




R

COE838: System-on-Chip Design



HPS/FPGA Interconnection

Lab 3 and 4, Project manual
DE1-SoC Datasheets [online]
Flynn and Luk book – Chapter 3

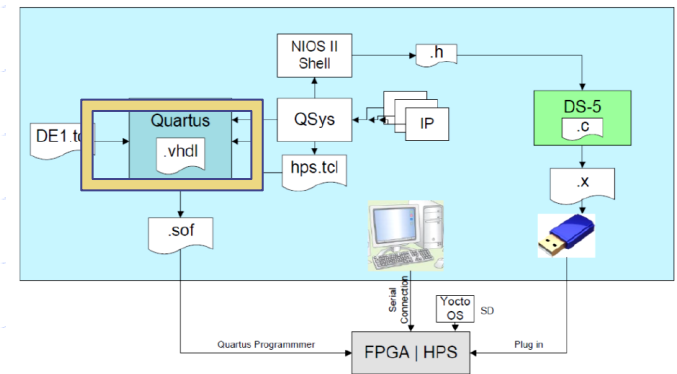


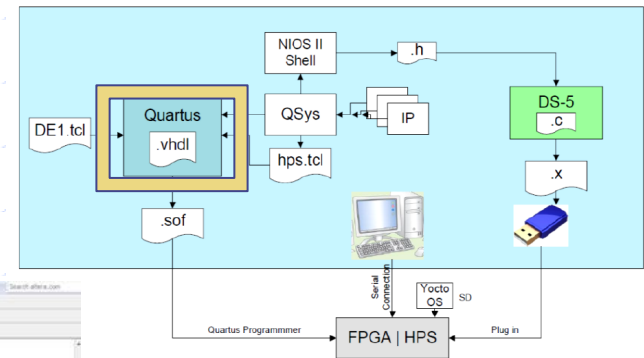
```

3  -- Lab 3
4  -- COES38 Systems-on-Chip Design
5
6  library ieee;
7  use ieee.std_logic_1164.all;
8  use ieee.numeric_std.all;
9
10 ENTITY LED_HEX_FPGA IS
11 PORT ( CLOCK_50, HPS_DDR3_REQ, HPS_ENET_RX_CLK, HPS_ENET_RX_DV          IN STD_LOGIC;
12       HPS_DDR3_ADDR          OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
13       HPS_DDR3_BA            OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
14       HPS_DDR3_CS_N         OUT STD_LOGIC;
15       HPS_DDR3_CK_N, HPS_DDR3_CKE          OUT STD_LOGIC;
16       HPS_USB_D1W, HPS_USB_HKT, HPS_USB_CLKOUT          IN STD_LOGIC;
17       HPS_ENET_RX_DATA      IN STD_LOGIC_VECTOR(3 DOWNTO 0);
18       HPS_SD_DATA, HPS_DDR3_DQS_N, HPS_DDR3_DQS_P      INOUT STD_LOGIC_VECTOR(3 DOWNTO 0);
19       HPS_ENET_MDIO         INOUT STD_LOGIC;
20       HPS_USB_DATA          INOUT STD_LOGIC_VECTOR(1 DOWNTO 0);
21       HPS_DDR3_DQ           INOUT STD_LOGIC_VECTOR(31 DOWNTO 0);
22       HPS_SD_CMD            INOUT STD_LOGIC;
23       HPS_ENET_TX_DATA, HPS_DDR3_DQS_IN, HPS_DDR3_RESET_N  OUT STD_LOGIC;
24       HPS_DDR3_ODT, HPS_DDR3_BA3_N, HPS_DDR3_RESET_N      OUT STD_LOGIC;
25       HPS_DDR3_CAS_N, HPS_DDR3_WE_N          OUT STD_LOGIC;
26       HPS_ENET_NDC, HPS_ENET_TX_EN          OUT STD_LOGIC;
27       LEDR                  OUT STD_LOGIC_VECTOR(8 DOWNTO 0);
28       HEX0, HEX1, HEX2, HEX3, HEX4, HEX5     BUFFER STD_LOGIC_VECTOR(6 DOWNTO 0);
29       HPS_USB_STP, HPS_SD_CLK, HPS_ENET_STX_CLK          IN STD_LOGIC;
30 END LED_HEX_FPGA;
31
32 ARCHITECTURE Behaviour OF LED_HEX_FPGA IS
33
34 --Instantiate the soc_system component here
35
36 --SIGNALS instantiated here
37
38 BEGIN
39
40 --port map soc_system here
41
42
43 End Behaviour;
44

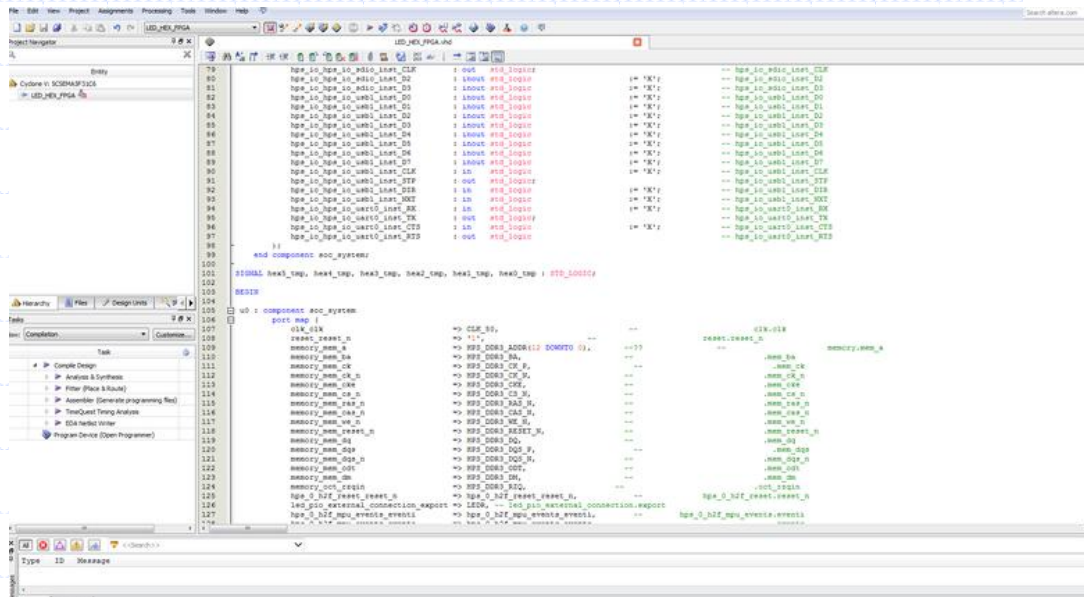
```

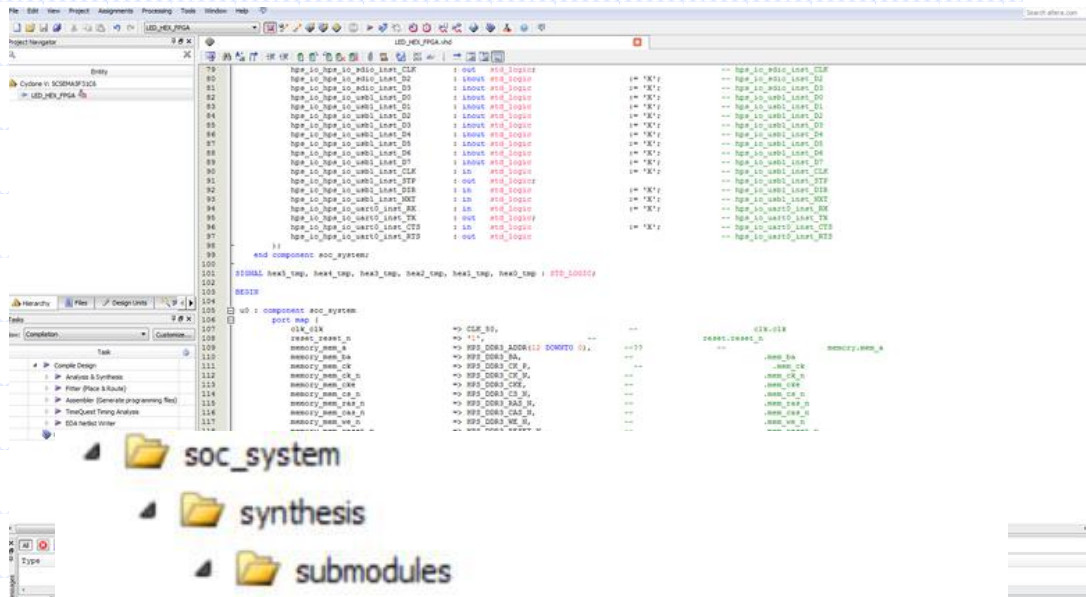
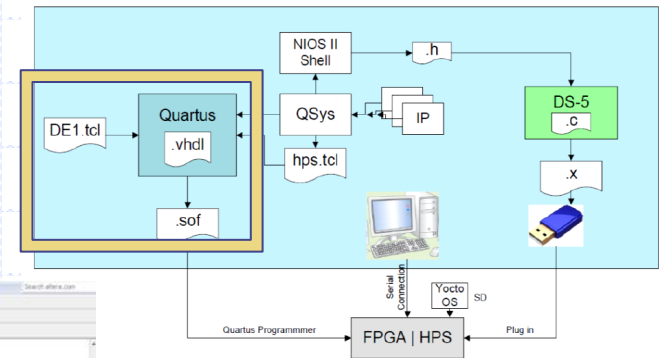
- Open Quartus, include top level files which represent your SoC system





- Port map top level with soc_system created in QSys

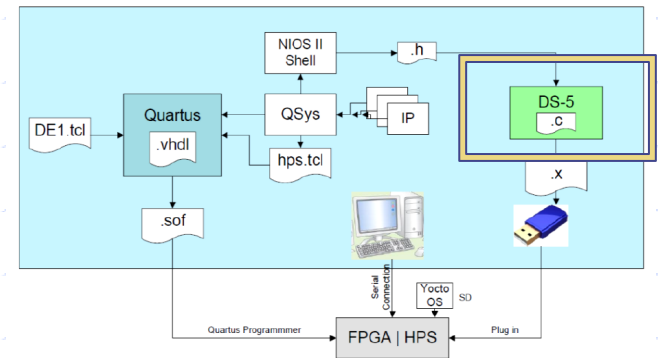




- hps_sdrnm_p0_parameters.tcl
- hps_sdrnm_p0_pin_assignments.tcl
- hps_sdrnm_p0_pin_map.tcl
- hps_sdrnm_p0_report_timing.tcl
- hps_sdrnm_p0_report_timing_core.tcl
- hps_sdrnm_p0_timing.tcl



- Pin assts, tcl scripts etc for IO and porting HPS and FPGA
- Compile/Synthesize



- Start a C project
- Create C application (memory map) using addresses generated by Qsys and NIOS

C Project
Create C project of selected type

Project name: HexLEDSW

Use default location
Location: C:\Users\Anita Tino\Documents\DS-5 Workspace\IntegerSor

Choose file system: default

Project type:
 Bare-metal Executable
 Bare-metal Library
 Executable
 Empty Project
 Hello World ANSI C Project
 Shared Library
 Static Library
 Makefile project

Toolchains:
 DS-5 GCC
 MinGW GCC

Show project types and toolchains only if they are supported on the platform

File Explorer:
 ui-hlp
 util
 Makefile
 Makefile.in
 winbuild.txt
 tools
 tutorial
 aclocal.m4
 AUTHORS
 ChangeLog
 ChangeLog.old
 config.guess
 config.rpath
 config.sub
 configure

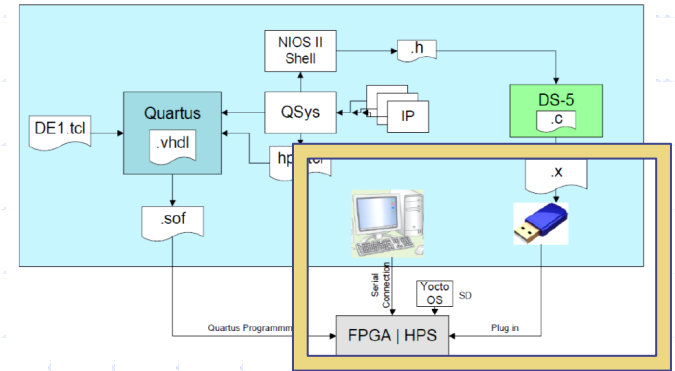
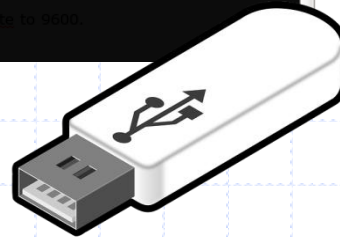
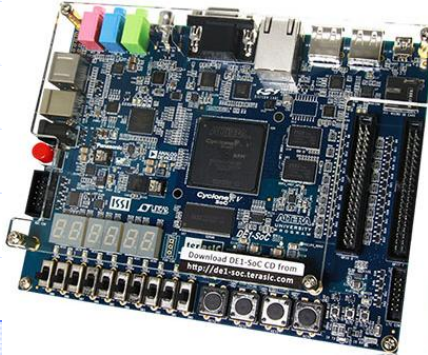
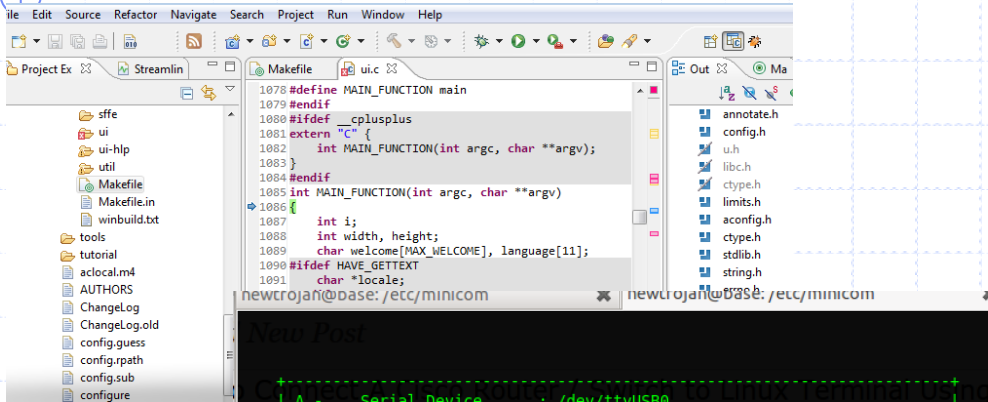
```

1082 int MAIN_FUNCTION(int argc, char **argv);
1083 }
1084 #endif
1085 int MAIN_FUNCTION(int argc, char **argv)
1086 {
1087     int i;
1088     int width, height;
1089     char welcome[MAX_WELCOME], language[11];
1090 #ifdef HAVE_GETTEXT
1091     char *locale;
1092 #endif
1093

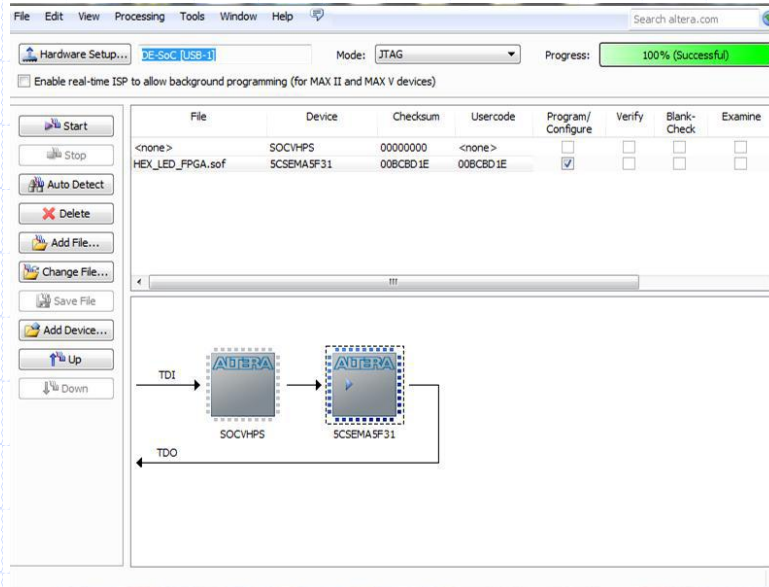
```

Tasks: 70 items

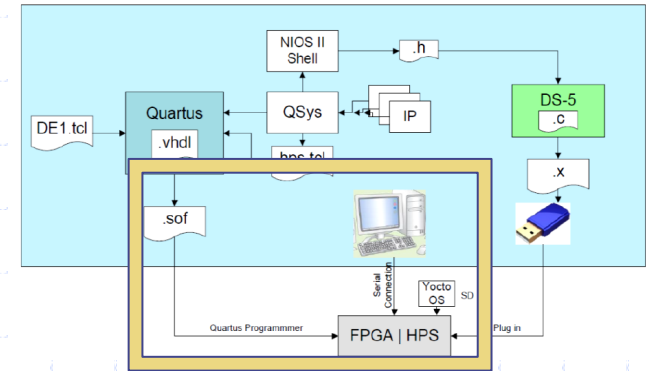
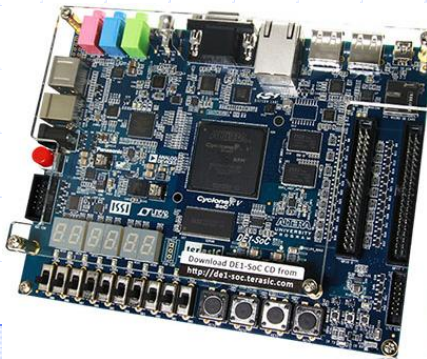
!	Description	Resource	Path
---	-------------	----------	------



- Compile & copy binary
- Start Minicom or DS-5 terminal
- Access Yocto linux, copy .x



launcher HexLEDsSW



- Program .sof on board (i.e. Bitstream)
- Execute the binary using host terminal

```

----- Iteration 27 -----
Reset done. Deasserting signal
Start successful
waiting for done
conversion done
0x0000001b * 0x0000001c = 0x000002f4. [Expected] 0x000002f4
[SUCCESSFUL]
----- Iteration 28 -----

```

Memory-Mapped IO

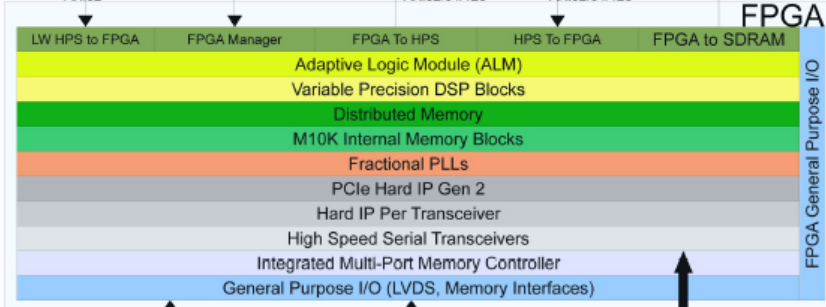
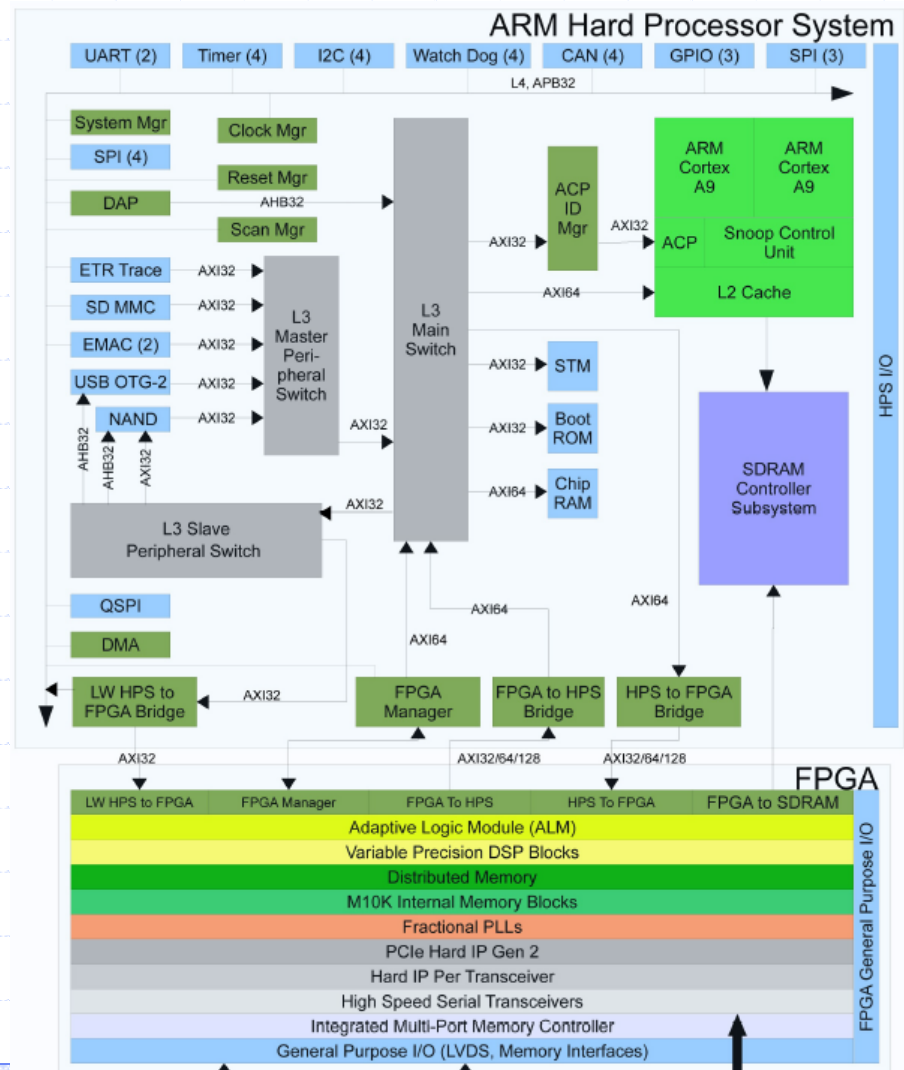
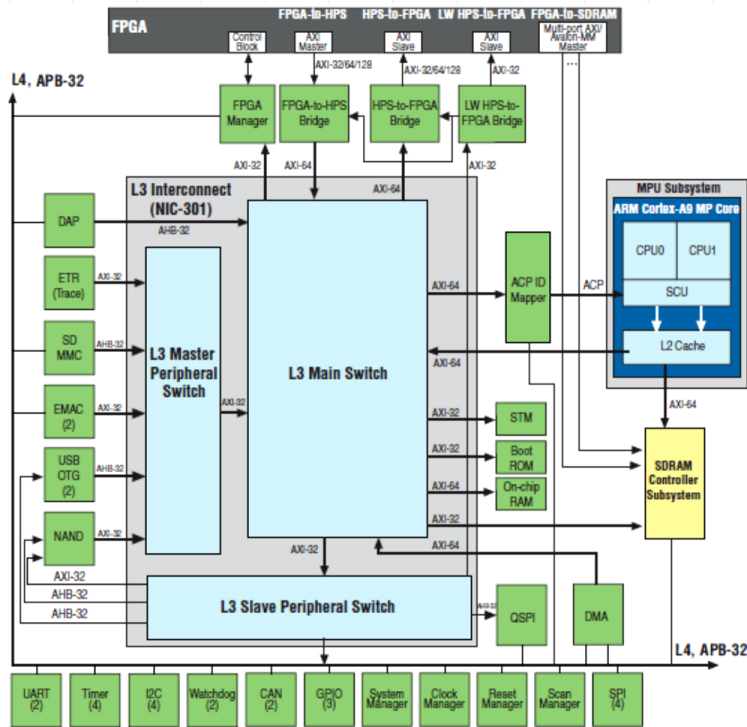
```
void *virtual_base;
#define LWHPS2FPGA_BASE 0xff200000
volatile uint32_t *h2p_lw_led_addr = NULL;
.....
//open the /dev/mem to access the FPGA space for reading and writing
if( ( fd = open( "/dev/mem", ( O_RDWR | O_SYNC ) ) ) == -1 ) {
    printf( "ERROR: could not open \"/dev/mem\"...\n" );
   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

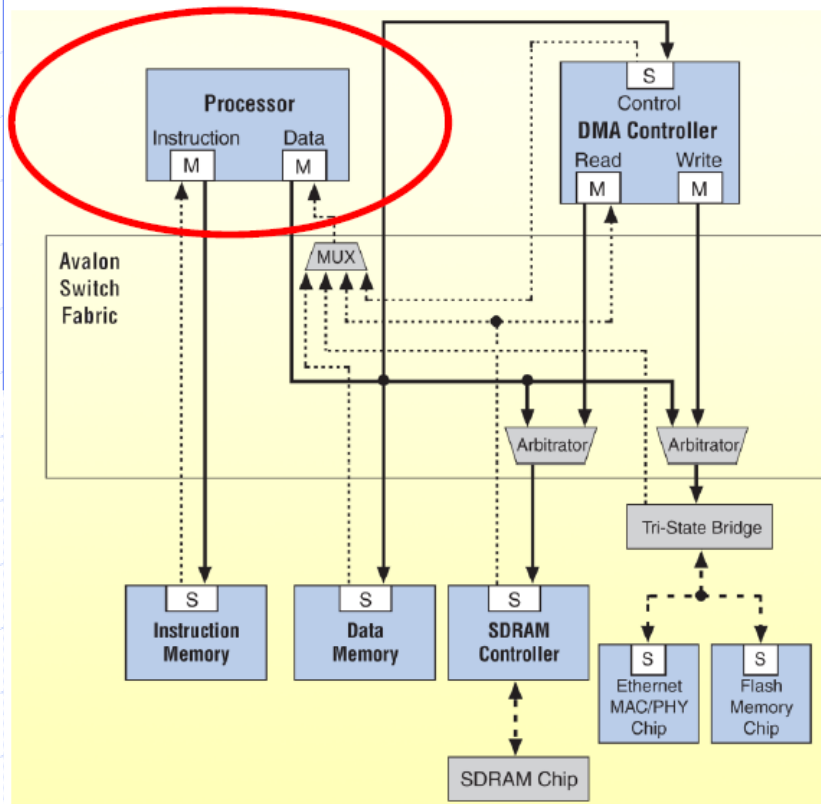
// map the address space for the LED and HEX registers into user space so
//we can interact with them.. virtual_base + the offset of your IP
component
h2p_lw_led_addr= virtual_base + ((uint32_t)(LED_PIO_BASE));
```



HPS/FPGA - ARM Cortex-A9

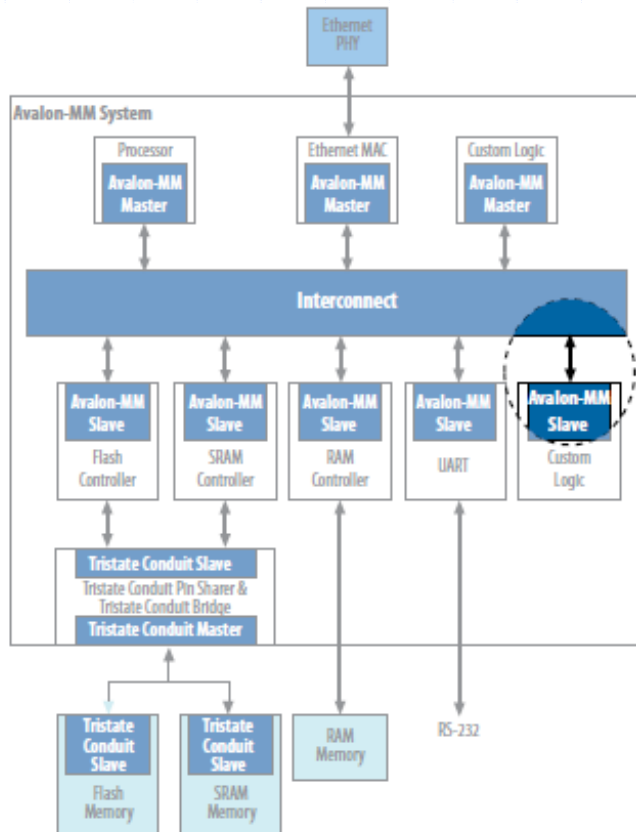


Avalon Bus



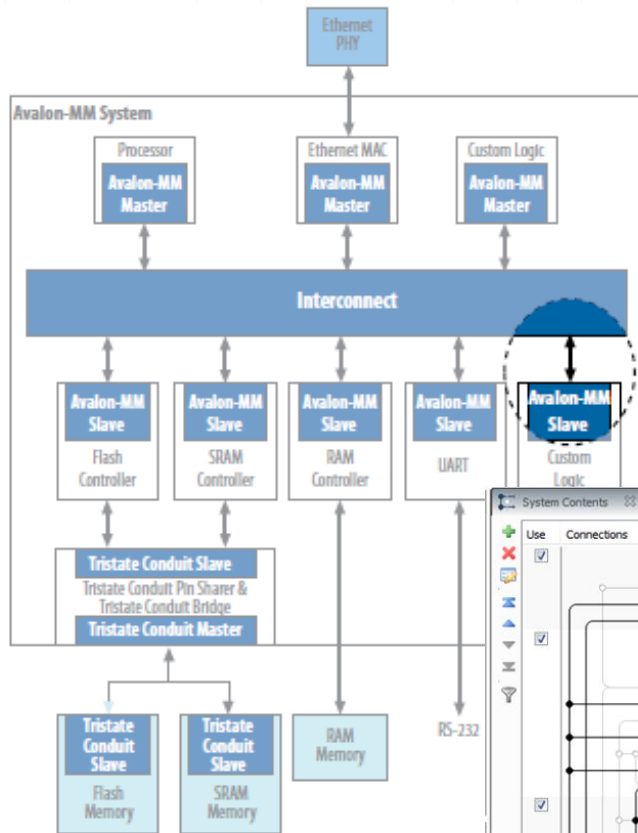
- Interconnect fabric inside FPGA (Altera)
- Used to connect master-slaves as required
- Generates the necessary “busses” using the fabric to make these connections
- Separates data in from data out
 - Uses multiplexers

Avalon Bus



- Follows a protocol
- You have an IP, however it does not necessarily know how to communicate with the FPGA fabric
- Need an Avalon-MM Slave (or Master) for controlling and w/r data
- Components we used in Lab3 were provided with Avalon-MM interfaces

Avalon Bus

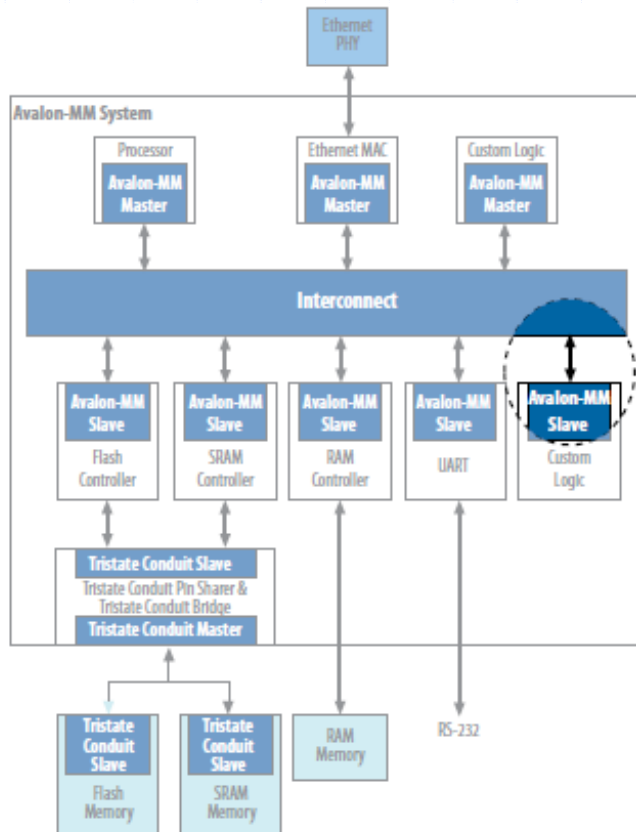


- Follows a protocol
- You have an IP, however it does not necessarily know how to communicate with the FPGA fabric

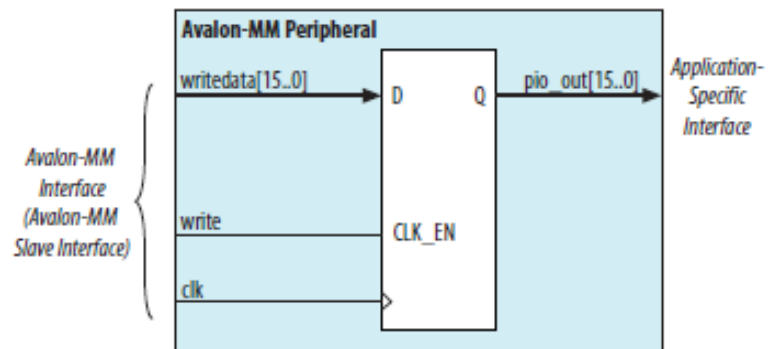
Need an Avalon-MM Slaving

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk_0	Clock Source	clk	exported		
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk_in_reset	clk_0		
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	clk			
<input checked="" type="checkbox"/>		clk	Clock Output	clk_reset			
<input checked="" type="checkbox"/>		clk_reset	Reset Output				
<input checked="" type="checkbox"/>		hps_0	Arria V/Cyclone V Hard Processor System	memory			
<input checked="" type="checkbox"/>		memory	Conduit	hps_io			
<input checked="" type="checkbox"/>		hps_io	Conduit	hps_0_h2f_reset			
<input checked="" type="checkbox"/>		h2f_reset	Reset Output				
<input checked="" type="checkbox"/>		h2f_axi_clock	Clock Input	memory	clk_0		
<input checked="" type="checkbox"/>		h2f_axi_master	AXI Master	h2f_axi_clock	[h2f_axi_clock]		
<input checked="" type="checkbox"/>		f2h_axi_clock	Clock Input	f2h_axi_slave	clk_0		
<input checked="" type="checkbox"/>		f2h_axi_slave	AXI Slave	h2f_jw_axi_clock	[f2h_axi_clock]		
<input checked="" type="checkbox"/>		h2f_jw_axi_clock	Clock Input	h2f_jw_axi_master	clk_0		
<input checked="" type="checkbox"/>		h2f_jw_axi_master	AXI Master	SEG7_IF_0	[h2f_jw_axi_clock]		
<input checked="" type="checkbox"/>		SEG7_IF_0	Avalon Memory Mapped Slave	avalon_slave			
<input checked="" type="checkbox"/>		avalon_slave	Conduit	conduit_end			
<input checked="" type="checkbox"/>		conduit_end	Conduit	dock_sink			
<input checked="" type="checkbox"/>		dock_sink	Clock Input	dock_sink_reset	clk_0	0x0000_0000	0x0000_001f
<input checked="" type="checkbox"/>		dock_sink_reset	Reset Input	led_pio	[clock_sink]		
<input checked="" type="checkbox"/>		led_pio	PPIO (Parallel I/O)	clk			
<input checked="" type="checkbox"/>		clk	Clock Input	reset	clk_0		
<input checked="" type="checkbox"/>		reset	Reset Input	s1	[clk]		
<input checked="" type="checkbox"/>		s1	Conduit	external_connection	[clk]	0x0000_0020	0x0000_002f

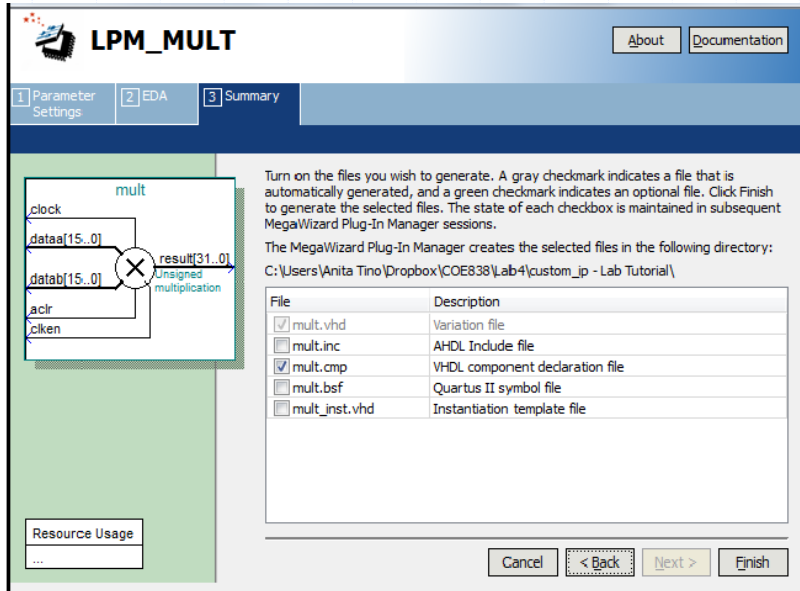
Avalon Bus – Custom IPs



- Need to create an Avalon-MM interface for your IP with the Avalon fabric
- Can also create a “wrapper” for integrating additional signals, not necessarily provided by your IP



Custom IP in HPS/FPGA



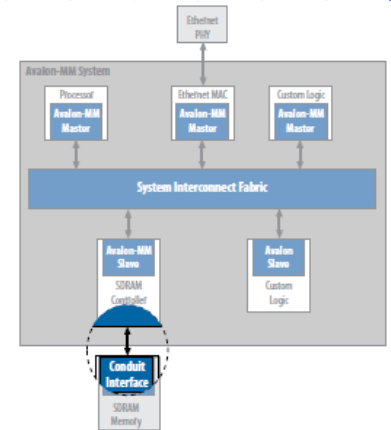
- Use IP Cores in Quartus to generate a custom multiplier (lab4)
- How do we determine when multiplication is complete from HPS side?



WRAPPERS FOR CUSTOM IPS



Conduits drive signals off-chip



```
U0 : soc_system
PORT MAP (
...
mult_data_0_mult_data_m_result      => mult_output_result,
mult_control_0_mult_control_m_done => "00000000000000000000000000000000" & done,
mult_data_0_mult_data_m_in1        => in1,
mult_data_0_mult_data_m_in2        => in2,
mult_control_0_mult_control_m_start => mult_input_start,
mult_control_0_mult_control_m_reset => mult_input_reset);

m0 : mult_unit
PORT MAP( clk => CLOCK_50, reset => mult_input_reset(0), enable => mult_input_start(0), mult_a =>
in1(15 DOWNT0 0), mult_b => in2(15 DOWNT0 0), mult_done => done, mult_result => mult_output_result);
```

Let's convert lab2a (multiplier) ->

HPS/FPGA CUSTOM IPS

Lab4 - Avalon MM Slave I/F

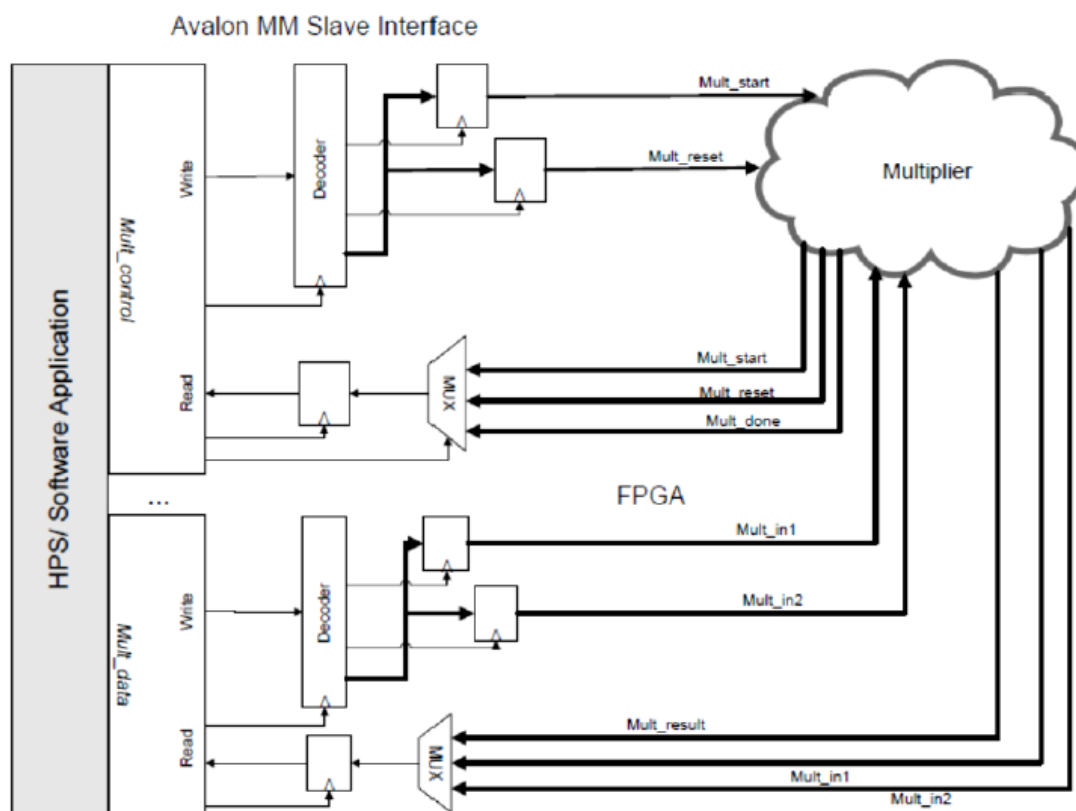
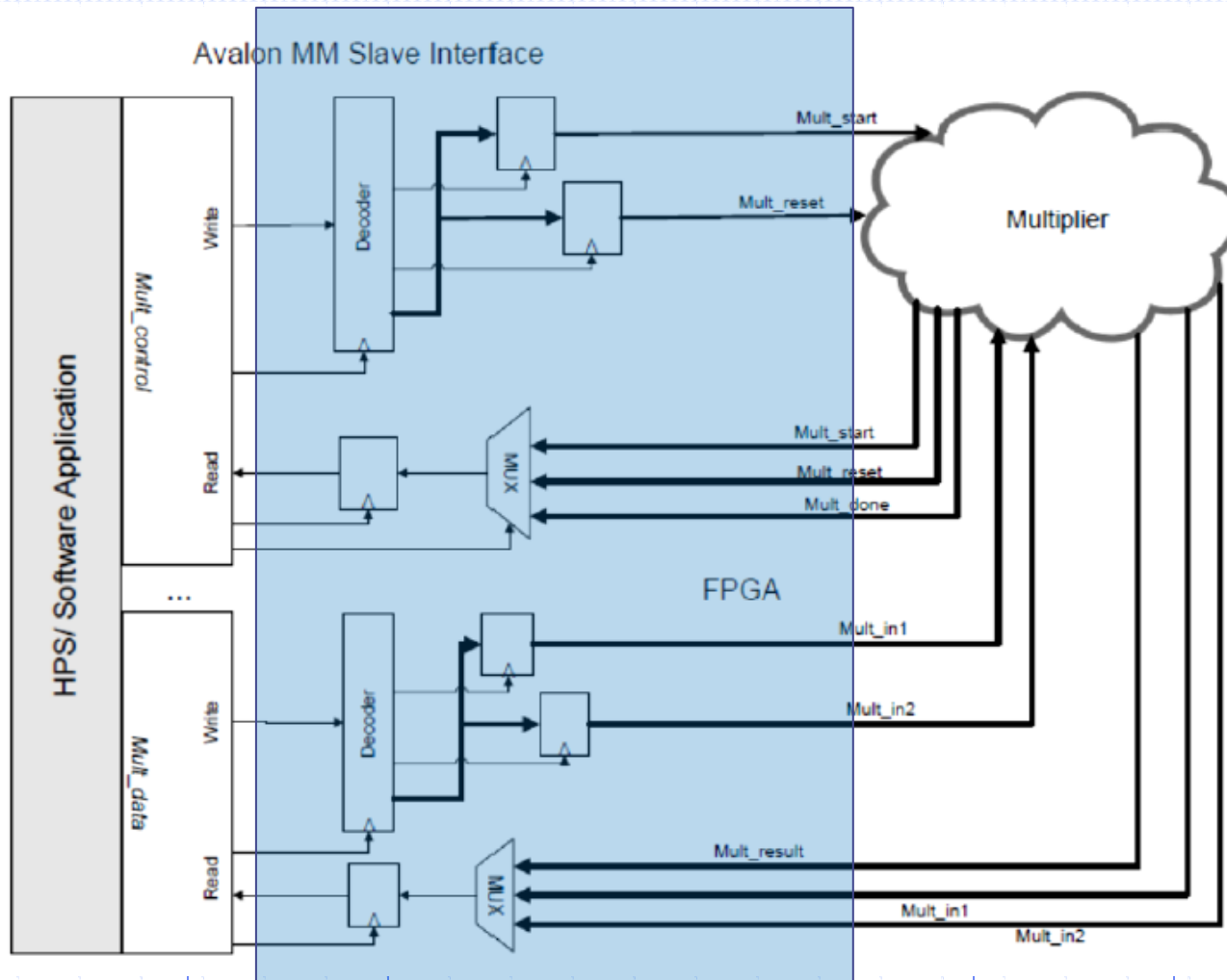


Table I: Common Avalon Bus Signals

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

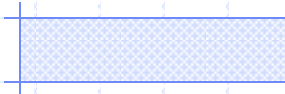
Lab4 - Avalon MM Slave I/F





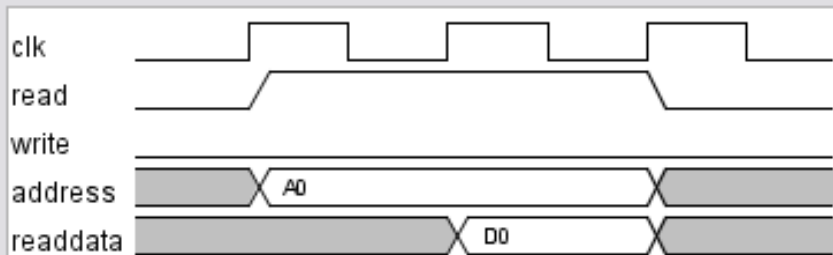
Common Timing Diagrams

READING & WRITING DATA

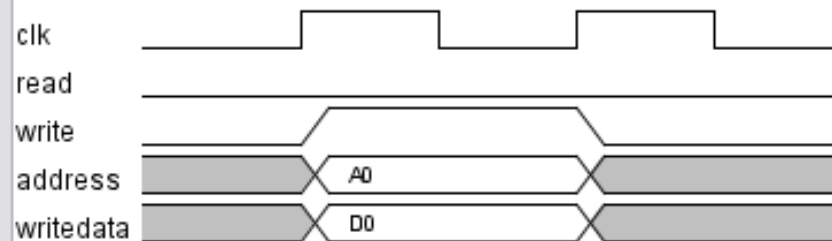


Avalon MM Slave Interface

Read Waveforms



Write Waveforms

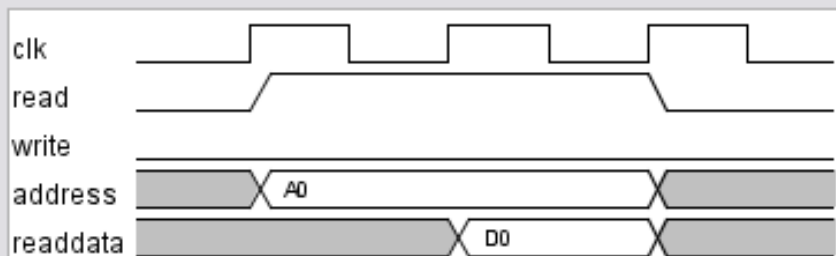


Avalon MM Slave Interface

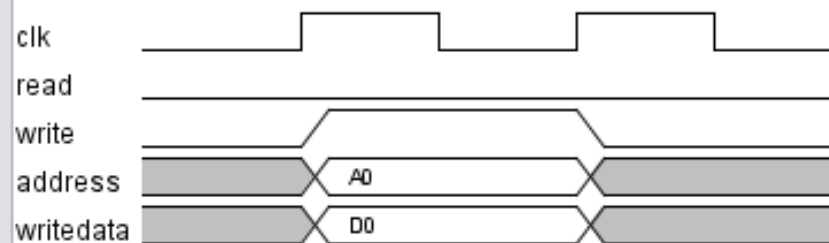
Table I: Common Avalon Bus Signals

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

Read Waveforms



Write Waveforms



Qsys: Create Interfaces

Qsys - soc_system.qsys (C:\Users\Anita Tino\Dropbox\COE838\Lab4\custom_ip - Lab Tutorial Solution\soc_system.qsys)

File Edit System Generate View Tools Help

IP Catalog System Contents Address Map Interconnect Requirements Device Family

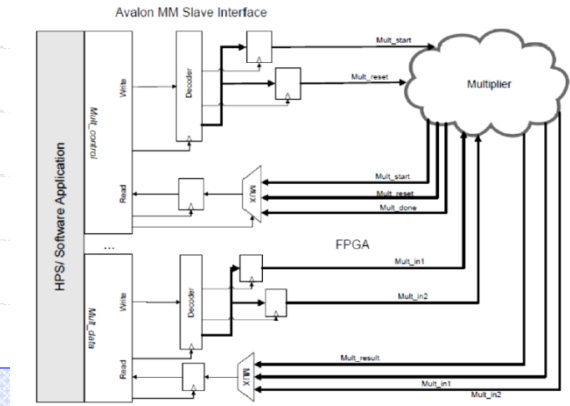
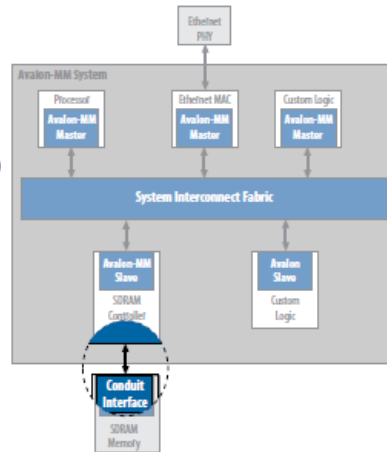
Project: New Component... mult_control mult_data System

Library: Basic Functions DSP Interface Protocols Memory Interfaces and Controllers PLL Processors and Peripherals Qsys Interconnect

Hierarchy: soc_system clk hps_0 h2f_reset

Use	Connections	Name	Description	Export	Clock	Base	End
<input checked="" type="checkbox"/>		clk_0	Clock Source				
		clk_in	Clock Input	clk			
		clk_in_reset	Reset Input	reset	<i>exported</i>		
		clk	Clock Output	<i>Double-click to export</i>	clk_0		
		clk_reset	Reset Output	<i>Double-click to export</i>			
<input checked="" type="checkbox"/>		hps_0	Arria V/Cyclone V Hard Processor System				
		memory	Conduit	memory			
		hps_io	Conduit	hps_io			
		h2f_reset	Reset Output	hps_0_h2f_reset			
		h2f_axi_clock	Clock Input	<i>Double-click to export</i>	clk_0		
		h2f_axi_master	AXI Master	<i>Double-click to export</i>	[h2f_axi_do...		
		f2h_axi_clock	Clock Input	<i>Double-click to export</i>	clk_0		
		f2h_axi_slave	AXI Slave	<i>Double-click to export</i>	[f2h_axi_do...	sl'	
		h2f_lw_axi_clock	Clock Input	<i>Double-click to export</i>	clk_0		
		h2f_lw_axi_master	AXI Master	<i>Double-click to export</i>	[h2f_lw_axi...		
<input checked="" type="checkbox"/>		mult_control_0	mult_control				
		s0	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clock]	sl' 0x0000_0040	0x0000_007F
		clock	Clock Input	<i>Double-click to export</i>	clk_0		
		reset	Reset Input	<i>Double-click to export</i>	[clock]		
		mult_control	Conduit	<i>Double-click to export</i>	[clock]		
<input checked="" type="checkbox"/>		mult_data_0	mult_data				
		s0	Avalon Memory Mapped Slave	<i>Double-click to export</i>	[clock]	sl' 0x0000_0000	0x0000_003F
		clock	Clock Input	<i>Double-click to export</i>	clk_0		
		reset	Reset Input	<i>Double-click to export</i>	[clock]		
		mult_data	Conduit	<i>Double-click to export</i>	[clock]		

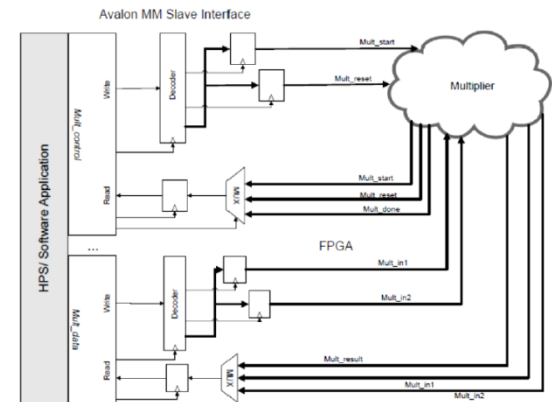
Conduits drive signals off-chip



Qsys: Create Interfaces

The screenshot shows the Qsys Component Editor interface. The main window displays the 'About Files' tab for a component named 'mult_data_hw.tcl'. It lists 'Synthesis Files' and 'Verilog Simulation Files'. A table shows the output path, source file, type, and attributes for the synthesis files. A warning message is visible in the Messages pane: 'Warning: mult_data: The QUARTUS_SYNTH fileset must specify the top-level module name.' To the right, an 'Avalon-MM System' diagram shows a central 'Interconnect' block connected to various masters and slaves. Masters include Processor, Ethernet MAC, and Custom Logic. Slaves include Flash Controller, SRAM Controller, RAM Controller, UART, and Custom Logic. A '46 KB' and '33 KB' label is present near the interconnect.

Use Qsys to create the Avalon-MM i/f so that your IP core may communicate through the FPGA fabric to the HPS (i.e. IP – MM I/F – Avalon bus - bridge – axi protocol) and vice versa

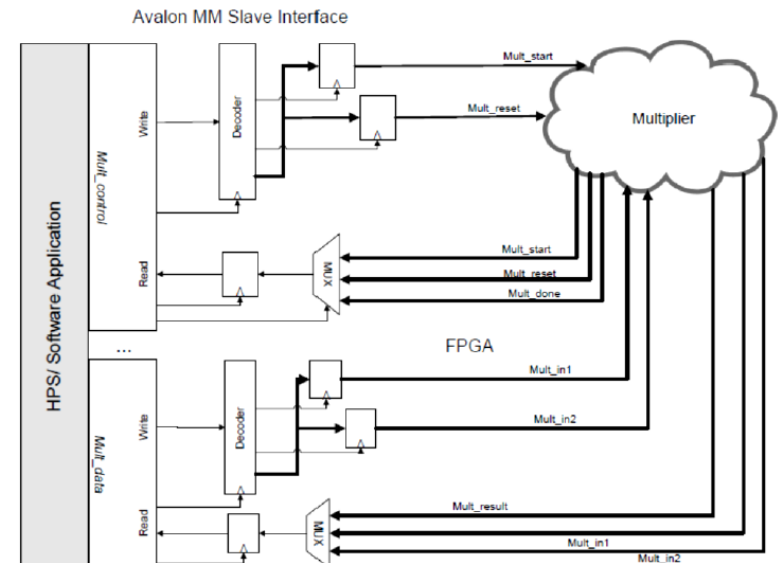


Avalon MM Slave Interface

```
IF(reset = '1') THEN
  avs_s0_readdata <= (OTHERS => '0');
  in1 <= (OTHERS => '0');
  in2 <= (OTHERS => '0');

ELSIF(rising_edge(clk)) THEN
  IF(avs_s0_read = '1') THEN
    CASE avs_s0_address IS
      WHEN "0000" =>
        avs_s0_readdata <= mult_result;
      WHEN "0001" =>
        avs_s0_readdata <= in1;
      WHEN "0010" =>
        avs_s0_readdata <= in2;
      WHEN OTHERS =>
        avs_s0_readdata <= (OTHERS => '0');
    END CASE;
  ELSIF(avs_s0_write = '1') THEN
    CASE avs_s0_address IS
      WHEN "0000" =>
        in1 <= "0000000000000000" & avs_s0_writedata(15 DOWNT0 0);
      WHEN "0001" =>
        in2 <= "0000000000000000" & avs_s0_writedata(15 DOWNT0 0);
      WHEN OTHERS =>
        --
    END CASE;
  END IF;
END IF;
```

```
entity mult_data is
  port (
    avs_s0_address : in  std_logic_vector(3 downto 0) := (others => '0'); --
    avs_s0_read    : in  std_logic                    := '0';           --
    avs_s0_write   : in  std_logic                    := '0';           --
    avs_s0_readdata : out std_logic_vector(31 downto 0); --
    avs_s0_writedata : in std_logic_vector(31 downto 0) := (others => '0'); --
    clk            : in  std_logic                    := '0';           --
    reset          : in  std_logic                    := '0';           --
    mult_in1       : out std_logic_vector(31 downto 0); -- mu
    mult_in2       : out std_logic_vector(31 downto 0); -- mu
    mult_result     : in  std_logic_vector(31 downto 0) := (others => '0') -- mul
  );
end entity mult_data;
```

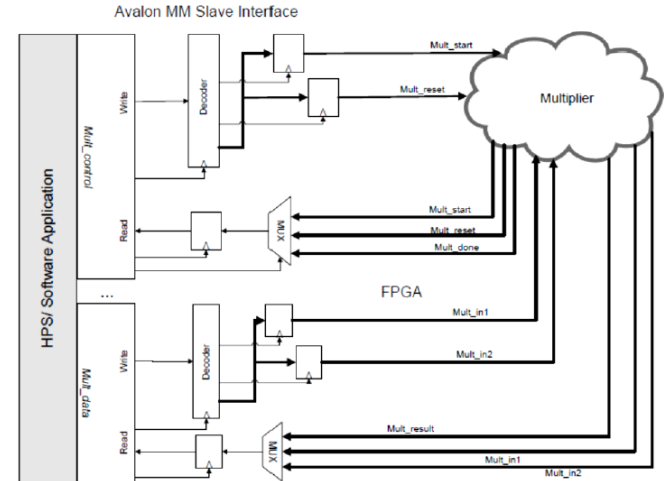


On the SW Side...

```
//initialize the addresses
mult_control = virtual_base + ((uint32_t) (MULT_CONTROL_0_BASE))
mult_data = virtual_base + ((uint32_t) (MULT_DATA_0_BASE));
```

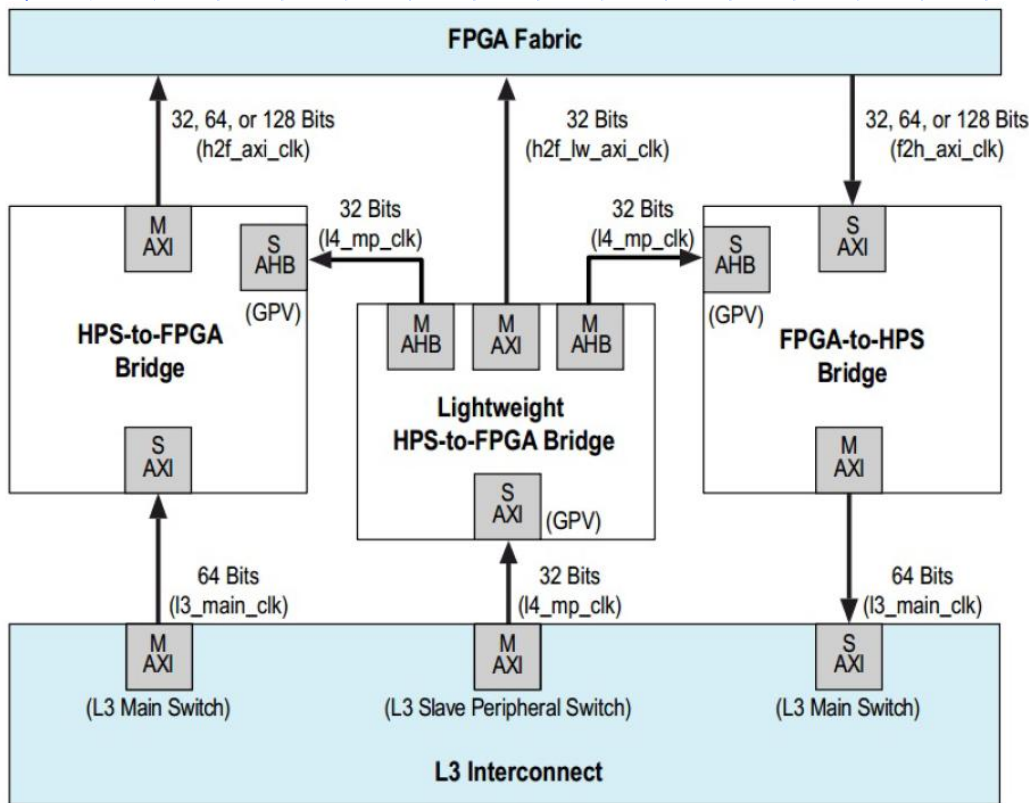
```
void copy_output() {
    uint32_t word, op1, op2;
    //wait for done
    printf("waiting for done\n");
    while(!(alt_read_word(mult_control+2) & 0x1));

    printf("conversion done\n");
    word = alt_read_word(mult_data+0);
    op1 = alt_read_word(mult_data+1);
    op2 = alt_read_word(mult_data+2);
    printf("0x%08x * 0x%08x = 0x%08x. [Expected] 0x%08x\n",
        word, op1, op2, word*op1);
    if(word == (op1*op2)) {
        printf("[SUCCESSFUL]\n");
        success++;
    } else {
```



```
IF(reset = '1') THEN
    avs_s0_readdata <= (OTHERS => '0');
    in1 <= (OTHERS => '0');
    in2 <= (OTHERS => '0');
ELSIF(rising_edge(clk)) THEN
    IF(avs_s0_read = '1') THEN
        CASE avs_s0_address IS
            WHEN "0000" =>
                avs_s0_readdata <= mult_result;
            WHEN "0001" =>
                avs_s0_readdata <= in1;
            WHEN "0010" =>
                avs_s0_readdata <= in2;
            WHEN OTHERS =>
                avs_s0_readdata <= (OTHERS => '0');
        END CASE;
    ELSIF(avs_s0_write = '1') THEN
        CASE avs_s0_address IS
            WHEN "0000" =>
                in1 <= "0000000000000000" & avs_s0_writedata(15 DOWNTO 0);
            WHEN "0001" =>
                in2 <= "0000000000000000" & avs_s0_writedata(15 DOWNTO 0);
            WHEN OTHERS =>
                --
        END CASE;
```


HPS – FPGA Bridges



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

-Connect your slaves to the required bus

-Qsys will make the necessary connections to the bridges for Avalon/AXI compatibility

Top-Level VHDL: Putting it all together

```
hps_io_hps_io_emac1_inst_TX_CTL => HPS_ENET_TX_EN, -- .hps_io_emac1_inst_TX_CTL
hps_io_hps_io_emac1_inst_RX_CLK => HPS_ENET_RX_CLK, -- .hps_io_emac1_inst_RX_CLK
hps_io_hps_io_emac1_inst_RXD1 => HPS_ENET_RX_DATA(1), -- .hps_io_emac1_inst_RXD1
hps_io_hps_io_emac1_inst_RXD2 => HPS_ENET_RX_DATA(2), -- .hps_io_emac1_inst_RXD2
hps_io_hps_io_emac1_inst_RXD3 => HPS_ENET_RX_DATA(3), -- .hps_io_emac1_inst_RXD3
hps_io_hps_io_sdio_inst_CMD => HPS_SD_CMD, -- .hps_io_sdio_inst_CMD
hps_io_hps_io_sdio_inst_D0 => HPS_SD_DATA(0), -- .hps_io_sdio_inst_D0
hps_io_hps_io_sdio_inst_D1 => HPS_SD_DATA(1), -- .hps_io_sdio_inst_D1
hps_io_hps_io_sdio_inst_CLK => HPS_SD_CLK, -- .hps_io_sdio_inst_CLK
hps_io_hps_io_sdio_inst_D2 => HPS_SD_DATA(2), -- .hps_io_sdio_inst_D2
hps_io_hps_io_sdio_inst_D3 => HPS_SD_DATA(3), -- .hps_io_sdio_inst_D3
hps_io_hps_io_usb1_inst_D0 => HPS_USB_DATA(0), -- .hps_io_usb1_inst_D0
hps_io_hps_io_usb1_inst_D1 => HPS_USB_DATA(1), -- .hps_io_usb1_inst_D1
hps_io_hps_io_usb1_inst_D2 => HPS_USB_DATA(2), -- .hps_io_usb1_inst_D2
hps_io_hps_io_usb1_inst_D3 => HPS_USB_DATA(3), -- .hps_io_usb1_inst_D3
hps_io_hps_io_usb1_inst_D4 => HPS_USB_DATA(4), -- .hps_io_usb1_inst_D4
hps_io_hps_io_usb1_inst_D5 => HPS_USB_DATA(5), -- .hps_io_usb1_inst_D5
hps_io_hps_io_usb1_inst_D6 => HPS_USB_DATA(6), -- .hps_io_usb1_inst_D6
hps_io_hps_io_usb1_inst_D7 => HPS_USB_DATA(7), -- .hps_io_usb1_inst_D7
hps_io_hps_io_usb1_inst_CLK => HPS_USB_CLKOUT, -- .hps_io_usb1_inst_CLK
hps_io_hps_io_usb1_inst_STP => HPS_USB_STP, -- .hps_io_usb1_inst_STP
hps_io_hps_io_usb1_inst_DIR => HPS_USB_DIR, -- .hps_io_usb1_inst_DIR
hps_io_hps_io_usb1_inst_NXT => HPS_USB_NXT, --
hps_0_h2f_reset_reset_n => reset_reset_n,
mult_data_0_mult_data_m_result => mult_output_result, -- mult_output_0_mult_output.result
mult_control_0_mult_control_m_done => "00000000000000000000000000000000" & done, --<<<<<< .done
mult_data_0_mult_data_m_in1 => in1, -- mult_input_0_mult_input.in1
mult_data_0_mult_data_m_in2 => in2, -- .in2
mult_control_0_mult_control_m_start => mult_input_start, -- .start
mult_control_0_mult_control_m_reset => mult_input_reset -- .reset
);

m0 : mult_unit
PORT MAP( clk => CLOCK_50, reset => mult_input_reset(0), enable => mult_input_start(0), mult_a => in1(15 DOWNTO 0), mult_b => in2(15 DOWNTO 0),
mult_done => done, mult_result => mult_output_result);
```

END Behaviour;

HPS Software: Putting it all together

```
void copy_output(){
    uint32_t word, op1, op2;
    //wait for done
    printf("waiting for done\n");
    while(!(alt_read_word(mult_control+2) & 0x1));

    printf("conversion done\n");
    word = alt_read_word(mult_data+0);
    op1 = alt_read_word(mult_data+1);
    op2 = alt_read_word(mult_data+2);
    printf("0x%08x * 0x%08x = 0x%08x. [Expected] 0x%08x\n", op1, op2, word, (op1*op2));
    if(word == (op1*op2)){
        printf("[SUCCESSFUL]\n");
        success++;
    }else{

```

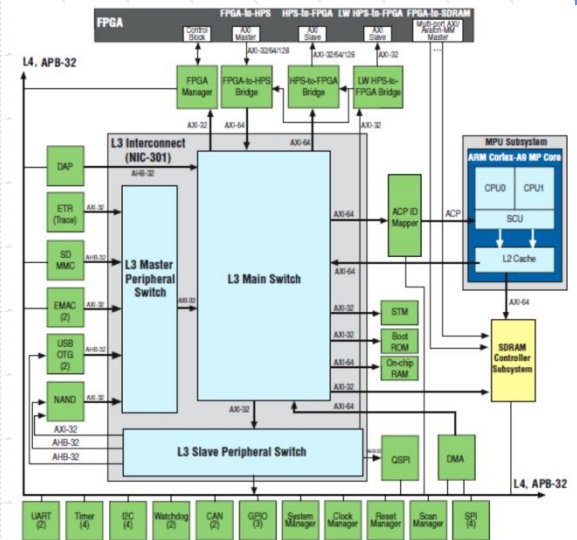
```
----- Iteration 27 -----
Reset done. Deasserting signal
Start successful
waiting for done
conversion done
0x0000001b * 0x0000001c = 0x000002f4. [Expected] 0x000002f4
[SUCCESSFUL]

----- Iteration 28 -----
Reset done. Deasserting signal
Start successful
waiting for done
conversion done
0x0000001c * 0x0000001d = 0x0000032c. [Expected] 0x0000032c
[SUCCESSFUL]

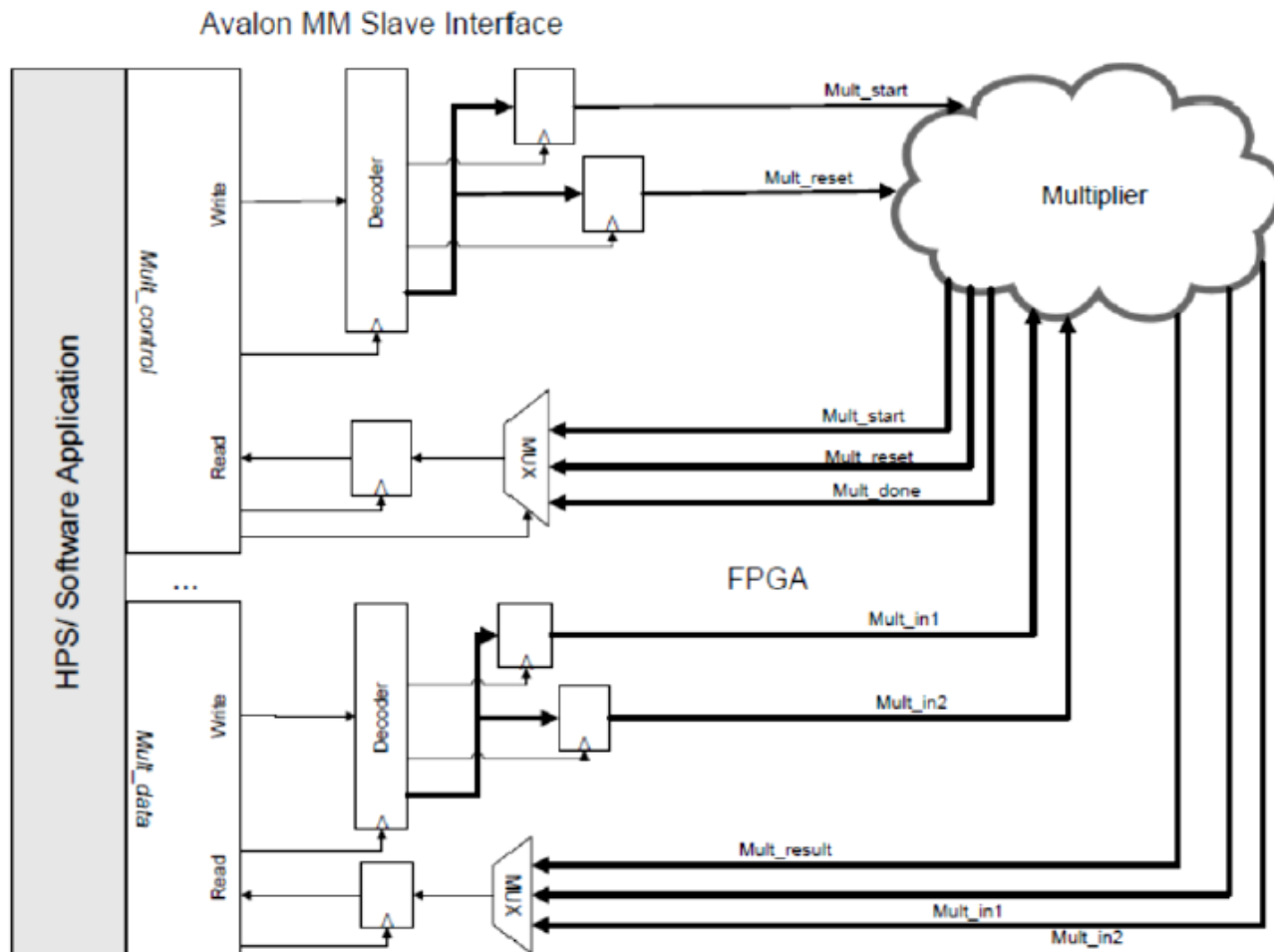
----- Iteration 29 -----
Reset done. Deasserting signal
Start successful
waiting for done
conversion done
0x0000001d * 0x0000001e = 0x00000366. [Expected] 0x00000366
[SUCCESSFUL]

----- Iteration 30 -----
Reset done. Deasserting signal
Start successful
waiting for done
conversion done
0x0000001e * 0x0000001f = 0x000003a2. [Expected] 0x000003a2
[SUCCESSFUL]

-----
[TEST PASSED] 30/30
```



Lab4 - Avalon MM Slave I/F



Lab 4 – Your Assignment

- Create a custom functioned SoC of your choice
 - Obviously, DO NOT use the multiplier as your design
- Points awarded for creativity
- Divide custom IP is obviously not that creative – just doing the opposite!



Project : **MD5 Decryption SoC Design**



MD5 Algorithm

```
//Note: All variables are unsigned 32 bit and wrap
var int[64] s, K
//s specifies the per-round shift amounts
s[ 0..15] := { 7, 12, 17, 22, 7, 12, 17, 22, 7,
s[16..31] := { 5, 9, 14, 20, 5, 9, 14, 20, 5,
s[32..47] := { 4, 11, 16, 23, 4, 11, 16, 23, 4,
s[48..63] := { 6, 10, 15, 21, 6, 10, 15, 21, 6,

//K constants
K[ 0.. 3] := { 0xd76aa478, 0xe8c7b756, 0x242070db,
K[ 4.. 7] := { 0xf57c0faf, 0x4787c62a, 0xa8304613,
K[ 8..11] := { 0x698098d8, 0x8b44f7af, 0xffff5bb1,
K[12..15] := { 0x6b901122, 0xfd987193, 0xa679438e,
K[16..19] := { 0xf61e2562, 0xc040b340, 0x265e5a51,
K[20..23] := { 0xd62f105d, 0x02441453, 0xd8a1e681,
K[24..27] := { 0x21e1cde6, 0xc33707d6, 0xf4d50d87,
K[28..31] := { 0xa9e3e905, 0xfcefa3f8, 0x676f02d9,
K[32..35] := { 0xffffa3942, 0x8771f681, 0x6d9d6122,
K[36..39] := { 0xa4bbee44, 0x4bdecfa9, 0xf6bb4b60,
K[40..43] := { 0x289b7ec6, 0xeaal27fa, 0xd4ef3085,
K[44..47] := { 0xd9d4d039, 0xe6db99e5, 0x1fa27cf8,
K[48..51] := { 0xf4292244, 0x432aff97, 0xab9423a7,
K[52..55] := { 0x655b59c3, 0x8f0ccc92, 0xffeff47d,
K[56..59] := { 0x6fa87e4f, 0xfe2ce6e0, 0xa3014314,
K[60..63] := { 0xf7537e82, 0xbd3af235, 0x2ad7d2bb,

//Initialize variables:
var int a0 := 0x67452301 //A
var int b0 := 0xefcdab89 //B
var int c0 := 0x98badcfe //C
var int d0 := 0x1032547e //D

//Process the message in successive 512-bit chunks:
for each 512-bit chunk of message
    break chunk into sixteen 32-bit words M[j], 0 ≤ j ≤ 15

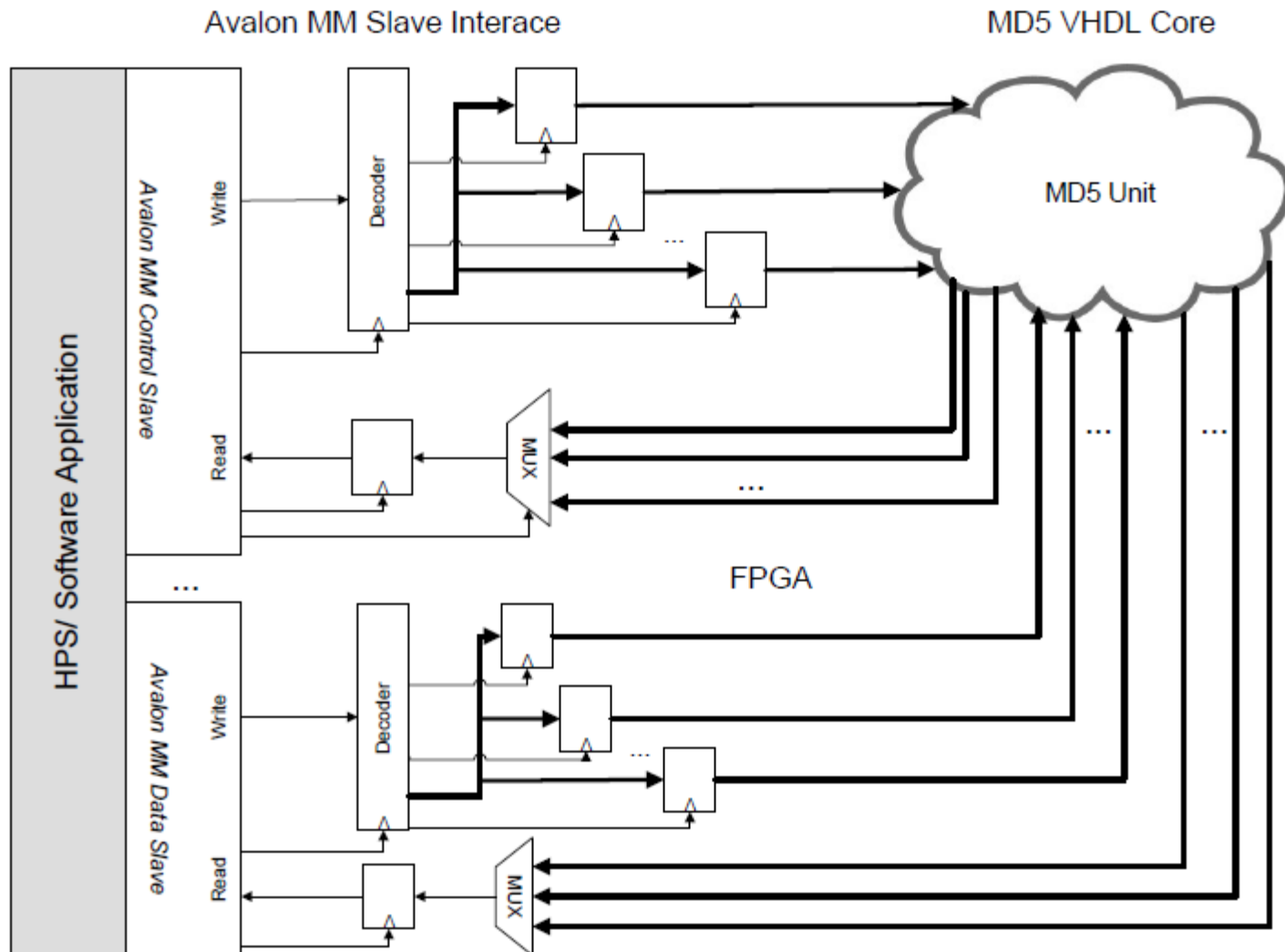
//Initialize hash value for this chunk:
var int A := a0    var int B := b0
var int C := c0    var int D := d0

//Main loop:
for i from 0 to 63
    for i from 0 to 63
        if 0 ≤ i ≤ 15 then
            F := (B and C) or ((not B) and D)
            g := i
        else if 16 ≤ i ≤ 31
            F := (D and B) or ((not D) and C)
            g := (5*i + 1) mod 16
        else if 32 ≤ i ≤ 47
            F := B xor C xor D
            g := (3*i + 5) mod 16
        else if 48 ≤ i ≤ 63
            F := C xor (B or (not D))
            g := (7*i) mod 16
        dTemp := D
        D := C
        C := B
        B := B + lefttrotate((A + F + K[i] + M[g]), s[i])
        A := dTemp
    end for
//Add this chunk's hash to result so far:
a0 := a0 + A
b0 := b0 + B
c0 := c0 + C
d0 := d0 + D
end for

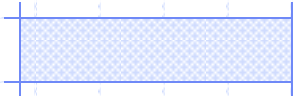
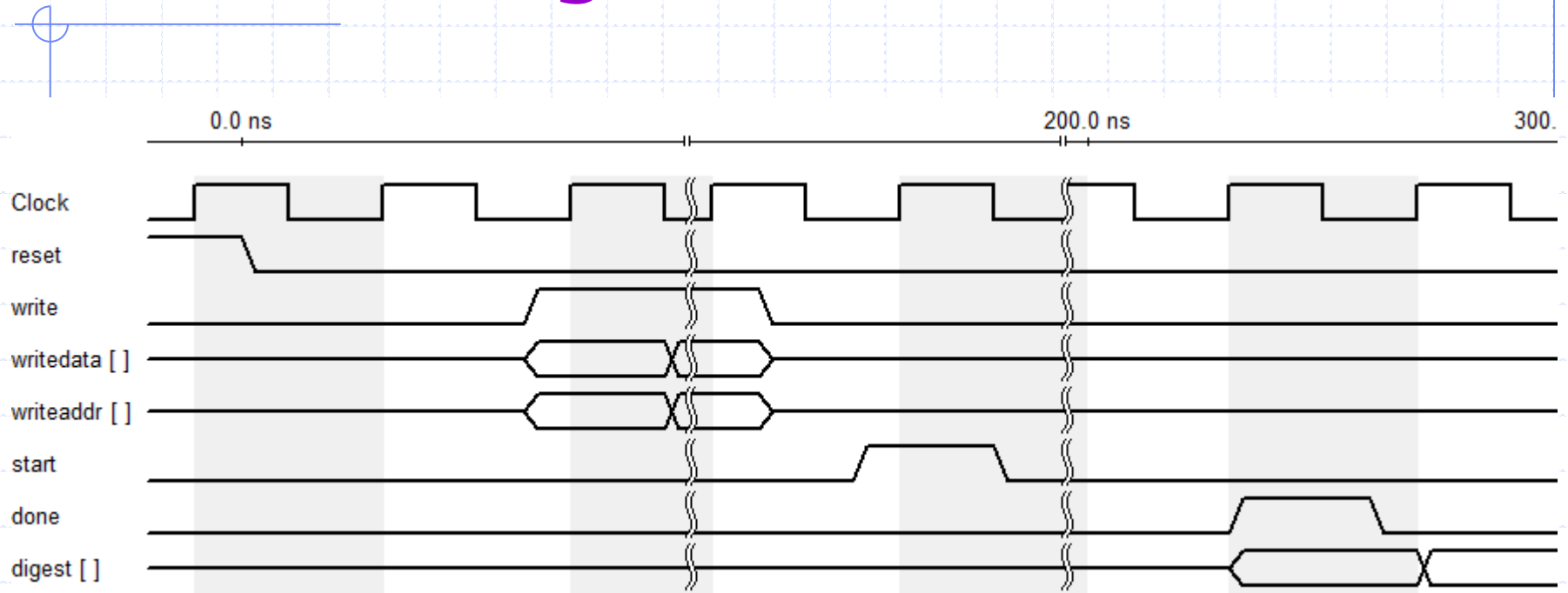
var char digest[16] := a0 append b0 append c0 append d0 //(Output is in little-endian)

//lefttrotate function definition
lefttrotate (x, c)
    return (x << c) binary or (x >> (32-c));
```

MD5 Standard Project



MD5 Timing Characteristics



MD5 Project

- **Analyze** the MD5 Core – Obtain a thorough understanding
 - VHDL RTL analysis
 - Test bench => 1 core vs 32 cores
 - Timing properties in ModelSim & lab manual
- **Design Avalon MM Interface**
- **Design HPS Software Application**
 - Generate Messages
 - Send *constant* data, send message
 - Receive digest when complete
 - Calculate hash time, # of hashes, hash rate, correct answer etc
- Parallel vs Sequential
- **Formal report** must follow specifications
- Bonus projects = 2-5% bonus on final COE838 grade



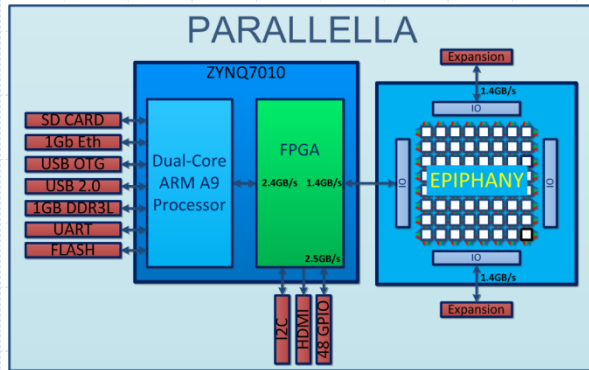
Some Bonus Projects ...

DE1-SoC MTL

- Develop an SoC that can be controlled by an Android app running on the DE1-SoC
 - Special report



Some Bonus Projects ...



Parallella – making heterogeneous high performance parallel platforms attainable to general public (16 – 64 cores on a board)

- Create a MM application (i.e video processing, bitcoin etc) for the Parallella, determining statistics and providing comparisons to another platform (i.e. X86)

<https://www.youtube.com/watch?v=hFWIC3RF0f8>

Some Bonus Projects ...

- DE1-SoC – video processing SoC design, inputs a graphic and displays an altered graphic on VGA (or screen)
 - Picture or video filtering using SoCs
- HLS – LegUp vs custom design (or IP) to develop same hardware for an SoC on DE1-SoC. Compare various stats of performance, power, area etc

Mini Bonus

- +1-2% on project mark - .c application equivalent which performs MD5 decryption. Use pure HPS vs HPS/FPGA vs x86 (compare stats)
 - +2-5% Implement above + Use h2f bus (64b, 128b) & compare statistics