Intel SOC Design Workshop-2

Subham Ball
Dept. of System of Chip Design
Indian Institute Of Technology–Palakkad
152202017@smail.iitpkd.ac.in

***Abstract*—The Embedded System Design with NIOS II Soft-Core Processor is the foundation of this Workshop. it helps us to understand how both hard-core and soft-core designs flow together and are evaluated on the DE2-115 remote laboratory setup at Intel Labs.**

I. INTRODUCTION

Hardware and software combination designed for a specific function is known embedded system. it may operate as part of a bigger system. it may be programmable or not have a fixed functionality.

SOC FPGA devices are powerful embedded computing platforms because they combine CPU and FPGA architecture into a single unit.

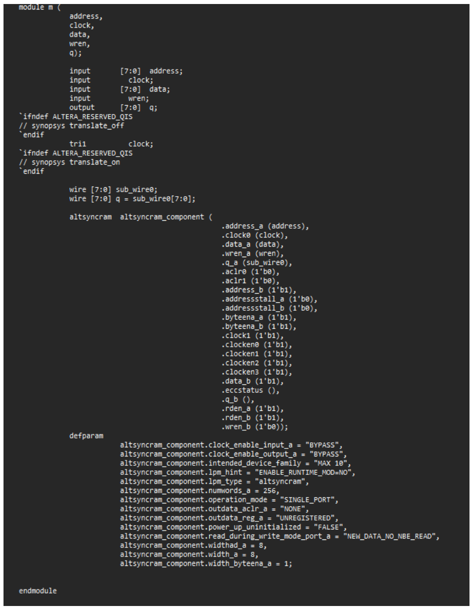
II. DESIGN RAM USING INTELLECTUAL PROPERTY

Since IPs are predesigned, pretested, and pre-verified, we don't need to design verilog code from start. Instead, we may use pre-made functions and libraries in our designs because they have already been tested and verified. we can directly use it in our design.

1) Create a new project in the first step with Device set to MAX10 (10M50DAF484C7G). At this point, we can find the 1 port RAM in the IP catalogue.
2) The width and depth of a memory, the clocking techniques, and the ports we can choose are crucial factors that must be needed.
3) There must be a read and write conflict while using a single port RAM. To avoid the issue of allocating q output, three options are available: fresh data, old data, and don't care and then use the .mif file or blank to initialise the memory content
4) By completing all of the steps, verilog code for the RAM memory shown in figure 1 is generated, as is the RTL viewer in figure 2

5) When we build the design, a resource summary is generated, as shown in figure 3, and when we check the chip planner, we get an image, as shown in figure 4, that informs us what memory we used, which is highlighted in blue.

Now, in the preceding technique, we left blank at the time of memory initialization; now, we create a mif file and provide it at the time of memory initialization.



```
module m (
            address,
            clock,
            data,
            wren,
            q);

            input       [7:0]  address;
            input         clock;
            input       [7:0]  data;
            input         wren;
            output      [7:0]  q;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_off
`endif
            tri1          clock;
`ifndef ALTERA_RESERVED_QIS
// synopsys translate_on
`endif

            wire [7:0] sub_wire0;
            wire [7:0] q = sub_wire0[7:0];

            altsyncram  altsyncram_component (
                                    .address_a (address),
                                    .clock0 (clock),
                                    .data_a (data),
                                    .wren_a (wren),
                                    .q_a (sub_wire0),
                                    .aclr0 (1'b0),
                                    .aclr1 (1'b0),
                                    .address_b (1'b1),
                                    .addressstall_a (1'b0),
                                    .addressstall_b (1'b0),
                                    .byteena_a (1'b1),
                                    .byteena_b (1'b1),
                                    .clock1 (1'b1),
                                    .clocken0 (1'b1),
                                    .clocken1 (1'b1),
                                    .clocken2 (1'b1),
                                    .clocken3 (1'b1),
                                    .data_b (1'b1),
                                    .eccstatus (),
                                    .q_b (),
                                    .rden_a (1'b1),
                                    .rden_b (1'b1),
                                    .wren_b (1'b0));
            defparam
                        altsyncram_component.clock_enable_input_a = "BYPASS",
                        altsyncram_component.clock_enable_output_a = "BYPASS",
                        altsyncram_component.intended_device_family = "MAX 10",
                        altsyncram_component.lpm_hint = "ENABLE_RUNTIME_MOD=NO",
                        altsyncram_component.lpm_type = "altsyncram",
                        altsyncram_component.numwords_a = 256,
                        altsyncram_component.operation_mode = "SINGLE_PORT",
                        altsyncram_component.outdata_aclr_a = "NONE",
                        altsyncram_component.outdata_reg_a = "UNREGISTERED",
                        altsyncram_component.power_up_uninitialized = "FALSE",
                        altsyncram_component.read_during_write_mode_port_a = "NEW_DATA_NO_NBE_READ",
                        altsyncram_component.widthad_a = 8,
                        altsyncram_component.width_a = 8,
                        altsyncram_component.width_byteena_a = 1;


endmodule
```

Fig. 1. verilog code

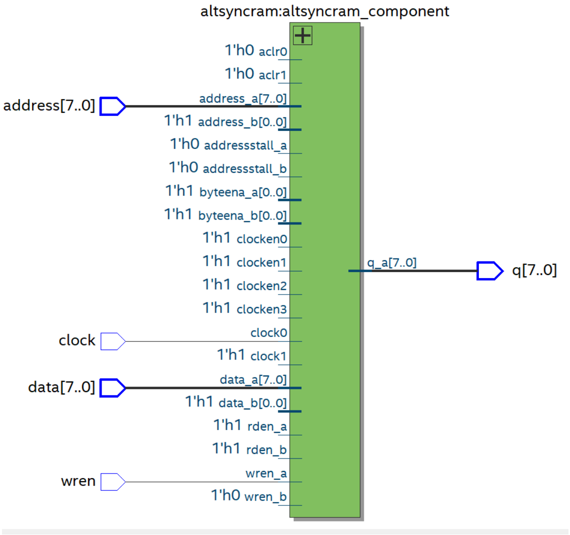
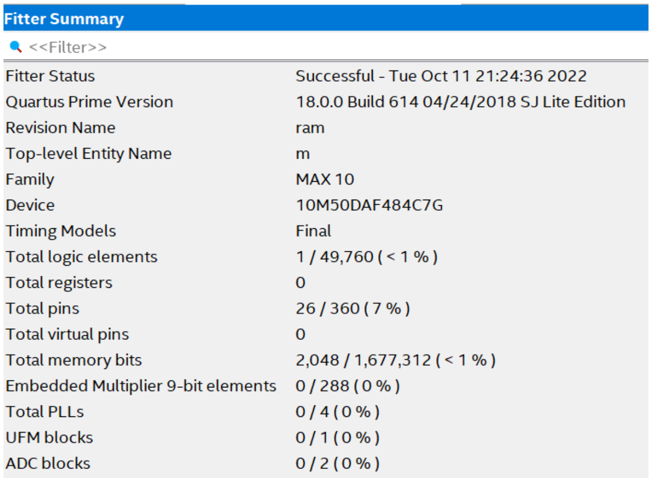


Fig. 2. RTL View

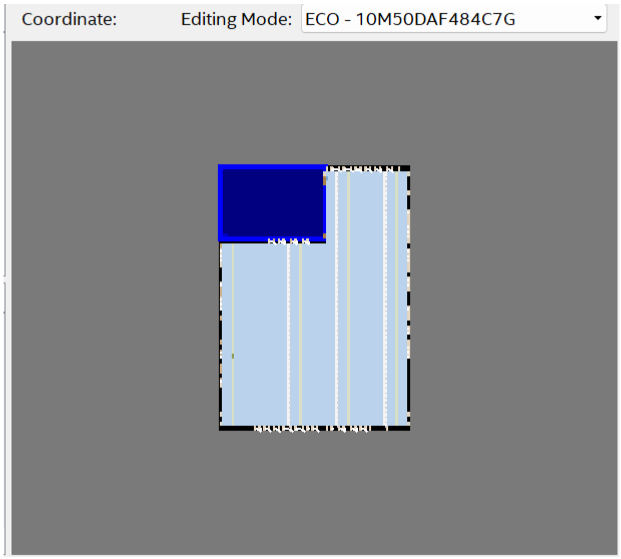


Fitter Summary

<<Filter>>

Fitter Status	Successful - Tue Oct 11 21:24:36 2022
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	ram
Top-level Entity Name	m
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	1 / 49,760 (< 1 %)
Total registers	0
Total pins	26 / 360 (7 %)
Total virtual pins	0
Total memory bits	2,048 / 1,677,312 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Fig. 3. Resource Usage Summary



Coordinate: Editing Mode: ECO - 10M50DAF484C7G

Fig. 4. Chip Planner

A. *Creation of .mif file*

1) create a new .mif file, By selecting file and then new,mif file.
2) first we choose the quantity and size of each word, then right-click any cell and choose ”custom fill cells.”
3) we give start address and ending address and value to start with , and whether we want the value increase or decrease.
4) finaly we save the file and use that file in the design process

III. System Design Entry Using Platform Designer

Consider a system with a master and slave devices. The master must communicate with each slave separately. Since the connection between the master and slave computers must be error-free, manual connection is complicated.it creates connection automatically.

A. *Advantage*

i) it simplifies complex development
ii) it raise level of design abstraction
iii) it enable Design reuse
iv) it reduce time to market
v) it provide standard platform,perform ip integration, custom ip authoring, ip verification

IV. NIOS II Design Flow

The NIOS II is a soft core 32-bit embedded RISCV processor that works with all Intel fpga devices.it is use the resourse of the FPGA. it is availble in two variants.

1) fast variant use for speed optimization
2) Economy variant used for area optimization

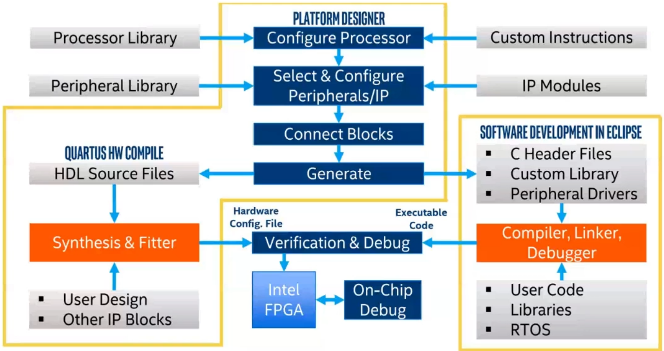


Fig. 5. Design Flow

There are two types of processors: soft core and hard core. An embedded system combines hardware and software. Hardware and software design are therefore included in the design.

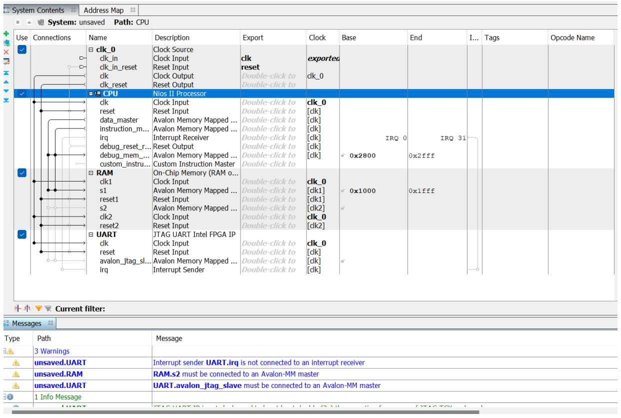
A. *Steps:*

1) choose all of the system’s components in the Platform Designer, then configure the chosen blocks and link them. Platform designers ensure that each block is connected correctly before producing .sopcinfo files.
2) The HDL file is supplied to Quartus Prime as input for hardware design. Eclipse, a software design tool included in Intel Quartus Prime, receives .sopcinfo as input.
3) Eclipse creates a file called .elf, often known as an executable file, when C programming is used.
4) A file called .sof file is produced by the Quartus Prime software.
5) .elf and .sof these two files givent to FPGA

V. DESIGN USING PLATFORM DESIGNER

A. Steps:

1) Select the family device to start a new project (max10,cyclone 5 etc).
2) By selecting Tools ¿ Platform Designer, launch a platform designer window, then add the NIOS II embedded processor.
3) select the variants
   a) \ f (fast variant use for speed optimization)
   b) \ e (Economy variant used for area optimization)
4) Therefore, they are automatically connected in platform designer. Simply click the dot at the connection to activate it.
5) Add the on-chip memory from the IP catalogue, initialise the memory content with the .mif file, and add jtag. Remove the error mentioned below and generate the compile file after removing the error.



System Contents | Address Map

System: unsaved Path: CPU

Use	Name	Description	Export	Clock	Base	End	I...	Tags	Opcode Name
☑	⊟ clk_0	Clock Source							
	clk_in	Clock Input	**clk**	*exported*					
	clk_in_reset	Reset Input	**reset**						
	clk	Clock Output	*Double-click to*	clk_0					
	clk_reset	Reset Output	*Double-click to*						
☑	⊟ **CPU**	Nios II Processor							
	clk	Clock Input	*Double-click to*	**clk_0**					
	reset	Reset Input	*Double-click to*	[clk]					
	data_master	Avalon Memory Mapped ...	*Double-click to*	[clk]					
	instruction_m...	Avalon Memory Mapped ...	*Double-click to*	[clk]					
	irq	Interrupt Receiver	*Double-click to*	[clk]	IRQ 0	IRQ 31			
	debug_reset_r...	Reset Output	*Double-click to*	[clk]					
	debug_mem_...	Avalon Memory Mapped ...	*Double-click to*	[clk]	0x2800	0x2fff			
	custom_instru...	Custom Instruction Master	*Double-click to*						
☑	⊟ **RAM**	On-Chip Memory (RAM o...							
	clk1	Clock Input	*Double-click to*	**clk_0**					
	s1	Avalon Memory Mapped ...	*Double-click to*	[clk1]	0x1000	0x1fff			
	reset1	Reset Input	*Double-click to*	[clk1]					
	s2	Avalon Memory Mapped ...	*Double-click to*	[clk2]					
	clk2	Clock Input	*Double-click to*	**clk_0**					
	reset2	Reset Input	*Double-click to*	[clk2]					
☑	⊟ **UART**	JTAG UART Intel FPGA IP							
	clk	Clock Input	*Double-click to*	**clk_0**					
	reset	Reset Input	*Double-click to*	[clk]					
	avalon_jtag_sl...	Avalon Memory Mapped ...	*Double-click to*	[clk]					
	irq	Interrupt Sender	*Double-click to*	[clk]					

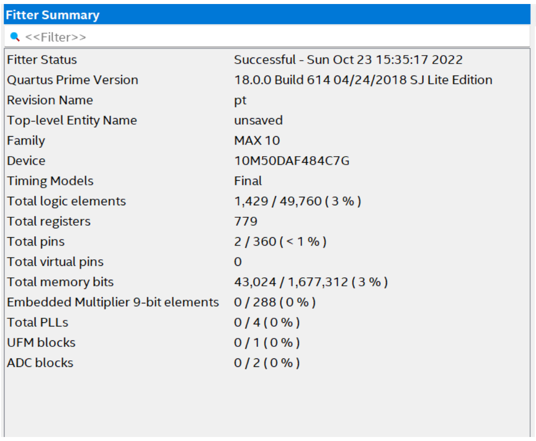
Current filter:

Messages

Type	Path	Message
⚠	3 Warnings	
⚠	**unsaved.UART**	Interrupt sender **UART.irq** is not connected to an interrupt receiver
⚠	**unsaved.RAM**	**RAM.s2** must be connected to an Avalon-MM master
⚠	**unsaved.UART**	**UART.avalon_jtag_slave** must be connected to an Avalon-MM master
ⓘ	1 Info Message	

Fig. 6. Platform Designer

6) Add .qsys file to the project and compile
7) use economy variant and it take more resource and here shows the Resource Summary in the figure 7.



Fitter Summary

<<Filter>>

Fitter Status	Successful - Sun Oct 23 15:35:17 2022
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	pt
Top-level Entity Name	unsaved
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	1,429 / 49,760 (3 %)
Total registers	779
Total pins	2 / 360 (< 1 %)
Total virtual pins	0
Total memory bits	43,024 / 1,677,312 (3 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Fig. 7. Resource Summary

8) The FPGA hardware development process is complete. Now, for the software process, open tools, pick NIOS II software, and go to file,new,NIOS II application, and add the previously produced sopc file.
9) open the design file copy and paste the helloworld.c code in system
10) By using the labsland remote access upload .sof file and .elf file choose verify and upload,.sof is produced.

VI. INTEL SOC DESIGN

the previous design which we discuss it is called system on board because we are adding the processor.SOC design mean System on Chip design. it has both processor and the FPGA architecture into a single chip.
ARM Processor + FPGA = SOC FPGA

A. Application of SOC:

1) we use in cell phone because it can operate at low frequency
2) we can use in network (enxample- 5G network)
3) used in television and set-top box etc.

B. Advantage of SOC:

1) it reduce cost
2) it reduce complexity of the design
3) it reduce power
4) it reduce external device because FPGA is Programmable

C. Design Flow

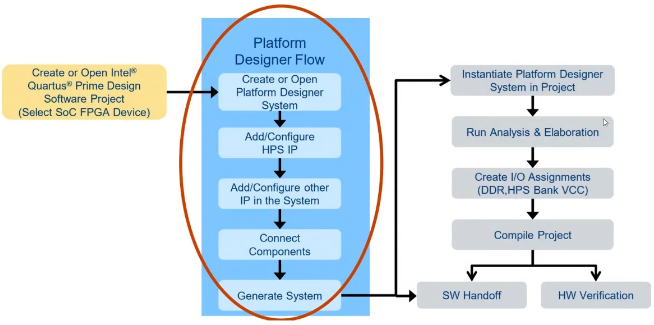

Create or Open Intel® Quartus® Prime Design Software Project (Select SoC FPGA Device)
Platform Designer Flow
Create or Open Platform Designer System
Add/Configure HPS IP
Add/Configure other IP in the System
Connect Components
Generate System
Instantiate Platform Designer System in Project
Run Analysis & Elaboration
Create I/O Assignments (DDR,HPS Bank VCC)
Compile Project
SW Handoff
HW Verification

Fig. 8. Design Flow

1) Select the SOC FPGA Device to start the project.
2) Configure the HPS IP and connect the component.
3) instantiate the platform designer by using Quartus prime software.
4) Make I/0 assignments using the same process analysis and elaboration.
5) compile the project.