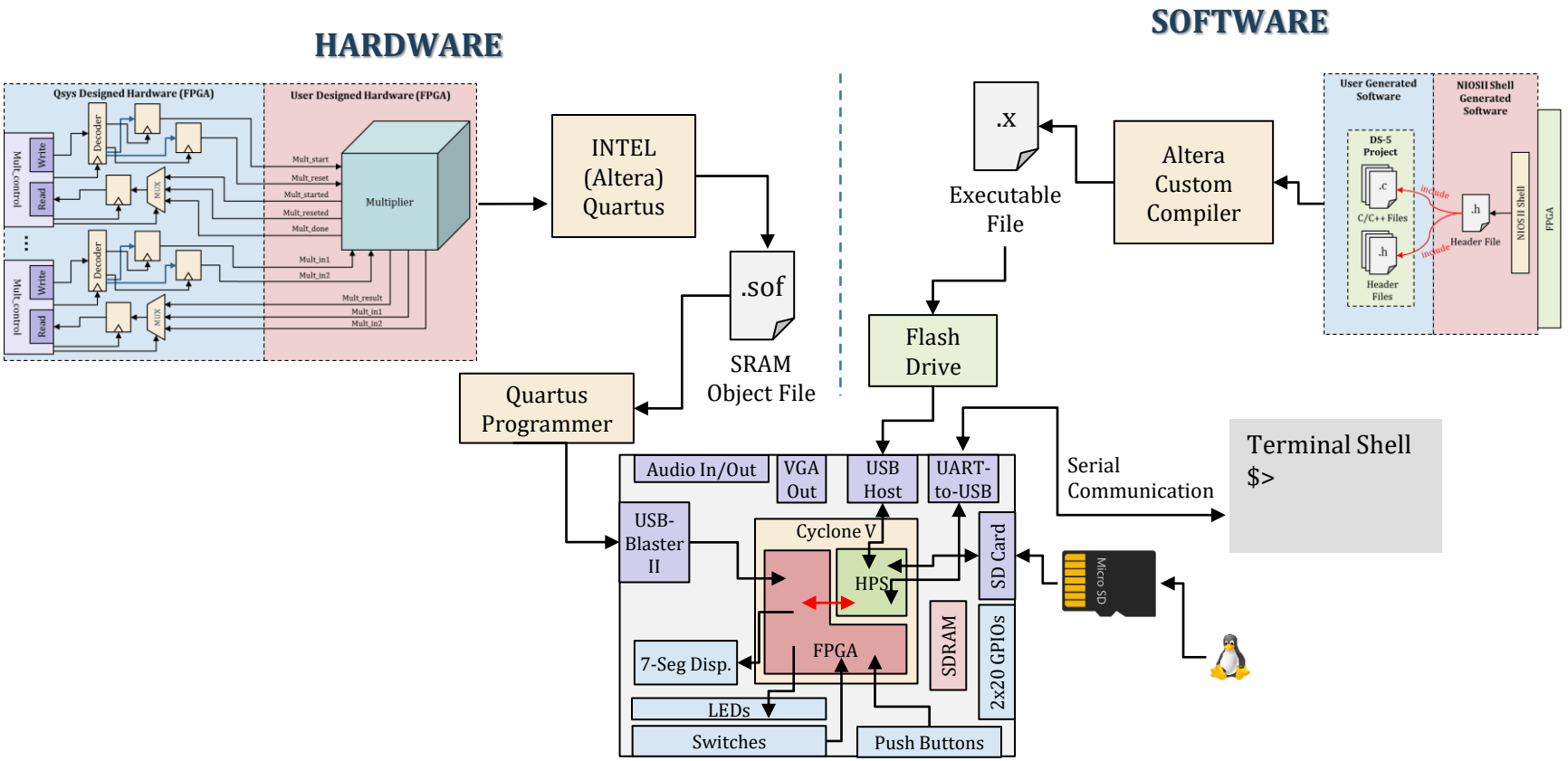
Avalon-MM Slave Interface – Hardware Side



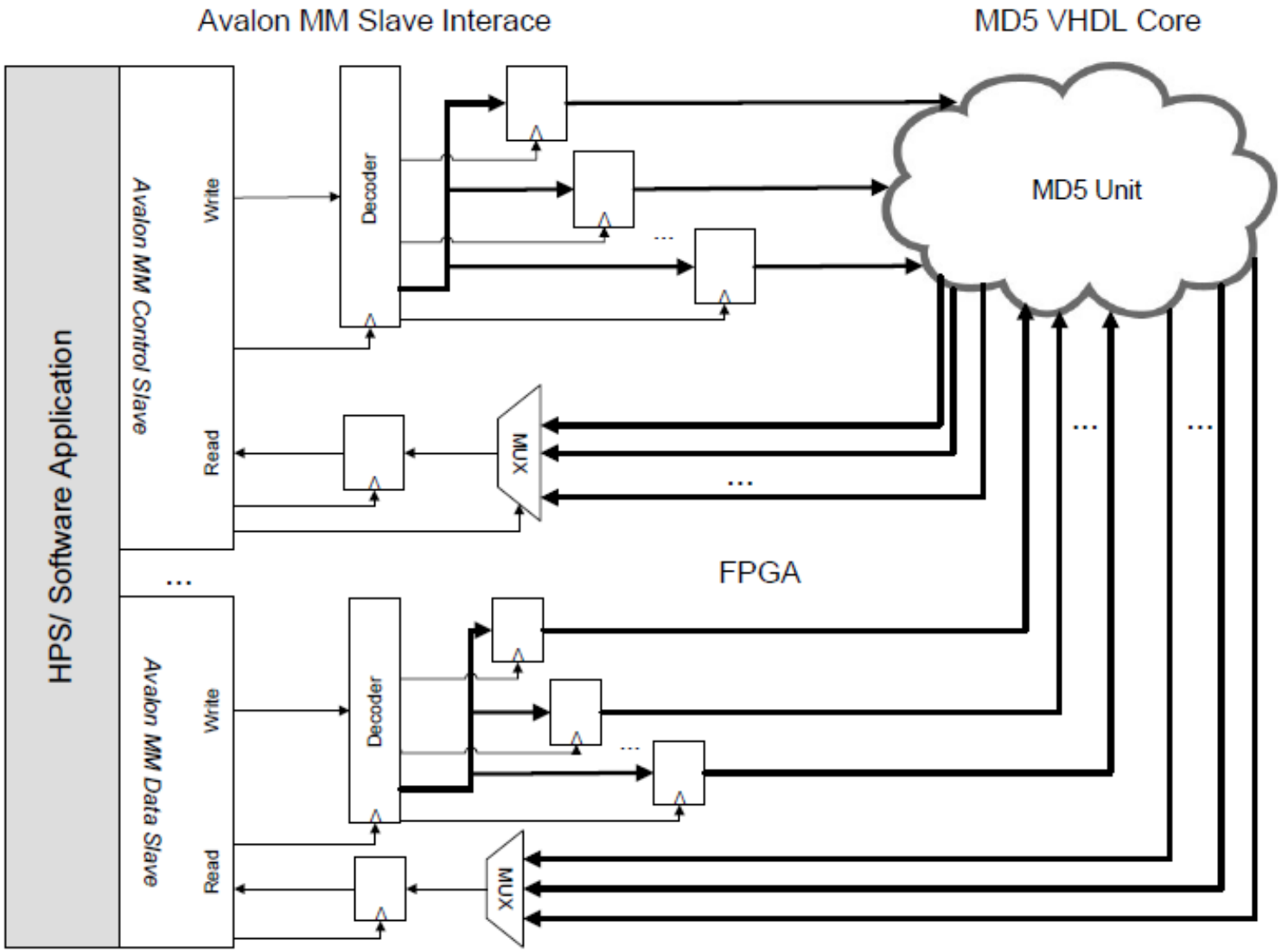
Avalon-MM Slave Interface – Software Side



Putting it all Together

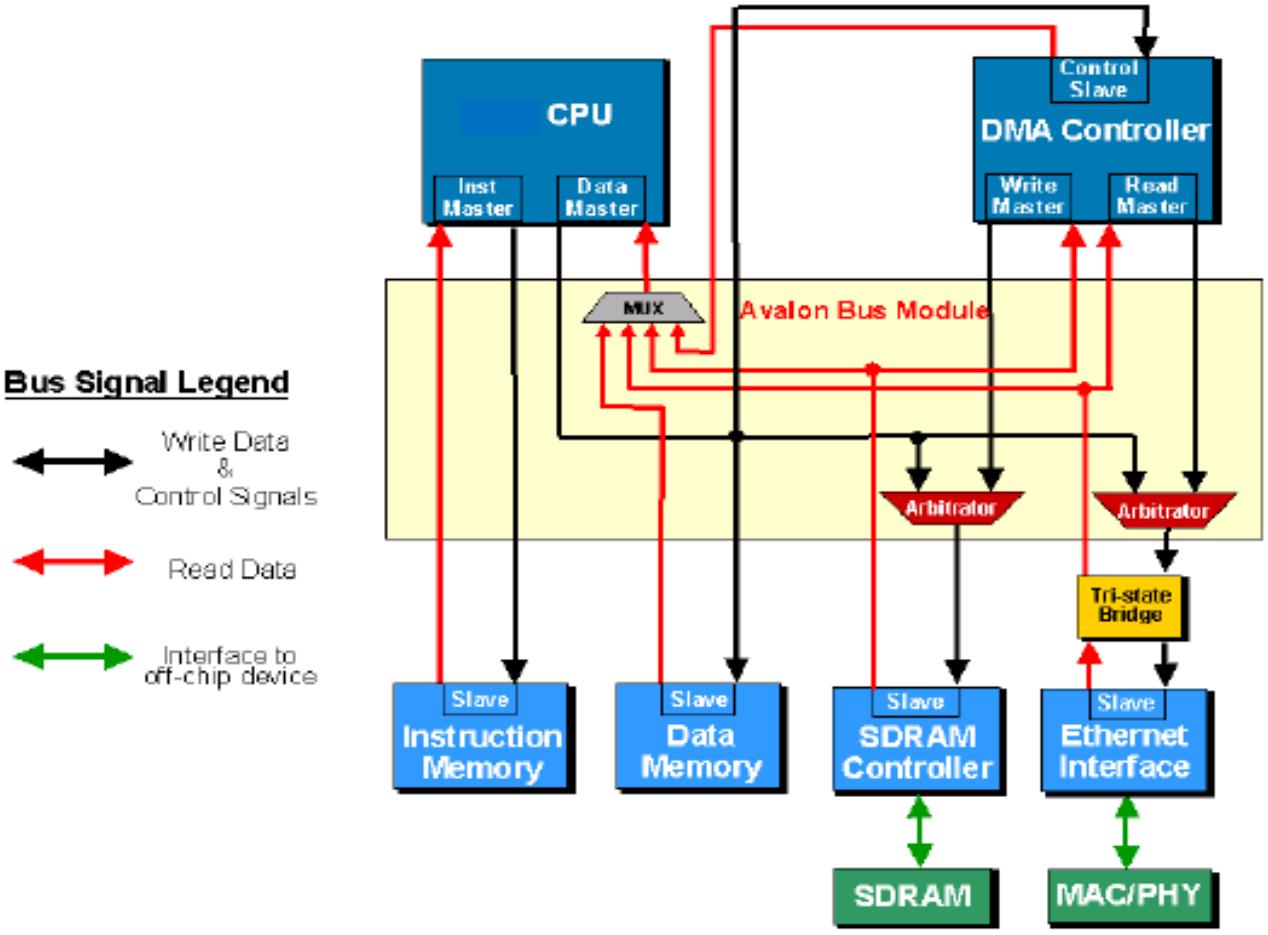


MD5 SoC Project



Avalon Bus Module

The Avalon bus module (an Avalon bus) is a unit of active logic, which takes the place of passive, metal bus lines on a physical PCB. Suitable as an on-chip BUS.



Avalon Bus

- **Avalon bus is on-chip bus architecture that accommodate the SOC environment.**
- **The interface to peripherals is synchronous with the Avalon clock.**
- **Multiplexers (not tri-state buffers) inside the bus determine which signals drive which peripheral. Peripherals are never required to tri-state their outputs.**
- **Address, data and control signals use separate, dedicated ports. It simplifies the design of peripherals as they don't need to decode address and data bus cycles as well as disable its outputs when it is not selected.**

Slave Arbitrator

Avalon bus module contains one slave arbitrator for each shared slave port. Slave arbitrator performs the following.

- **Defines control, address, and data paths from multiple master ports to the slave port and specifies the arbitration mechanism to use when multiple masters contend for a slave at the same time.**
- **At any given time, selects which master port has access to the slave port and forces all other contending masters (if any) to wait, based on the arbitration assignments.**
- **Controls the slave port - based on address, data & control signals presented by the currently selected master port.**