COE838: Systems-on-Chip Design

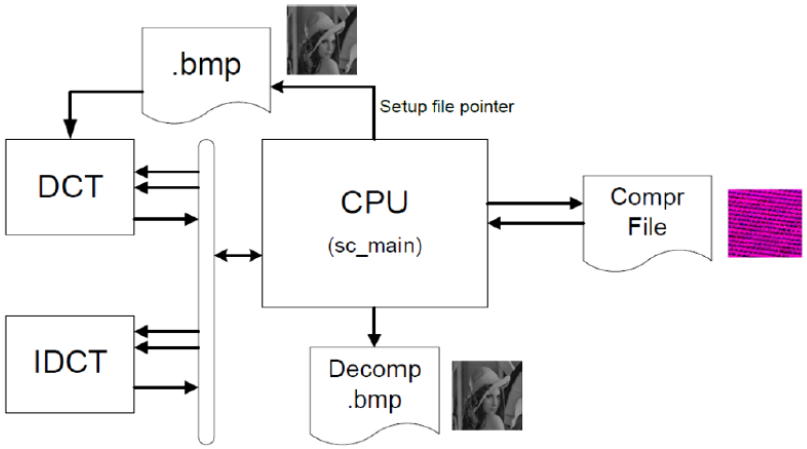
DE1-SoC Labs
Introduction to HPS/FPGA Systems

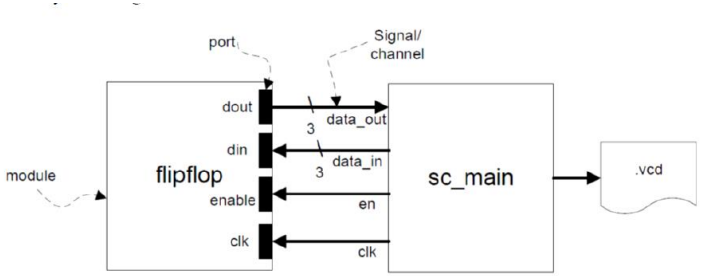
Labs 3, 4 and SoC Project
DE1-SoC Datasheets [online course webpages]

What is covered in Labs 1, 2a & 2b

SystemC CPU/accelerator systems

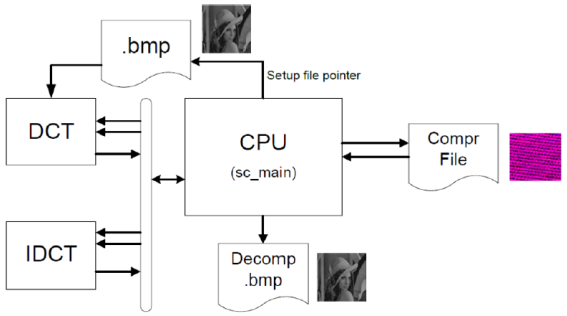
- Accelerators (multiplier, JPEG)
- Binding, offloading, test-benches and software execution on simulated CPU





Initial Labs 1, 2a and 2b

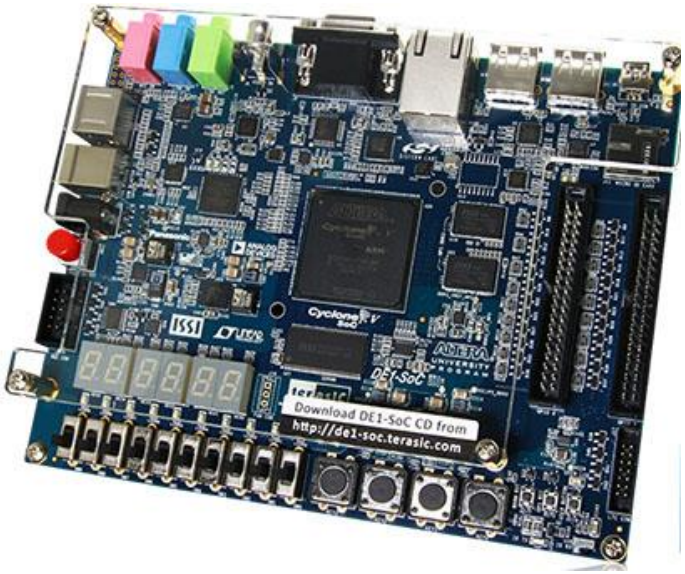
- SystemC CPU/accelerator systems
  - Just modeling the system
  - Can't actually determine some of the system characteristics such as:
    - Frequency
    - Timing errors
    - Chip Area or logic utilization
    - Early estimations of power consumption
    - Etc ...



What we'll be covering

- HPS/FPGA Systems
  - 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

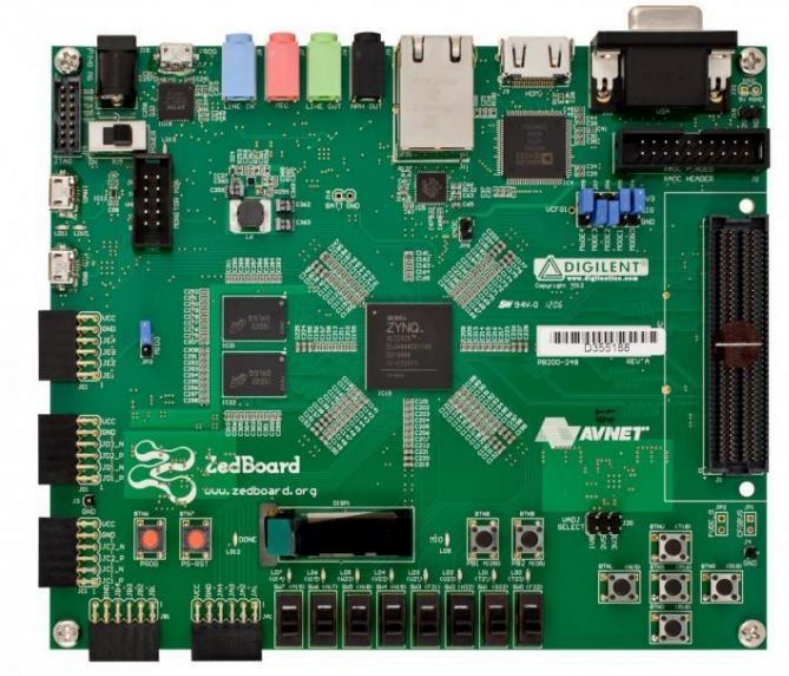
Main HPS/FPGA Systems



Altera DE1-SoC

Cyclone V

Arria V, etc.



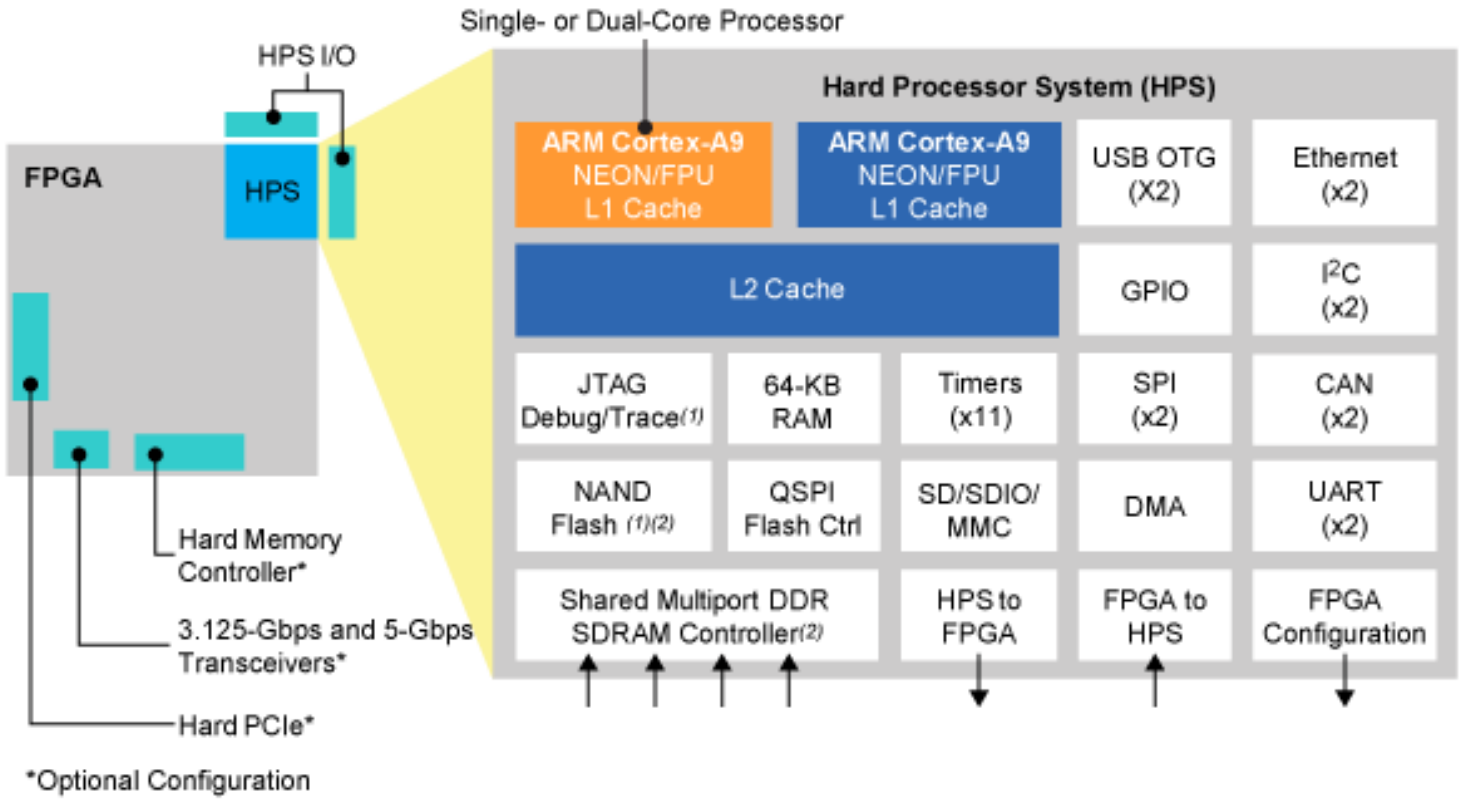
Xilinx Zynq 7000,

Zynq 7XXX, etc

Prototyping SoCs

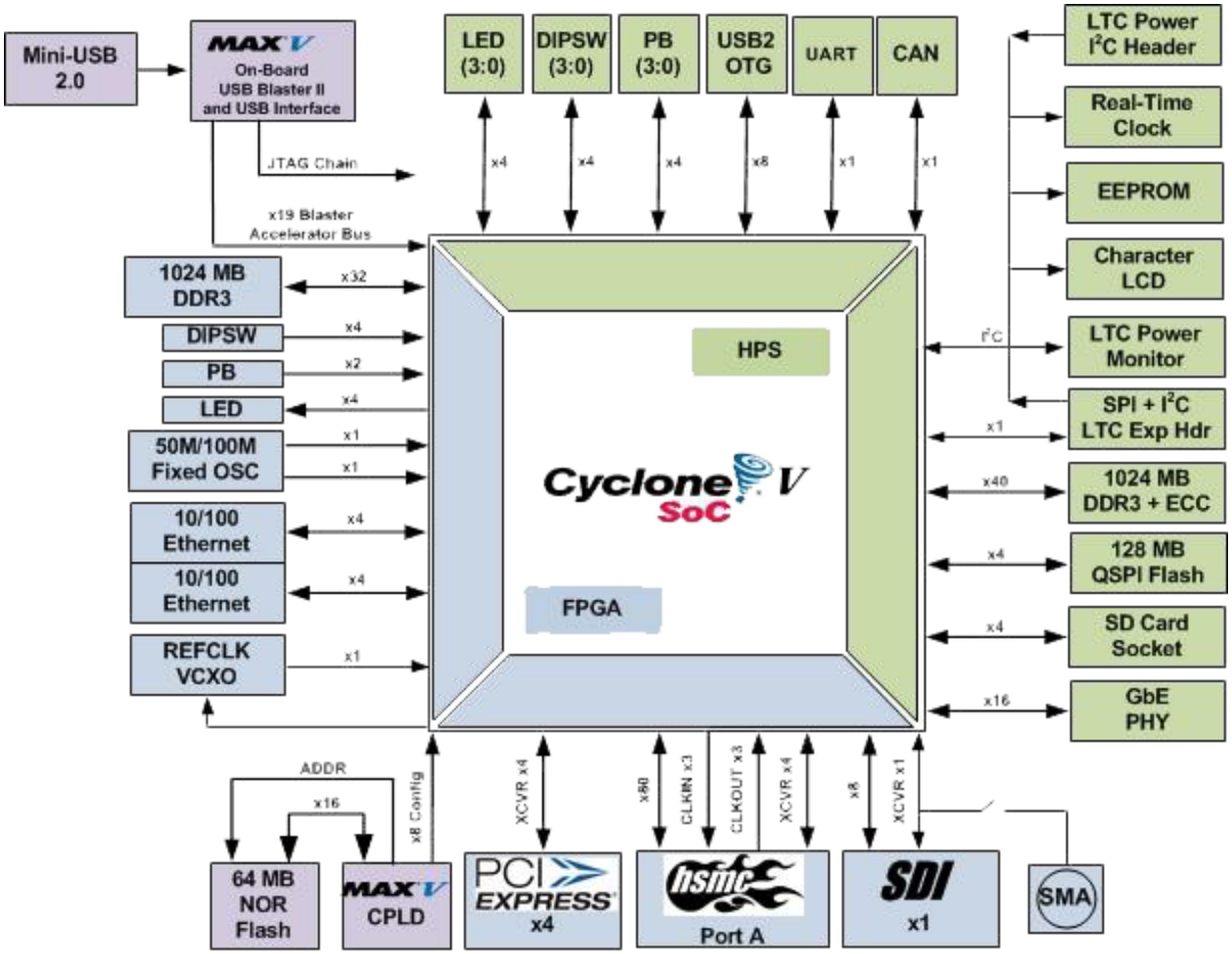
HPS/FPGA SYSTEM ARCHITECTURES

General Overview - FPGA/HPS



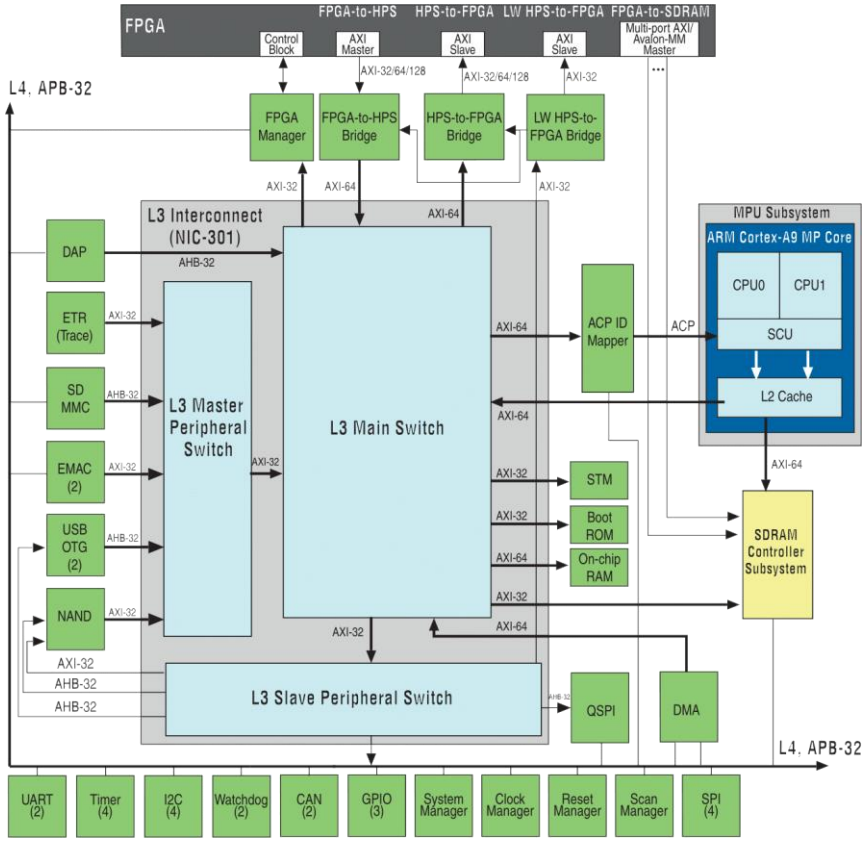
Source: Intel Altera

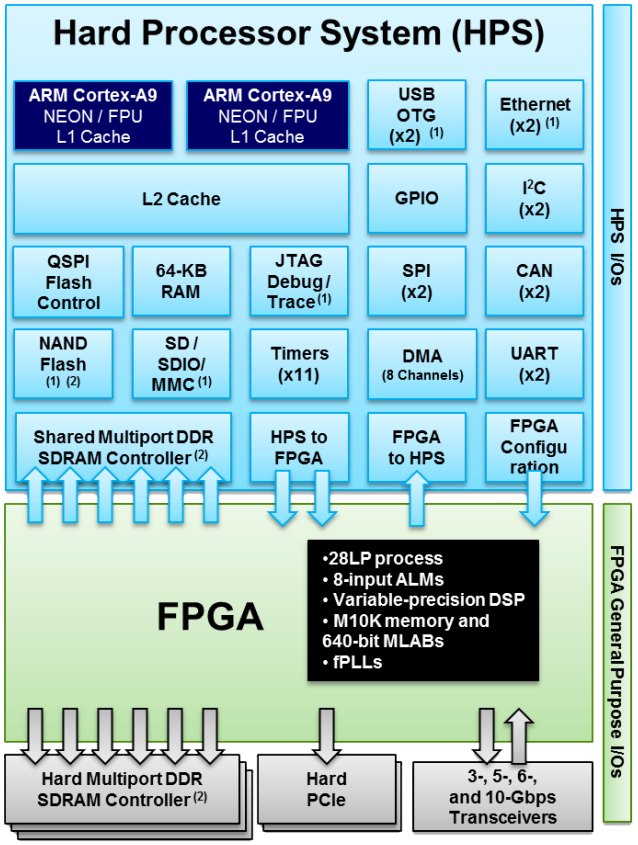
General Overview - FPGA/HPS



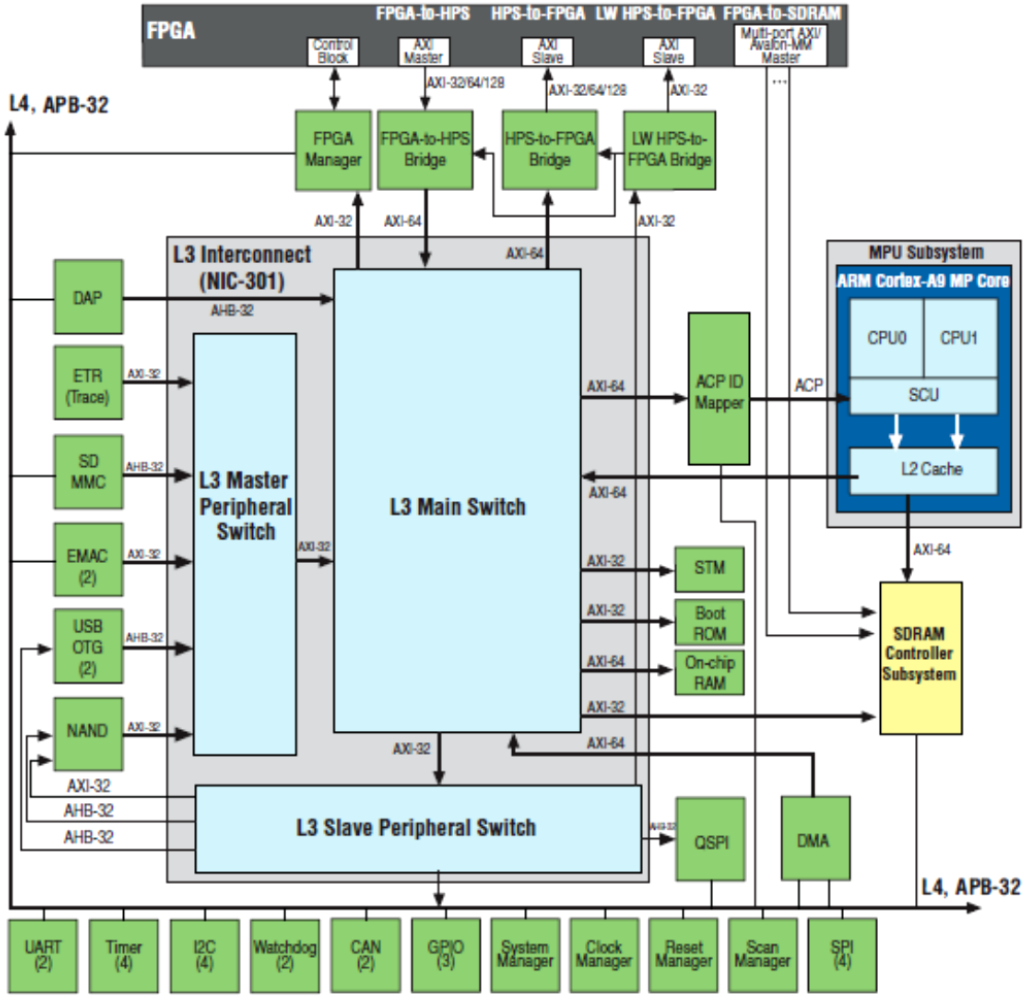
Source: Intel Altera

HPS: ARM Cortex-A9





AXI / HPS-FPGA Interconnects

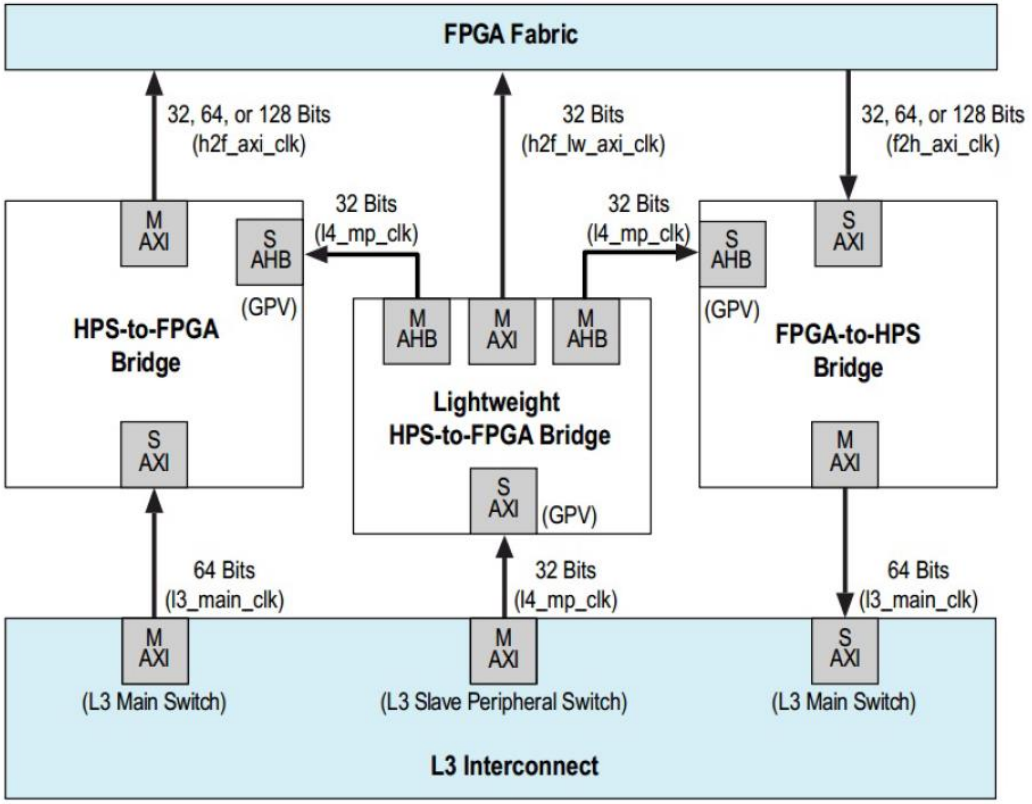


HPS-to-FPGA: Configurable 32, 64 or 128bit AMBA AXI i/f optimized for high bandwidth

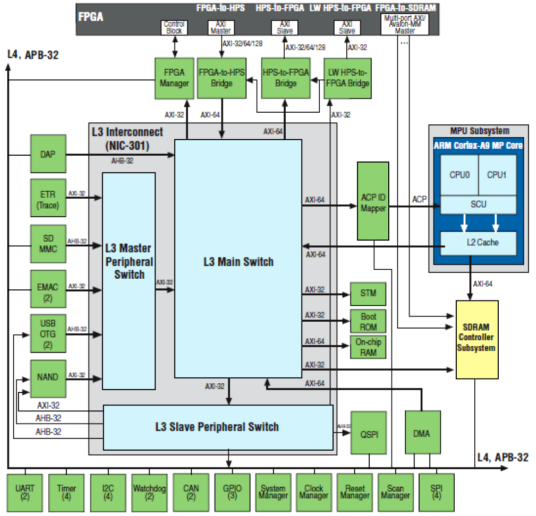
FPGA-to-HPS: --- " ---- Opposite direction (used for ACP mostly)

Lw HPS-to-FPGA: 32-bit AMBA AXI i/f optimized for low latency

HPS – FPGA Bridges

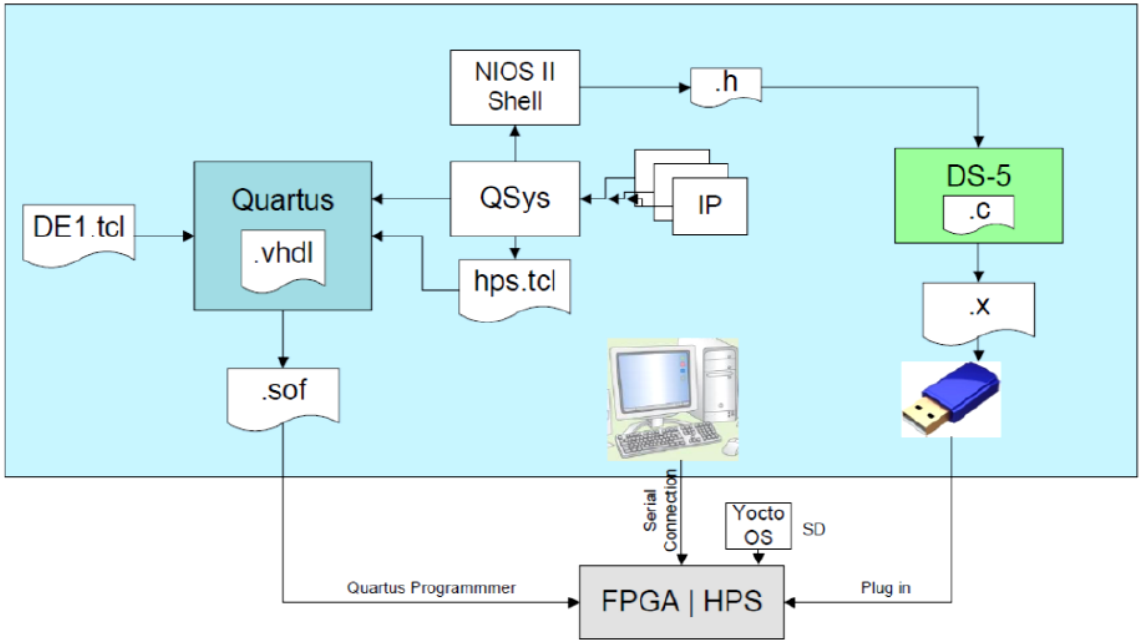


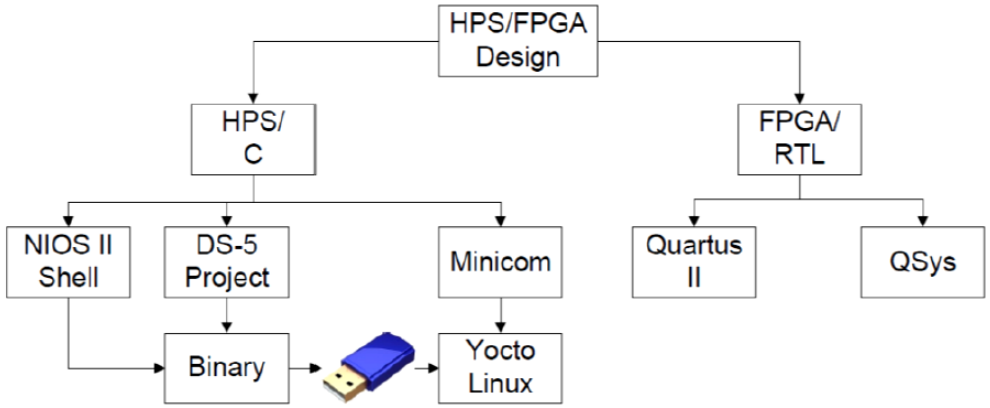
-Modifies data and clock signals to support transportation (protocols, clocking etc) between components



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

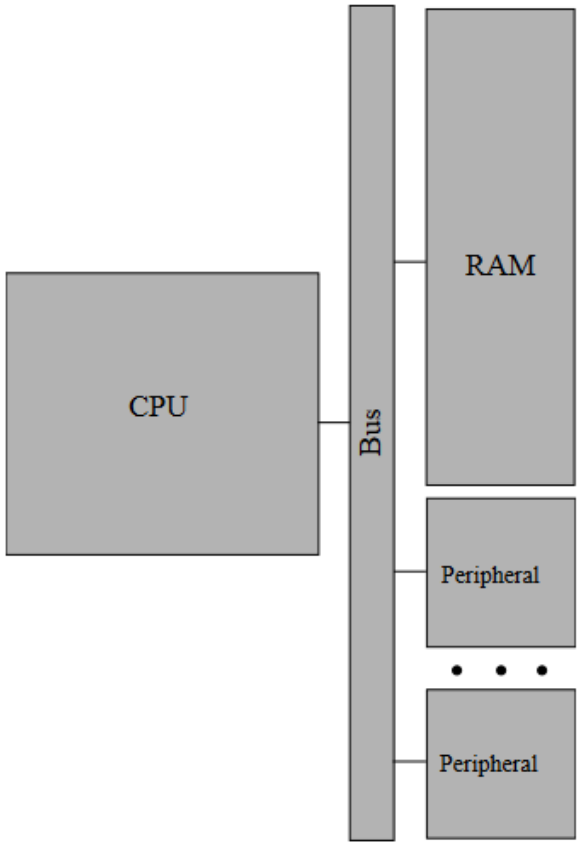
DE1-SoC Tutorial





Memory-Mapped IO

- Used so software can control hardware peripherals
- Processor sees "memory", but it's actually a bus/interconnect with connected peripherals/RAM
- "Memory" has a base address
- Access a component with an "offset" to the base address

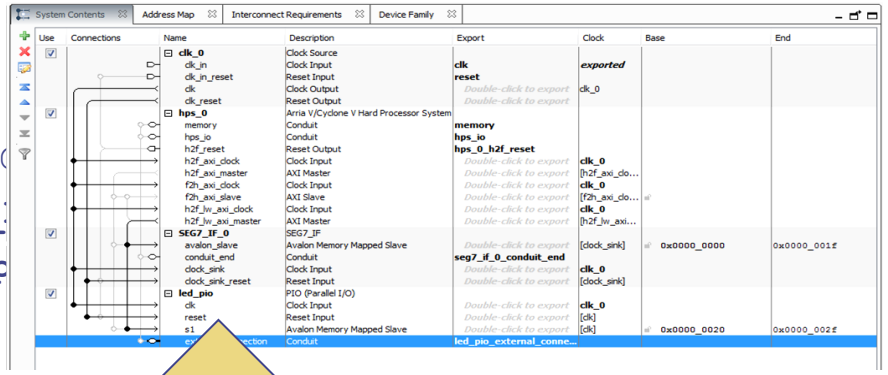


Memory-Mapped IO

```
void *virtual_base;
#define LWHPS2FPGA_BASE 0xff200000
volatile uint32_t *h2p_lw_led_addr = NULL;
....
//open the /dev/mem to access the FPGA space for reading and writing
   if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
        printf( "ERROR: could not open \"/dev/mem\"...\n" );
        return( 1 );
   }

   //map the virtual memory space to virtual_base, that is 2MB in size
   //(0x00200000), at address LWHPS2FPGA_BASE
   virtual_base =  mmap( NULL, LW_SIZE, ( PROT_READ | PROT_WRITE ),
   MAP_SHARED, fd, LWHPS2FPGA_BASE);

   // map the address space for the LED and HEX registers into user space so
   //we can interact with them.. virtual_base + the offset of your IP
   component
   h2p_lw_led_addr= virtual_base + ((uint32_t)(LED_PIO_BASE));
```

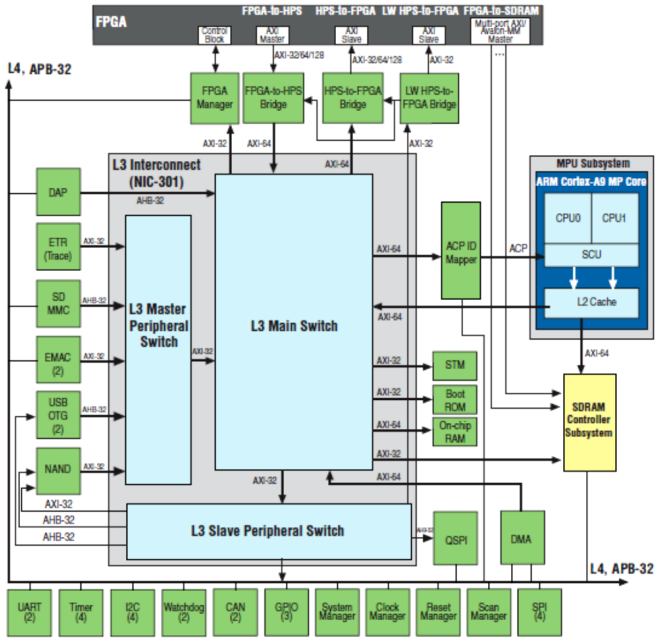
Memory-Mapped IO

```
void *virtual_base;
#define LWHPS2FPGA_BASE 0xff200000
volatile uint32_t *h2p_lw_led_addr =
....
//open the /dev/mem to access the FP
    if( ( fd = open( "/dev/mem", ( O_
          printf( "ERROR: could not op
          return( 1 );
    }

    //map the virtual memory space to virtual   se, that is 2MB in size
    //(0x00200000), at address LWHPS2FPGA_BAS
    virtual_base =  mmap( NULL, LW_SIZE, ( PR   READ | PROT_WRITE ),
    MAP_SHARED, fd, LWHPS2FPGA_BASE);

    // map the address space for the LED and   registers into user space so
    //we can interact with them.. virtual_bas   the offset of your IP
    component
    h2p_lw_led_addr= virtual_base + ((uint32_t)(LED_PIO_BASE));
```

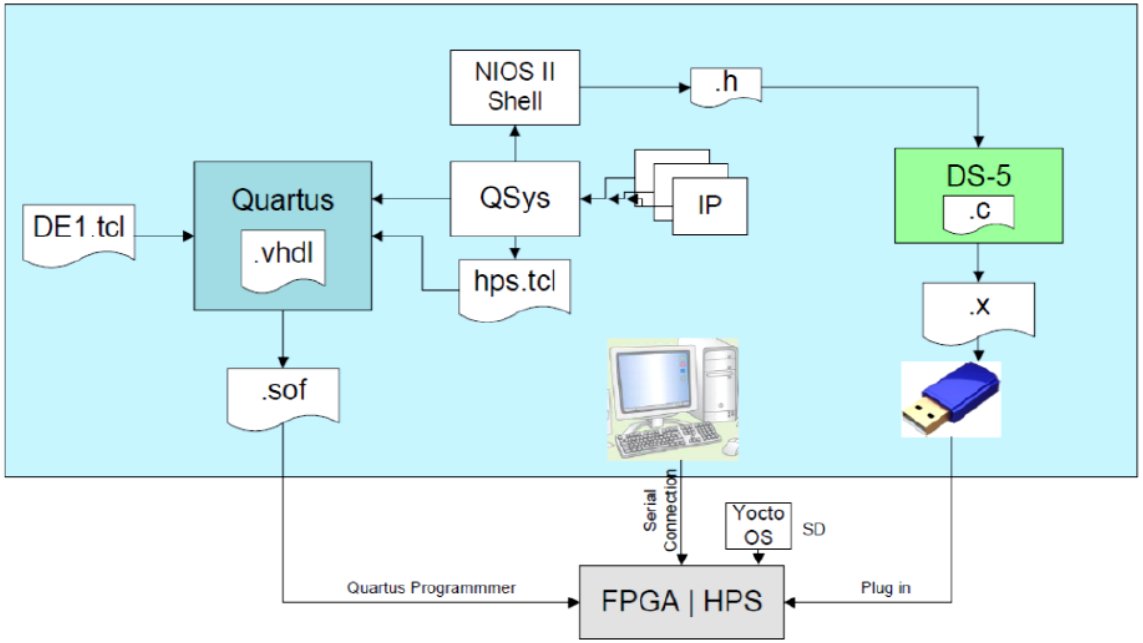


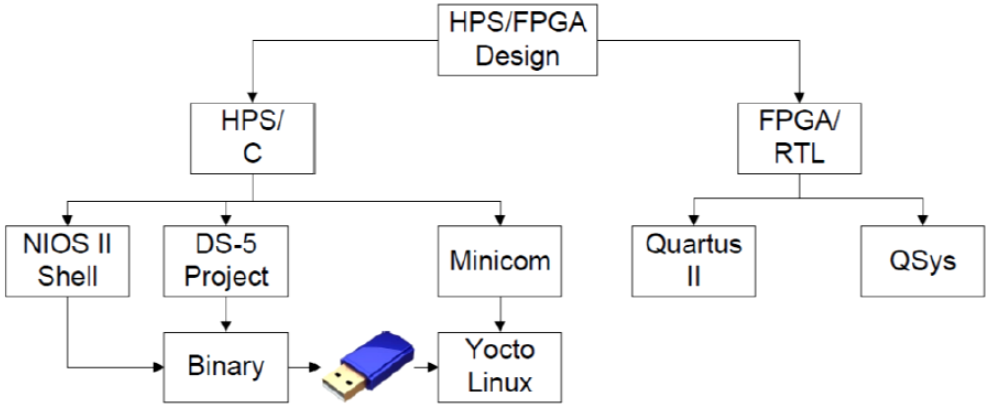
Memory-Mapped IO

```
volatile uint32_t *h2p_lw_led_addr = NULL;
....
   // map the address space for the LED and HEX registers into user space so
   //we can interact with them.. virtual_base + the offset of your IP
   component
   h2p_lw_led_addr= virtual_base + ((uint32_t)(LED_PIO_BASE));


.....

   alt_write_word(h2p_lw_led_addr, 0x3FF);
```



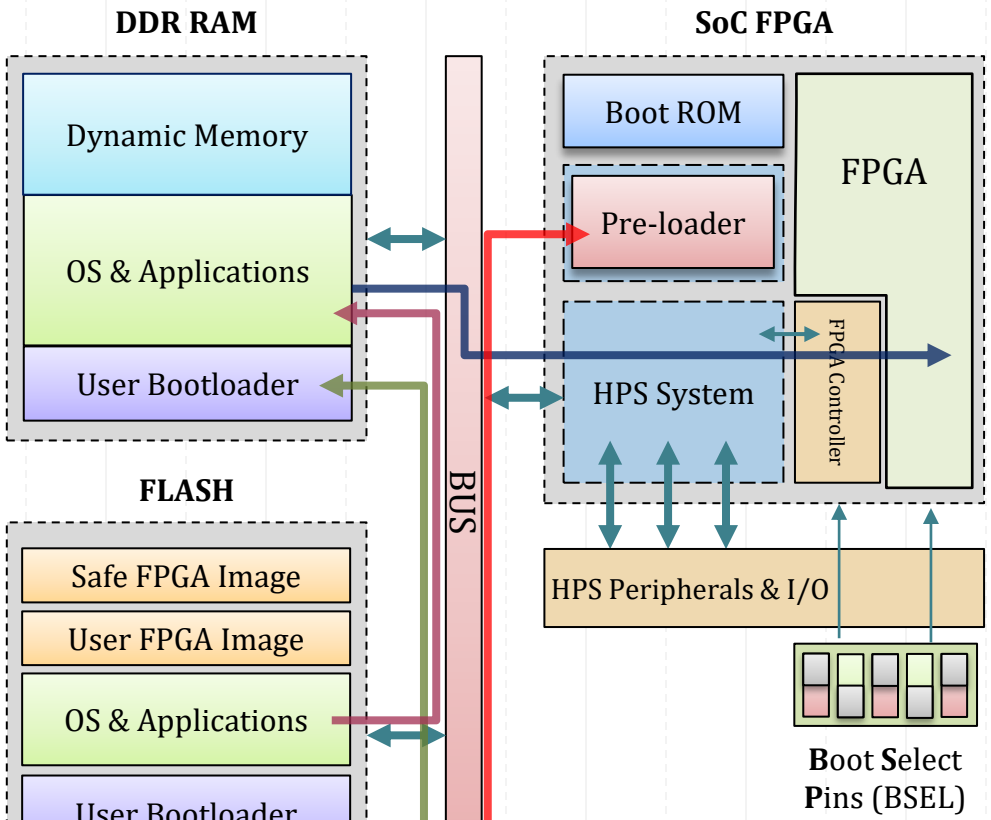
Lab 3 – DE1-SoC Tutorial



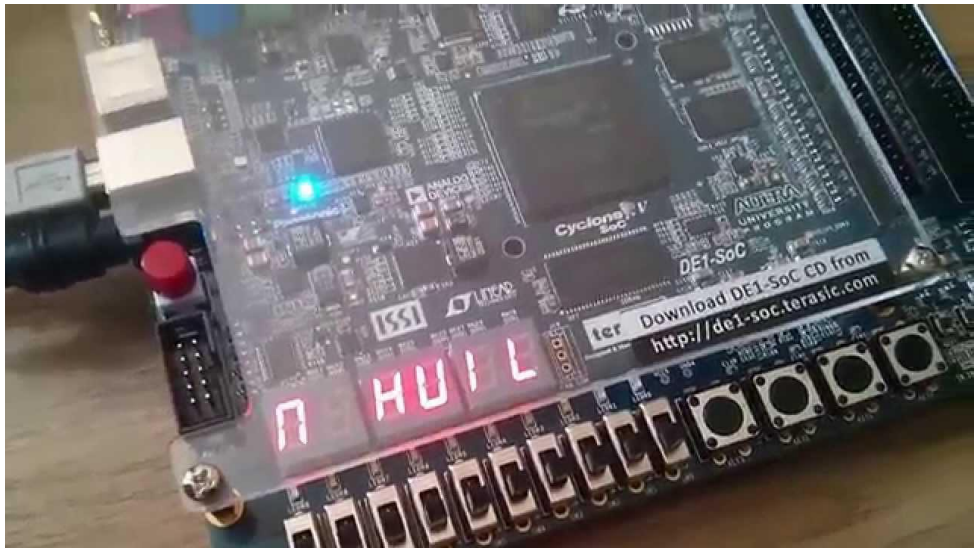


DE1-SoC Boot Stages

1. Reset
2. Boot from ROM
3. Setup boot source from BSEL pins
4. Copy Pre-loader to On-chip RAM
5. Run pre-loader
6. Setup HPS I/O and DDR
7. Configure FPGA (Optional)
8. Copy User Boot-loader to DDR RAM
9. Run User Boot-loader
10. Configure FPGA (optional)
11. Copy OS into DDR RAM
12. Run OS
13. Run Applications
14. Configure FPGA (optional)



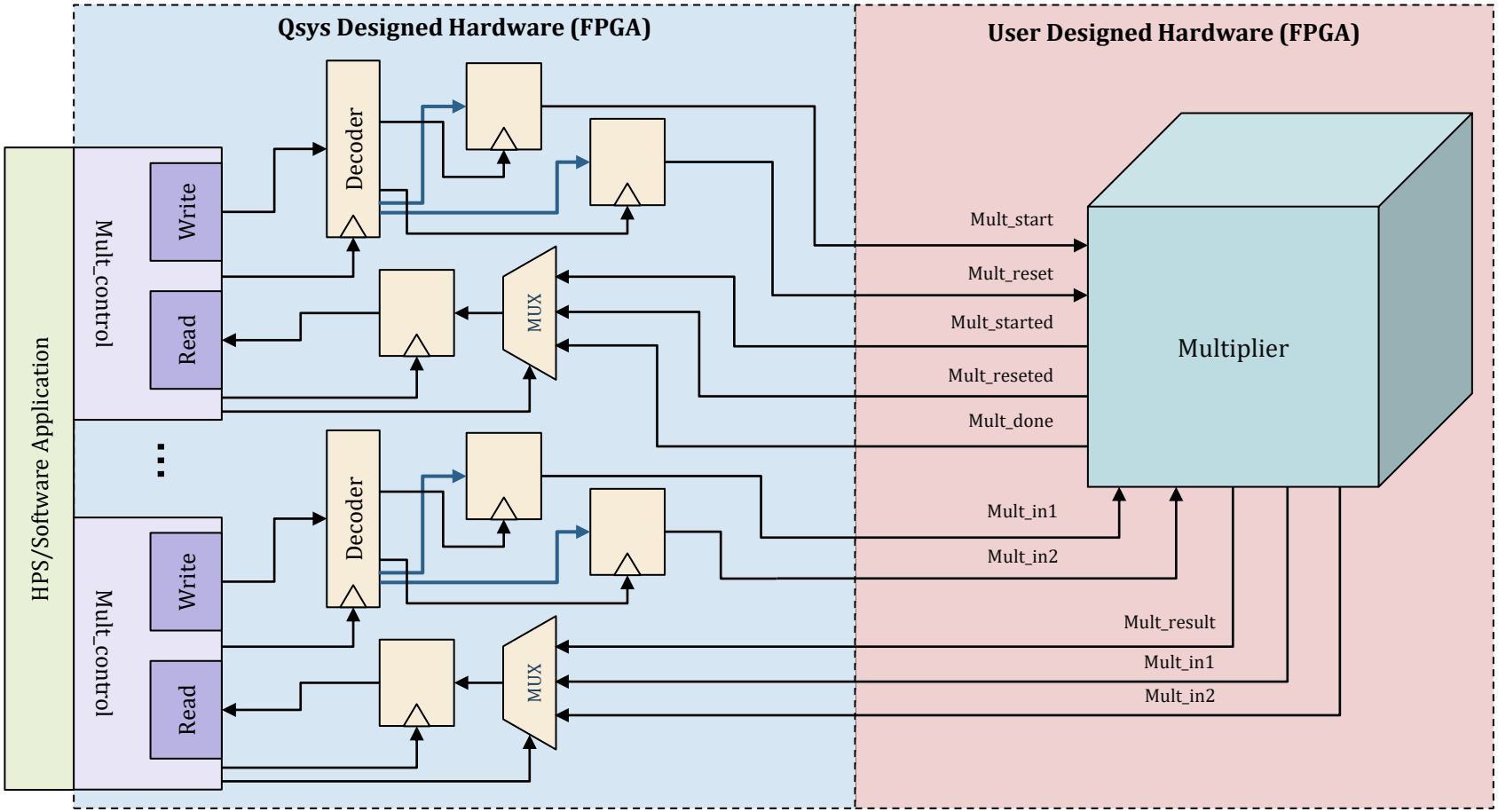
Lab3 – DE1-SoC Tutorial



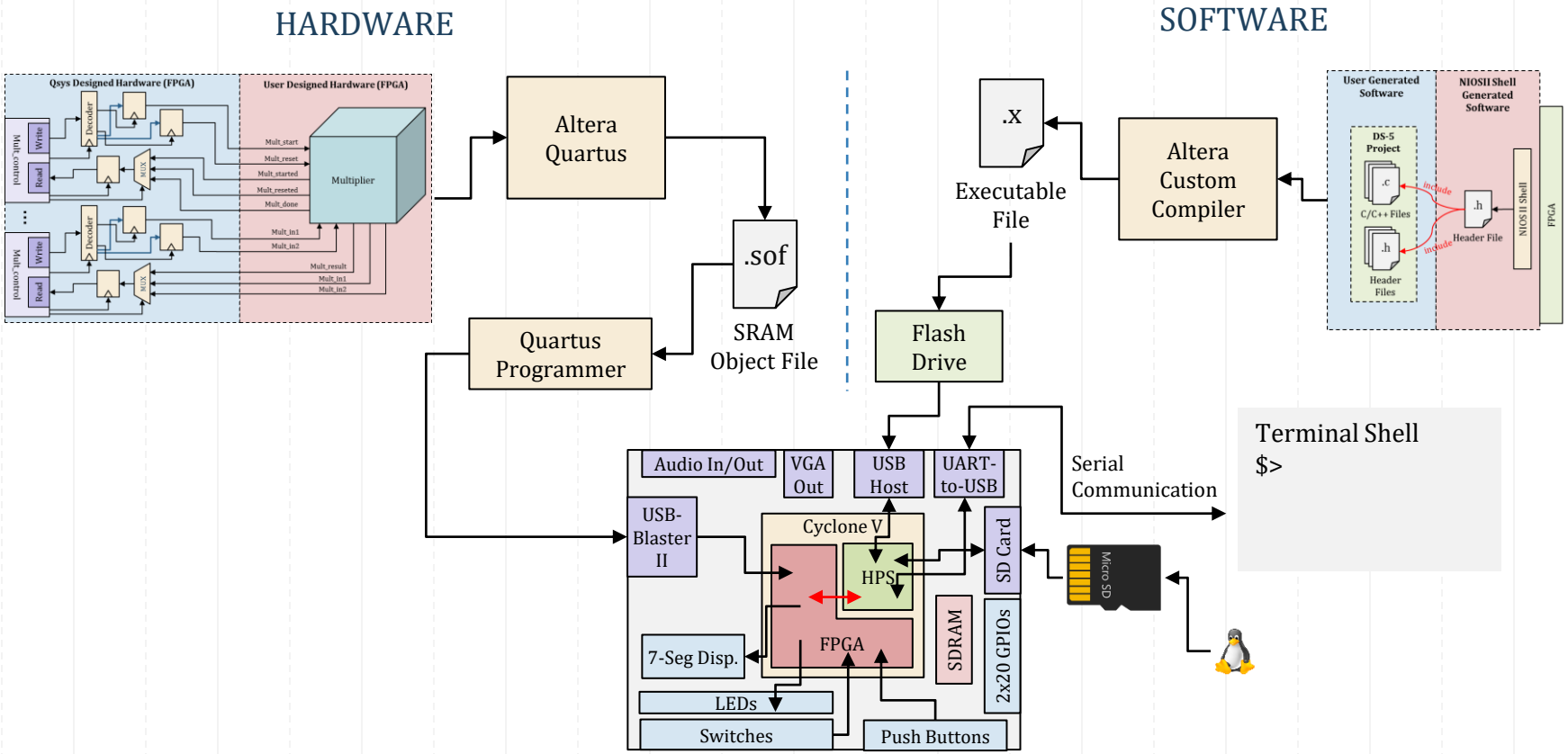
- Creating a HPS/FPGA which controls 7-seg and LEDs on FPGA through software on HPS

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

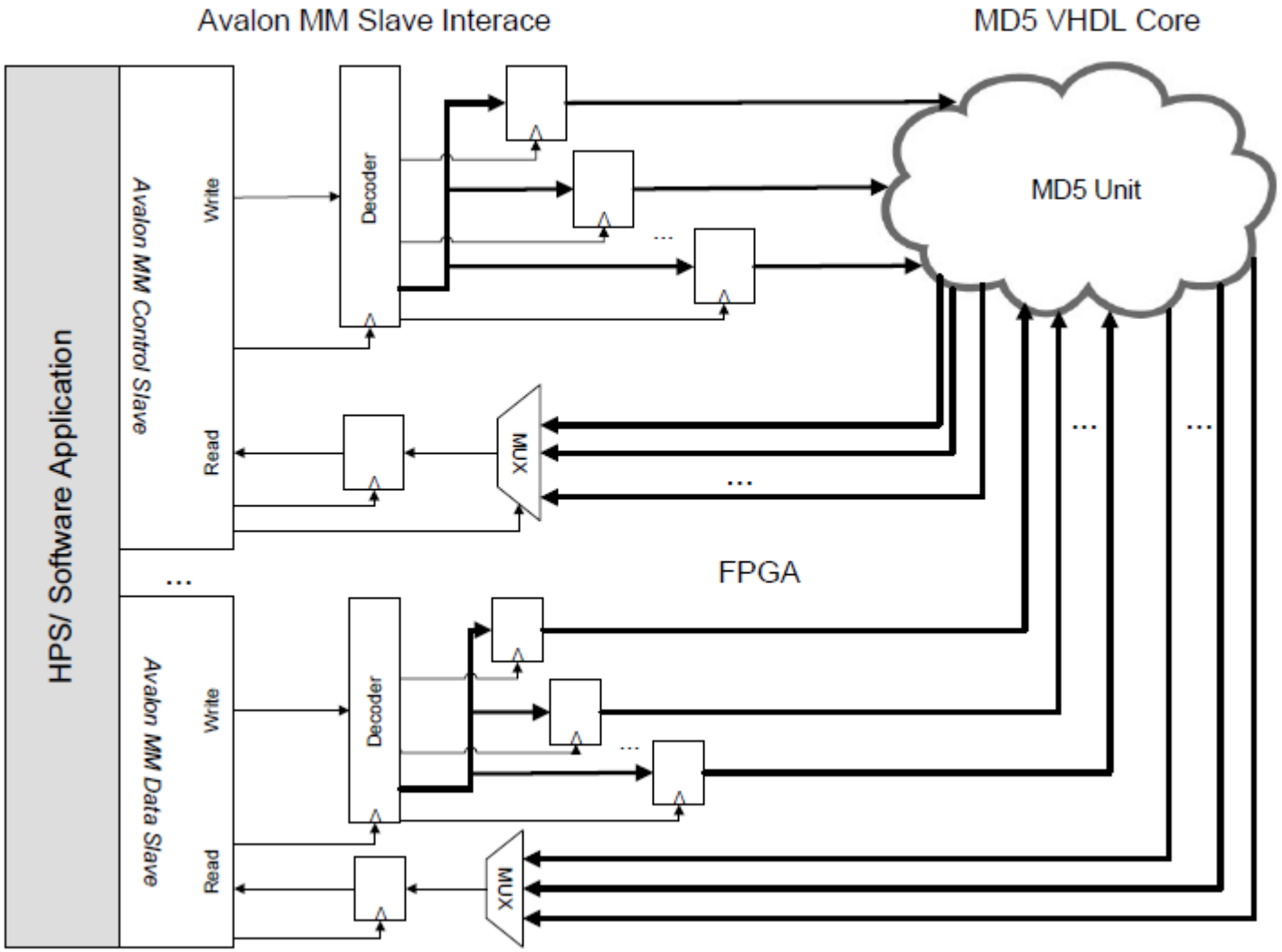
Lab 4 Multiplier IP



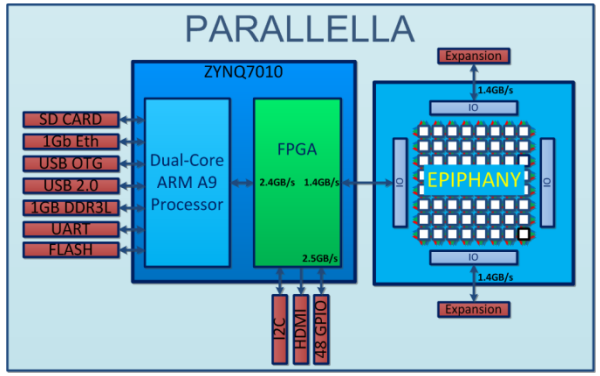
Putting it all together



MD5 Encryption Project



More SoC Boards for the Project ...



DE1-SoC MTL-2

https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=205&No=930#contents

Parallella Zynq SoC

https://parallella.org/