Implementation of 4 bit array multiplier using Verilog HDL and its testing on the Spartan 2 FPGA

The aim here is to take you through the design and implementation steps of FPGA implementation for 4-bit binary multiplier. The algorithm used here is a simple one that uses repeated addition. Refer to HDL description for Adder and Full Adder given in the text Digital Principles and Applications, 6e by Leach, Malvino and Saha, TMH, 2006.

The gate level diagram of the 4 bit array multiplier was obtained as follows : -(The unconnected inputs are the combinations of the input bits ANDed in pairs. The exact combinations at each adder box can be found out from the Verilog code)



From the diagram, the gate level verilog code is then written in the Xilinx text editor :

output sout, cout; input a,b; assign sout=a^b; assign cout=(a&b); endmodule module FA(sout,cout,a,b,cin); output sout,cout; input a,b,cin; assign sout=(a^b^cin); assign cout=((a&b)l(a&cin)l(b&cin)); endmodule module multiply4bits(product,inp1,inp2); output [7:0]product; input [3:0]inp1; input [3:0]inp2; assign product[0]=(inp1[0]&inp2[0]); wire x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17; HA HA1(product[1],x1,(inp1[1]&inp2[0]),(inp1[0]&inp2[1])); FA FA1(x2,x3,inp1[1]&inp2[1],(inp1[0]&inp2[2]),x1); FA FA2(x4,x5,(inp1[1]&inp2[2]),(inp1[0]&inp2[3]),x3); HA HA2(x6,x7,(inp1[1]&inp2[3]),x5); HA HA3(product[2],x15,x2,(inp1[2]&inp2[0])); FA FA5(x14,x16,x4,(inp1[2]&inp2[1]),x15); FA FA4(x13,x17,x6,(inp1[2]&inp2[2]),x16); FA FA3(x9,x8,x7,(inp1[2]&inp2[3]),x17); HA HA4(product[3],x12,x14,(inp1[3]&inp2[0])); FA FA8(product[4],x11,x13,(inp1[3]&inp2[1]),x12); FA FA7(product[5],x10,x9,(inp1[3]&inp2[2]),x11); FA FA6(product[6],product[7],x8,(inp1[3]&inp2[3]),x10); endmodule

module HA(sout,cout,a,b);

🔀 Xilinx - Project Navigator - D:\array_multiplier\array_multiplier.ise - [multiply4bits]	_ 8 ×
File Edit View Project Source Process Simulation Window Help	» _ B ×
Sources in Project 30 module FA (sout, cout, a, b, cin);	
□ □ aray_multiple.se 31 output sout,cout; □ □ □ □ □	
□ □ multiply4bits.v) 32 input a,b,cin;	
Findiputabis.v) 34 assign coute ((asb))(ascin) / (b&cin));	
HA (multiply4bits.v) 35 endmodule	
36 37 wedule wultipluthits(weduct inv1.ipv2);	
<pre>38 output [7:0] product;</pre>	
39 input [3:0] inp1;	
Module Chibrary	

Processes for Source: "multiplydbits"	111
Add Existing Source 44 FA FA1(x2,x3,inp1[1]sinp2[1],(inp1[0]sinp2[2]),x1);	
Create New Source	
Wiew Design Summary 47 HA HA3 (product[2],x15,x2, (inp1[2] & inp2[0]));	100
Design formation 48 FA FA5 (x14,x16,x4, (inp1[2] sinp2[1]),x15);	
Launch ModelSim Simulate 49 FA FA4 (x13,x17,x6, (inp1(2)sinp2(2)),x16);	
Wew Command Line Log F 51 HA HA4 (product[3],x12,x14, (inpl[5]slipp2[0]));	-
View HDL Instantiation 1 er	
Process View multiply4bits	
	-
Compiling verilog file "multiply4bits.v"	
Compiling verilog file "multiply4bits.v"	
E Console 🦛 Find in Files 🗶 Errors 🔮 Warnings	_
erarchy is up to date. In 51 Col 1 Verilog	H I
🏦 Start 📔 🚮 🥭 🗊 🕘 💷 Local Disk (D:) 🛛 🔯 Xilinx - Proje 🖸 array_multiplier 🛛 🖓 multiplier 🛛 🖓 Starten 2 - Paint 🧔	9:23 AM

Next the test bench file has to be created to test the design. The test bench file can be created by right clicking on the main module name and selecting the "Add New Source" option.

Xilinx - Project Navigator - D:\array_multiplier\ar	ray_multiplier.ise - [multiply4bits]			
				1
			4	
Sources in Project:	3			
array_multiplier.ise	module FA(sout,cout,a,b,c output sout.cout:	in);		
B- C xc2s200-5pq208	2 input a,b,cin;			
FA (multiply4bits.v)	New Source	×		
HA (multiply4bits.v) 3				
3	BMM File	-		
3	Implementation Constraints File			
	MEM File	File Name:		
Module In Snapsh In Library 4	Schematic	bench		
4	State Diagram	Location:	x15,x16,x17;	
Processes for Source: "multiply4bits"	User Document	D:\array multiplier	:[1]));	
Add Existing Source	Verilog Module		;	
∑ View Design Summary 4	Verilog Test Fixture			
E Design Utilities	VHDL Module		-	-
Lieate Schematic Symbol Launch ModelSim Simulatc	VHDL Package			
View Command Line Log F	VHDL Test Bench			
View HDL Instantiation Ter		Add to project	Þ	-
				_
	< Back Next	> Cancel Help		
×				
TT I				- 1
Compiling verilog file "multiply	y4bits.v"			
Compiling verilog file "multiply	y4bits.v"			- 1
				- 1
				- 1
E Canada Mar Find in Elea Y Fuero II Vien				
Ready				
📴 Start 📋 🙆 🐉 🤭 📋 🖃 Local Disk (D:)	Xilinx - Proje 🔄 array_multiplier	multiplier Alscreen1 - Paint	🛛 🖓 🔁 🕮 ঝ 🌮 9:23 A	M

Xilinx - Project Navigator - D:\array_r	multiplier\array_multiplier.ise - [bench.tbw]			
			• •	
	ns 🔻			
	Initialize Timing		×	[
Sources in Project:				
array_multiplet ise array_multiply4bits.v) multiply4bits (multiply4bits.v) FA (multiply4bits.v) HA (multiply4bits.v)	Assign Inputs Wait To Check	Check Assign Dutputs Inputs → Wait To Assign		
Contraction of the second seco	Clock Timing Information Inputs are assigned at "Input Setup Time" and outputs are designed at "Unput Valid Delay". C Rising Edge C Faling Edge C Dual Edge (DOR or DET) Clock Time High 100 ns Clock Time Low 100 ns Input Setup Time IS ns Output Valid Delay IS ns Initial Offset 0 ns	Clock Information Single Clock Conductorial Clock Combinatorial (or internal clock) Combinatorial Timing Information Inputs are assigned, outputs are decoded the deteded. A delay between inputs and outputs assignment/checking conflicts. Check Outputs 50 ns After Inputs ar Assign Inputs 50 ns After Outputs 4	en s avoids re Assigned are Checked	Ĩ
Process View Hierarchy - bench.tt	Global Signals	Initial Length of Test Bench: 1000 r	ns	
× ×	High for Initial: 100 ns	Time Scale: ns		
Compiling verilog file Compiling verilog file Compiling verilog file	OK	Cancel Next >	Help	
Compiling verilog file	"d:\array_multiplier\multiply4bi	ts.v"		
Console 🖓 Find in Files 🗶 Erro	ors 🛂 Warnings			
🏦 Start 📗 🚮 🅭 🎲 🧶 🗍 🖃 L	.ocal Disk (D:) 🛛 🙀 Xilinx - Proje 🔄 array_mu	ltiplier Screen2 - Pain	ıt	🍕: 📘 🕮 祠 🌮 9:24 AM

Once the test bench file is created, it is time to apply the inputs in the input box. The inputs can be specified both in binary number format as well as by specifying their waveform. Let for example, the input to given as inp1=14 and inp2=13:

Xilinx - Project Navigator - D:\ar	ray_i	multiplier\array_mu	Itiplier.ise ·	[bench.tt	w]					_ 8 ×
File Edit View Project Source	Proc	ess Simulation Test	Bench Win	dow Help				-		» _ ð ×
	1		≁ №?	る国	副のの構	A		<u>- 8</u>		
II 9≡ ► ► ⁸ 1000	•	15 💌								
(I)	×									ſ
Sources in Project:		INNA	ચિવ.	8						
E c2s200-5pq208		End Time:								
multiply4bits (multiply4bits.v		1000 ns		50 ns	150 ns 250 n	s 350 ns	450 ns 5	50 ns 650 ns	750 ns 850	ns 950 ns
HA (multiply4bits.v)	Ш	E ax product[7:0]	0	$\overline{}$			0			
	Ш	🗉 📈 inp1(3:0)	14				14			
	Ш	🗉 💦 inp2[3:0]	13				13			
📑 📽 Mod 💼 Sna 🖺 Libr	-									
	×									
or Source: "multiply4	1									
Design Utilities										
Launch ModelSim Simulator										
View Command Line Log File										
View HDL Instantiation Temp User Constraints										
Create Timing Constraints										
Assign Package Pins			FEL F	4						
	- -	multinly4hits	bench thu							
Heraichy •			borrorr.com							
×										
T										
🗐 Console 🙀 Find in Files 🕽	C Erro	rs 🛛 🖁 Warnings								
Ready										
🏽 🔀 Start 🛛 🚮 🍮 😒 🙆	0	ocal Disk (D:)	🔷 Xilinx - P	roject N	multiplier	Screene	6 - Paint	1	📢 🗖 🗊	⁄ 🔊 9:32 AM



Its time now to simulate the design. To do this the testbench file is saved and from the processes menu in the left the "Simulate Behavioral HDL" option is selected. The code is thus simulated in ModelSim :

Modelsim XF III/Starter 5.0a - Lustom Xilinx Version		<u></u>
Objects +1 df X Name Value Name Value	999600	
	La companya di seconda di	
I ranscript	Workspace Design unit	Design unit tune Visibili
	FHI bench bench	Module +acc=
# roplever indudies: # multiply4bits	- CHECK_product bench	Task +acc=
# Model Technology ModelSim XE III vlog 6.0a Compiler 2004.11 Nov 10 2004 # Compiling module bench	the the second	: Module +acc=
#	HIMPLICIT WIRE (inp bench	Process
# Lop level modules:	#INITIAL#32 bench	Process
# Model Technology ModelSim XE III vlog 6.0a Compiler 2004.11 Nov 10 2004	HINITIAL#34 bench	Process
# ~ Compiling module glbl	HINITIAL#44 bench	Process
# Top level modules:	🖃 🗐 🗐 🖂 🗐 🔄	Module +acc=
l# gbl # wim Juiinwooralih warJuminima warJihuwak J1na banah albi		Process
# Vain 12 Xin Xoleid_Ver 12 Unisins_Ver 10 Work 1 (1): Dench gibi # Loading work.bench		Process
# Loading work.multiply4bits		Process
# Loading work FA		Process
# Loading work.gbl	HINITIAL#44 glbi	Process
# .main_pane.mdi.interior.cs.vm.paneset.cli_0.wt.clip.cs	HINITIAL#52 gibi	Process
#.main_pane.signals.interior.cs		
# No errors or warnings. # Reads at baseds the line 11		
# Simulation Breakpoint: Break at bench.tfw line 41		
# MACR0 ./bench.fdo PAUSED at line 14		<u> </u>
VSIM(paused)>	Library 🔊 sim 🗟 Files	<u></u>
Now: 1 us Delta: 0 sim:/bench		Ln: 41 Col: 0 READ
🅦 Start 🛛 🚮 🧔 🏐 🕘 🔰 🔤 Local Disk (D:) 🛛 🖗 Xlinx - Project 🖯 🔄 multiplier 🛛 🖓 screen 5 - Paint 🛛 🙀	ModelSim XE	🔰 🕮 ⁄ 🎝 🦘 9:47 AM

The "wave " window is then maximized since that is where the output waveform will be coming. Here we see the product = 10110110 in binary = 182 in decimal.



It's time now to download the design on the Spartan 2 FPGA.

To do so first the "Implementation Constraint File " (.ucf) file is created by adding a "New Source " from the main module.



Xilinx Pace opens with a top view of the FPGA package on the right and input boxes for specifying the input and output ports on the left. The pin numbers of the input and the output are obtained from the manual accompanying the device, and entered in the input boxes in the desired order.

À Xilinx	- Project Navigator - D:\array_multiplier\array_multiplier.ise - [m	ultiply4bits]	_ 8 ×
File	Xilinx PACE - D:\array_multiplier\mult.ucf		× - B×
0 🗳	File Edit View IOBs Areas Tools Window Help		
Per In		☞ 🔝 🔏 🛅 🖩 🖷 🖾 🗆 🛠 q q 💥 q 😰 🛛 🗖 🗖 🗖	
Mar II	B Design Browser	Package Pins for xc2s200-5-pg208	×1
Source	S Global Logic		- ∥ - ∣
	🚛 🔁 Logic		
-			
1			
- ■C M			
	🕻 Design Object List - I/O Pins		
	I/O Name I/O Direction Loc Bank I/O Std. Vr 🔺		
Proces	inp2<2> Input P61 BANK		
pC	product<0> Output P89 BANK		
	product<1> Output P100 BANK		
	product<2> Output P87 BANK	Top View	
	product<3> Uutput P97 BANK	8	
	product<5> Output P98 BANK		
	product<6> Output P84 BANK		
	product<7> OutputP95▼ BANK		
_ ■ C P	G # Group 1/O Direction Loc 1/O Std. Vref Vc		
×	8 product Output		
	4 inp2 input		
τ	7:		
III T	r.		
	1		
l l			
		Deckage View (Architecture View (그는 그는
Hierarchy			
Star	: 📋 🍮 🐉 🤭 🔄 🔄 Local Disk (D:) 🛛 🚺 Xilinx - Project	Nav I 🔄 Multiplier 🍸 Screen 9 - Paint 🛛 🗱 Xilinx PACE - D:\ 🛛 🍕 🕩 👰	🕂 9:53 AM

The next step in the download process is configuring the device according to our specifications . To do this the "Configure Device (IMPACT)" option is double clicked in the "Process" box on the left.



Xilinx then starts configuring the device according to the demands of our program. The progress of the process in shown in the bottom right as percentages.

The different modes of operation include the "Slave Serial mode " and the "JTAG" (Joint Test Action Group) mode. Here the Slave Serial Mode is used ,and the check box for the Slave Serial Mode is selected when asked the mode has to be specified.

The bit file of our design is selected when the necessary input box comes up :

🐈 screen12 - Paint	_ 8 ×
File Edit View Image Colors Help	
Xilinx - Project Navigator - D:\array_multiplier\array_multiplier.ise - [multiply4bits]	
2 🚱	
💫 A 🔤 🔤 d:\array_multiplier/array_multiplier.ipf [Configuration Mode] - iMPACT	
Control Control Control Control Continuation	
Cook in: 🔄 array_multiplier 🔽 🗭 🖽 🕶	
Desktop	
r Source: "multiply4.	
Gener. // ★ My Documents	
Place & Rout // * My Computer	
Programming CB_P;	
Generate PR Conni 💭 File name: multiply4bits 🔽 Open	
Contrigure De Co	
Driver windivio.sys version = 0.2.2.2. Lit base address = 0.70n.	
ECP base address = FFFFFFh. Cable connection established.	
NGDBUILD CB_PROGRESS_END - End Operation.	
Number For Help, press F1 Configuration Mode Slave Serial	
Writing NCD file "multiplu/bits ngd"	
	<u> </u>
For Help, click Help Topics on the Help Menu.	
🕅 Start 🛛 🖉 🕘 🔰 💷 Local Disk (D:) 🛛 🖗 Xilinx - Project Nav 🔄 multiplier 🛛 🖄 screen 12 - Paint 🛛 🐺 dt \array_multipl 🛛 👯 🖓 🖗 🧐	:56 AM

The device is configured and a top view of the device comes up.

Xilinx - Project Navigator - E	Dr\array_multiplier\array_multiplier.ise - [multiply4bits] urce Process Smulation Window Help	_ 8 ×
		1
Sources in Project	17 // Revision 0.01 - File Created	_
- e array_multiplier.ise	d\array_multiplier/array_multiplier.ipf [Configuration Mode] - iMPACT	
☐ xc2s200-5pq208	File Edit View Mode Operations Output Help	
- D bench (bench th		
- 🚺 mult.ucf	Boundary-Scan Slave Serial SelectMAP Desktop Configuration	
- FA (multiply4bits.	Right click device to select operations	
I In (nonpy-toro.		
	20000g	
📑 📽 Mod 💼 Sna 🖺 Li		
L Courses Productor	xc2s200	
r Source: multiply4	multiply4bits.bit	
Manually Place & B	Dout	
E Generate Post-Map		
🖓 🖌 Place & Route	Please note: Configuration operations are executed	
Generate Programming File	only on all devices as a group.	
Generate PROM. ACE. or		
Configure Device (iMPACT		
		<u> </u>
🗖 📲 Process 🔛 Hierarchy	GUI Desktop Configuration selected	
	// *** BATCH CMD : setMode -dtconfig GUI SelectMAP Mode selected	
Total time: 1 se	// *** BATCH CMD : setMode -sm	
	GUI Slave Serial Mode selected	
	For Help, press F1 Configuration Mode Slave Serial Parallel III lpt1 : //	
Started process	"Generate Programming File".	
		<u> </u>
📗 📃 Console 🏧 Find in File	es 🗶 Errors 🦞 Warnings	
Process "Configure Device (iMPACT)	" is up to date. In 54 Col 2 Verilog	
🔀 Start 🛛 🚮 🥭 😂 🤭	📄 🔄 🖾 🕹 🔤 🔤 🔤 🔤 🔤 🔤 🔤 🔤 🔤 🔤	5 9:56 AM

On left clicking on the picture of the device, the last step of the programming is completed.



Now the inputs are to be made available to 4+4 = 8 pins of the FPGA configured as input in previous step and output will be available at 8 pins configured as output. The FPGA unit now behaves like a 4-bit multiplier.