CS227- Lab 10

Digital design using HDL(Verilog)

The goal of this lab is to get familiar with digital modeling in Verilog Hardware Description Language (HDL) and to learn how to handle the simulator. The goal is also to interpret outputs from a simulator and to understand how Verilog code is interpreted by the simulator.

For this purpose we will use **modelsim** simulator.

Modelsim Installation link for Windows:

https://www.intel.com/content/www/us/en/software-kit/660907/intel-quartus-prime-lite-edition-design-software-version-20-1-1-for-windows.html

Modelsim Installation link for Linux:

https://www.intel.com/content/www/us/en/software-kit/661017/intel-quartus-prime-lite-edition-design-software-version-20-1-for-linux.html

In your local system, create a lab directory for the course (say. D:/CS227/Verilog). Download the Lab10 files .

Start modelsim by the following steps

Start -> programes->modelsim->

Create a project (File->New->Project)

Name the project and project location as shown in the following figure.

ModelSim - INTEL FPGA START	ER EDITION 2020.1	
File Edit View Compile Sime	ulate Add Library Tools Layout Bookmarks Window Help	
🖹 • 🚔 🖬 🤹 🍈 🐰 🕅 (🏝 之 🗁 🖉 - 桷 計 📆 🕸 🖄 🎬 💭 🔣 🍷 🏤 🏌 🍰 - 🏠 - 🏦 Layout NoDesign 📃	
ColumnLayout AllColumns		
Library		+ # ×
▼ Name Type	Path	<u> </u>
work (empty) Library	D:/Drive2/courses/DigitalSystem/Verilo	
+ 1 220model Library	\$MODEL_TECH//altera/vhdl/220model	
+ 220model_ver Library	\$MODEL_TECH//altera/verilog/220m	
	\$MODEL_TECH//altera/vhdl/altera	1
	\$MODEL_TECH//altera/vhdl/altera_l	
ditera_Insim_ver Library	\$MODEL_TECH//altera/verilog/altera	
	\$MODEL_TECH//altera/vhdl/altera_mf	
	\$MODEL_TECH//altera/verilog/altera	
	\$MODEL_TECH//altera/verilog/altera	
⊕ arriaii Library	\$MODEL_TECH//altera/vhdl/{	
⊕ arriaii_hssi Library	\$MODEL_TECH//altera/vhdl/{ M Create Project X	
	\$MODEL_TECH//altera/verilo Project Name	
	\$MODEL_TECH//altera/vhdl/	
	\$MODEL_TECH//altera/verilo	1
	\$MODEL_TECH//altera/verilo	
	\$MODEL_TECH//altera/vhdl/	
	\$MODEL_TECH//altera/vhdl/ urses/DigitalSystem/Verilog/Labio Browse	
⊕ arriaiigz_hssi_ver Library	\$MODEL_TECH//altera/verilo	
	\$MODEL_TECH//altera/vhdl/	
Library arriaiigz_pcie_hip_v Library	\$MODEL_TECH//altera/verilo	
	\$MODEL_TECH//altera/verilo	
⊕ arriav Library	\$MODEL_TECH//altera/vhdl/ Copy Settings From	
	\$MODEL_TECH//altera/verilo modelsim_ase/modelsim.ini Browse	
	\$MODEL_TECH//altera/verilo 📀 Copy Library Mappings 💭 Reference Library Mappings	
	\$MODEL_TECH//altera/verilo	
	\$MODEL_TECH//altera/vhdl/	
	\$MODEL_TECH//altera/vhdl/	
	\$MODEL_TECH//altera/verilog/arriav	
arriavgz_pcie_hip Library	\$MODEL_TECH//altera/vhdl/arriavgz	-

Figure 1: Mdelsim: create project option

Then appropriate option to be selected for adding files in the project. The options are shown as below.

File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help
Image: Statut Type OrderModified
ColumnLayout [AllColumns] Image: Project - D:/Drive2/courses/DigitalSystem/Verlog/Lab 10/CS227 Image: Project - D:/Drive2/courses/DigitalSystem/Verlog/Lab 10/CS227
Image: Project - D:/Drive2/courses/DigitalSystem/Ver/log/Lab 10/CS227
▼Name Statu{Type Orde[Modified
Add items to the Project ×
Click on the icon to add items of that type:
* 1 1
Create New File Add Existing File
Create Simulation Create New Folder
Cince
🕼 Library 🛛 🕮 Project 🗵

Figure 2: Modelsim: add items to the project option

After adding files (say a AND design file and the corresponding testbench file), they will be shown as below.



Figure 3: Modelsim: after adding the files

After adding the files, to show the contents, you need to double click on the file name. The corresponding file will then be available for editing purpose. Double clicking the design file will open it in a separate window as shown below.



Figure 4: Modelsim: Displaying the design file

In a similar way, the test bench file will be available in a separate window.

ModelSim - INTEL FPGA STARTER EDITION 2020.1		- u ×
<u>File Edit View Compile Simulate Add Source Tools Layout Bookn</u>	narks <u>Window</u> <u>Help</u>	
] 8 • ☞ 등 중 卷 1 등 등 요그 ○ • ♣ 등 兩 ♦ 韓 @	🗱 🕅 🕈 🏤 🛊 🍰 - 🏤 - 🏦 Layout NoDesign 🚽 ColumnLayout AllColumns 💌	
🖄 Project - D:/Drive2/courses/DigitalSystem/Verilog/Lab10/CS227 🚃 📰 🛨 🛃 🗴	D:/Drive2/courses/DigitalSystem/Verlog/Lab10/tb_and_gate.v - Default	+ d ×
Name Statut Type Orde Modified	Ln#	
	<pre>timestain 1 Ar/1 is 5 timestain 1 Ar/1</pre>	
4		>
Library × 🖽 Project ×	b_and_gate.v × b_and_gate.v ×	<[]
Q Transcript		: + # ×
<pre># Compile of and_gate.v was successful. # Compile of tb_and_gate.v was successful. ModelSim>iles. 0 failed with no errors.</pre>		

Figure 5: Modelsim: Displaying the testbench file

The codes can be compiled using the compile all option as shown below.



Figure 6: Modelsim: Compile All option

Once compilation is successfully done then you are ready for the simulation. Go to simulation menu and select Start Simulation as shown below.

ModelSim - INTEL FPGA STA	RTER EDITION 2020.1								-	Ð	×
<u>File Edit View Compile</u>	Simulate Add Source	e T <u>o</u> ols Lay	yo <u>u</u> t Boo <u>k</u> ma	arks <u>W</u> ir	ndow <u>H</u> elp						
🖹 • 🚘 🗑 🍮 🚳 🐰	Start Simulation Runtime Options	M 🗄 M	🕸 😰 🖽	2	*****	Layout NoDesign	◄	ColumnLayout AllColumns			
	Run 🕨										
Project - D:/Drive2/courses/D	Step 🕨	:S227::::::		D:/Dr	rive2/courses/DigitalSystem/Verilog/L	ab 10/tb_and_gate.v - Default 💳					+ 2
▼ Name Sta	Restart			Ln#							
and gate.v Break End Simulation 20239:25 D0239:09 1 timescale 1 ns/1 ns timescale 1 ns/1 ns to gate.v big and gate.v End Simulation 2 module tb and gate(); // Test bench fortb_ and gate.v reg a,b; vire out; and gate uut(out,a,b); // this instantiates a and gate, uut is a label initial 0 Begin a = 1'b0; begin a = 1'b0; begin b											
		1									>
Library V Project V			<u>-</u>	bos	nate v Y						
the containty on the project of			4 1 2								1
A Transcript											+ 7
<pre># Compile of tb_and_gate # 2 compiles 0 failed;</pre>	e.v was successful	•									

Figure 7: Modelsim: start simulation

Expand the work folder and select the testbench file for starting the simulation as shown in the following figure.

<pre>dels - INTEL PEGA STARTER EDITION 2020.1</pre>	
┃	A 🛊 📩 - 🚠 - 🚠 Layout NoDesign 🗾 ColumnLayout [AllColumns 💌
Project - D:/Drive2/courses/DigitalSystem/Verlog/Lab10/CS227 d X D:/Drive2/courses/DigitalSystem/Verlog/Lab1	ourses/DigitalSystem/Verlog/Lab10/tb_and_gate.v - Default= # a
and_gate.v	<pre>timescale 1 ms/1 ms module tb_and_gate(); // Test bench fortb_ and_gate.v reg a,b; // a reg, to allow us to assign the input, and a wire to receive the output wire out; and gate uut(out,a,b); // this instantiates a and gate, uut is a label initial begin a = 1'b0; // here we apply inputs to the gate b = 1'b0;</pre>
10 11 11 12 13 14 14 15 16 17 18 19 20 - 21 22 23 24 24 25 26 27	✓ Start Simulation × Design VHDL Verlog Librares SDF Others ✓ Mame Type Path
Library X Mroject X and gate.x dif Trenscript Compile of th_and_gate.v was successful.	Design Unit(s)
2 compiles 0 failed with no errors	OK Cancel

Figure 8: Modelsim: selecting the testench file for starting the simulation

For showing the simulation in wave form, go to view and select the wave option.

_				
M ModelSi	im - INTEL FPGA STARTI	ER EDITI	ION 2020.1	– a ×
<u>Eile E</u> dit	<u>View</u> <u>C</u> ompile <u>S</u> in	nulate	Add Structure Tools Layout Bookmarks Window Help	
🖹 - 🚔	Time Mode (5)		. 🗅 💿 - 桷 監 🗰 🕹 鄧 🕮 梁 函 💁 🛊 🖛 🗰 話 🛛 100 ns 🗄 副 副 副 級 🚳 📓 🗐 🚸 📩 - 🏤 - 🏦 Layout Simulate	-
ColumnLa	Call Stack Capacity	•		
34 - 26	Class Browser Coverage	- - - -		
💭 sim - Def	Dataflow		nve2/courses/DigitalSystem/Verliog/Lab 10/and_gate.v (/tb_and_gate/uut) - Default	
▼ Instance	Files (n)			1 Now 2
🔲 📕 🖬 🕁 🖪	FSM List (q)		// name: and gate.v	
😐 🗾 u	✓ Library (u)		'timescale 1 ns/1 ns	
	List		Dimodele and gate (out, a, b); //you list all inputs and outputs, by convention outputs go list	
	Locals		input a, b;	
🔀 #Vsi	Message Viewer	•		
	Memory List (w)		assign out = a & b; // output function	
	Objects		² endmodule	
	Process			
	Profiling	•		
	 Project (x) 			
	✓ Structure (z)			
	✓ Transcript			
	UVM Details (4)			
	Verification Manager	ment 🕨		
	Watch		-	
	Wave			
	Sort	+		
	Filter	•		
	Justify	+		
	Properties			
•		<	3	>
× 🛗 Proje	ct 🛛 🛺 sim 🛛 🕩 🛛	tba	and gate.v × and gate.v ×	
				اه الد
prig manscrip		0.00		<u> </u>

Figue 9: Modelsim: wave view opening

Once the wave view option is clicked, it will pop up another window as shown below.



Figure 10: Modelsim wave window

To view the objects, go to view object as shown in the following

ModelSim - INTEL FPGA STARTER EDITION 2020.1	als Lavout Book	marks Window Heln						- 0 ×
- - - - - - - - - - -	E 🖬 🛛 🕸 🗱 i	🖱 🖉 🕺 🖕 🛧	🗰 🖦 📑 🗍 10	0 ns 🔶 🗽 🚉 💱 🕷 :	s 💁 💁 🚸 🕴	* 🏕 🕇 🏦 - 🏤 - 🏦	Layout Simulate	_
ColumnLayout Default	-------------	• 🖷 • 🥥 🔰 🕺 👔	10	1/0 🗔 🚛 🍻 📘 🔪	L 4 11 R B	╮ <mark>世上世ॻ</mark> २२		
] 3+ - →€ - 3+ Search: 💌 🏚 🕷	🔅 👘 🛛 🔍 🔍	L # 21 P		J J				
Vave - Default								+ æ
i b_and_gate i b_and_gate	E COL							
Cursor 1	0 ns 0 ns	s 100 ns	200 ns	300 ns 400 ns	500 ns	600 ns 700 ns	800 ns 90	<u>0 ns 100</u>
		<u>الم</u>						•
Project × Sim × • tb_and_gate.v × and_ga	ite.v × 📰 Wave	×						
Transcript # Start time: 06:38:23 on Oct 28.2022								

Figure 11: Modelsim view object

Clicking the view object option will open another window as shown below

M ModelSim - INTEL	PGA STARTER EDITION 2020).1	Profession	Mendeus IIele								-	U /
	impile <u>simulate Ad</u> a (I X Im CL ⊇ ⊇ (⊘ - Ma ि⊑		<u>window ri</u> eip	m m) 17 .	100 ns 🔶 🖹 🖹	i 🕅 🕼 🕴	<u>n</u> 0	† 科 †	≵ • ⋧ • 1	Layout Simul	ate	◄
ColumnLayout Defau	ilt		• 🐴 • 🚱 🐴 • 🛛	3 X 🕅 🛛	1 K	🐐 🔝 🖓	🔯 🎆	🔅 🏩 🛛 🕅	G 🔶 11	III B D	말 두 두 귀 년	t t f	
े दे क र क्र¥ Se	arch:	V (88, 89	🛛 ବ୍ ବ୍ 🙎	13 🖉 🛛 🔳 🔳		· 」」							
🕼 fault 🗤 🕂 🗗 🗙	Objects	Value Med A			Wave - Default								:+ 3
inconce in the and gate in the and ga	Processes (Active) == Processes (Active) ==	Type (filtered)	internal Internal Internal State Order Ready 4 Ready 5	Parent Path /b_and_gate									
					≌≅∙ ⊜∕°≎	Now Cursor 1	0 ns 0 ns 0	20 ns	10 ns	400 ns	600 ns	800 ns	100
• 📕 🕹 sim × 🜗	•				tb_and_gate.v	× and_gate.v :	Vave ×	c					•
A Transcript							<u>, </u>						

Figure 12: Modelsim: objects and wave window view

Now select the input output objects by pressing control key and drag them in the wave window (grey color area)

ModelSim - INTEL FPGA STARTER EDITION 2020.1													٥	×	
<u>Eile E</u> dit <u>V</u> iew <u>C</u> o	mpile <u>S</u> imulate A <u>d</u> e	d W <u>a</u> ve T <u>o</u> ols	Layo <u>u</u> t Boo <u>k</u> marks <u>W</u> i	ndow <u>H</u> elp											
🗟 - 🚅 🖶 🛸 🖨	i 🐰 🖻 🛍 🗅 🗋	🛛 - 🏘 🗄 T	Ā 🛛 🕸 🖾 🖧 🕅	🔁 🕇 🖛 🖬 🖬	100 ns 🚖	i i i	: 🛣 🅼 🎦	💁 🕘 🚦	t 🍽 🕇 I 🛣 -	🏤 = 🏦 📔 Li	ayout Simulate		-		
ColumnLayout Defau	ilt	_	n. n. g n. a	X• >X 🖹 🔏 🛛 🗉	0 10 1	ALL 🧼	R 🖪 🕸	11 🕫 🕫	╮ ૻ ૻ ૻ ૻ :	1 L L F	.J				
] 3+ - +€ - ∰• Se	arch:	.	s 🛛 🔍 🗢 🖉 🖓	. 🭳 🔍 🧏 🍇 📗 🔳 🔳 🔳 🔳 🔳											
🛺 fault 🛲 🛨 🖻 🗙	🔷 Objects 🔅		🛛 🔳 Wave - Default 🚃					- 0000						+ 0	
▼ Instance	▼ Name	1 Now 1	2 💫 -	Msgs											
fundation fundation	b out	x regi.	 (b_and_gate) (b_and_gate) (b_and_gate) 	a x b x out SDX											
			A 📰 👁	Now 0 ns	11111111111111111111111111111111111111	100 ns	200 ns	300 ns	400 ns	500 ns	600 ns	70 r	0 ns		
			<u> </u>	irsor 1 0 ns	0 ns										
• •			1	► I F	<u> </u>										
ect 🛛 🖉 sim 🗡 🕩	•		tb_and_gate.v ×	and_gate.v × 📰 Wave	×									4	
A Transcript													;	+ 🖻 >	

Figure 13: Modelsim: After selecting the objects and dragging them in the wave panel

We can now run the simulation. To run it for 100 ns, we select the appropriate run option in the Simulate menu as shown below.

ModelSim - INTEL FPGA STARTER EDITION 2020.1				– 8 ×
Eile Edit View Compile Simulate Add Trans	cript Tools Layout Bookmarks Window	Help		
Start Simulation Runtime Options	여 한 🖬 🛛 🧇 🛱 🕮 🚑 🕺 🔄 🚹	🦇 🛶 Ef 🛛 100 ns 불 El El El 🕅 🤹	🛅 🖞 🕘 🕇 🏞 🕇 🏦 - 🏤 - 🏦	Layout Simulate
ColumnLayout Default Run	Run 100 8 - 🥵 🖷 - 🥵 🕺 X 🕨	🖹 🌾 📔 T O TO 🗊 🛲 🌽 🛛 🦝 🕯	🛔 🏘 🏟 📗 🔊 🖉 🖾 🕾 📗	*******
Bearch: Step ► Restart	Run -All Run			
🛺 fault :::::: 🛨 🖻 🗶 🏠 Ob. Break	Run -Next	Wave - Default		
Instance Instance Instance Instance End Simulation	e Kind Mode [♥ ■ Now ♥] ► Regi Internal	Msgs		
b x	Regi Internal	/tb_and_gate/a x		
→ #INITIAL#€ → out StX	Net Internal	/tb_and_gate/out StX		
😤 #vsim_capacity				
🎲 Processes (Active)				
▼ Name Typ	e (filtered) State Order Parent Path			
#INITIAL#6 Initi	ial Ready 4 /tb_and_gate ial Ready 5 /tb_and_gate			
		Now 0 ns	ns 200 ns 400 ns	600 ns 800 ns 100
		Cursor 1 0 ns		(
	>	to_ano_gate.v > and_gate.v >	/e ≍	
🟳 Transcript 💷		2000		

sim:/tb_and_gate/a \
sim:/tb_and_gate/b \

Figure 14: Modelsim: run simulation for 100 ns option

Clicking in the wave window and pressing 'f' will fit the wave in the available screen area. The yellow line highlights the time and the corresponding input and output values as shown in the following figure.

ModelSim - INTEL F	PGA STARTER EDITION 202	0.1															- 0	3 ×	
<u>File Edit View Co</u>	mpile <u>S</u> imulate A <u>d</u> d	W <u>a</u> ve T <u>o</u> ols	Layo <u>u</u> t Boo	<u>k</u> marks <u>W</u> ir	ndow <u>H</u> elp														
🖹 • 📽 🖶 🛸 🖷) 🐰 🖻 🛍 🕰 😂	○ - #4 문 7	ज 🛛 🕸 🔯	🏥 💭 🗮	<u>6</u>	🖛 🕪 Ef	100 ns 🔹	e e e	1 🕺 🕵	1 😰 😰	0	‡ 🏊	↑ * •	🔉 - 🏦	Layout Si	nulate	-		
ColumnLayout Defau	ilt	_	% - % - 	} 🐔 - 🤹	X« »X 🛓	0 🐐 📔 I	0 1/0 []]	ALL 🍻	🏟 🎪	🔅 🔅		<u>s</u>	11 11	B L	╘┺╼	1225	æ	-	
3 + - →E - 3}- Se	arch:	.	ې 🔍 🔍	a 78 18	R I II		LT J J							,					
🕼 fault ::::: + 🗗 🗙	술 Objects				-+ & ×	Wave - De	fault											- + 2	×
▼ Instance	* Name	Value Kind	Mode	1년 🔍 1	5 ns গ্ৰ 🕨	<u></u>		1	Msas										ā.
tb_and_gate	🔷 a	0 Regi.	Internal			/tb a	and gate/a	0											ĩ
庄 🧾 uut	🔶 ^b .	1 Regi.	Internal			🔶 /tb_a	and_gate/b	1											٦
#vsim_capacity	🧇 out	St0 Net	Internal			🔷 /tb_a	and_gate/out	St0											
																			1
																			1
																			1
																			1
	Processes (Active)			I	= - - ×														
	* Name	Type (filtered)	State	Order P	arent Path														1
																			1
																			-
								_	100										
							Now		100 ns		20	ns	40 ns		60 ns	80 ns		100 ns	I
							Cursor 1		15 NS		15 ns								
										<u> </u>									1
ect 🛛 🏭 sim 🖂 🖣	4		1		1	tb_and_ga	ite.v × an	d_gate.v ×	Wave	X								4	>

Figure 15: Modelsim: Timing behavior

Task1: Simulating an AND gate

In this exercise at first simulate an AND gate (Figure 16). Consider the code in listing. This code declares a single module, which you can think of as a class. The module's name is *and_gate*, which is descriptive, as we want to simulate an AND gate.



assign out = a & b; // output function endmodule

Now that we have a module (and_gate), we need to make an instance and simulate it for testing purposes. We only need to simulate and test, so we need to write a test bench. The terminology goes back to the old days of hardwired testing (usually with wire wrap), where you would have a physical testing bench, which typically was set up to rapidly connect and test circuits. The test

bench creates virtual environment to verify the correctness design (in this case "and" gate). See Figure 17 for a conceptual representation.



The testbench file is given below. //name: tb_ and_gate.v

module tb_and_gate(); // Test bench for tb_ and_gate.v reg a,b; // a reg, to allow us to assign the input, and a wire to receive the output wire out; and_gate uut(out,a,b); // this instantiates a and gate, uut is a label initial begin a = 1'b0;// here we apply inputs to the gate b = 1'b0:#10; a = 1'b0;b = 1'b1;#10: a = 1'b1;b = 1'b1;#10; a = 1'b1; b = 1'b0;#10; end // set up the monitoring initial begin \$monitor("a=%b, b=%b, out=%b, time=%t\n", a, b, out, \$time); end endmodule

The **\$monitor** command continuously monitors the values of the variables or signals specified in the parameter list and displays all parameters in the list whenever the values of any one variable or signal changes. **\$monitor** only needs to be invoked once. Only one monitor list can be active at a time.

Step by step intructions:

From modelsim window:

File-> Add to Project->Existing Files: Select file 'and_gate.v' and tb_and_gate.v

Select *and_gate.v* in the project tab

Compile

Now compile *tb_and_gate.v*

To simulate:

Simulate -> Start simulation

Click the plus sign ('+') near work

Choose tb_and_gate and click OK

In the structural window (labeled as 'sim') you can see the structure of the design. The source code for the module that is chosen in the structure window is displayed in the source window.

In the object window: Right click and choose Add to Wave -> signals in Region

To simulate select; Simulation -> Run -> Run -all

In the wave window, you can see the waveform for the selected signals

In this simulation outputs are also visible at main console window.

Study the simulation output and compare them with the code. Examine the output signal of the gate.

ModelSim PE Student	Edition 10.4a	A literature in	And Design of the Owner	in terms that had a 1	A Distance Surgery		Indexed IV		- Anna	transmitt in) –)
File Edit View Comp	pile Simulate	Add Wave Tools	Layout Bookmarks Wi	ndow Help			1.							
🖹 • 🚅 🖶 🛸 🗃	3, 🖻 🛍 j	ລ∴!©•#≣]	- 🌾 🏙 🕮 🙀 🕺	🔁 🕇 🖛 🛸 📰 🛛 100	ns 🛊 🖹 🖹 🗎	🕱 🗯 i 🌇 💁 🖑	1 t 🍕 🕇 🛛	🏝 🛠 🏦 📗	Layout Simul	ate	Colu	mnLayout Defa	ult	
_ ¶ • ¶ • ∯ ¶ • ¶	0 1	10 🗊 🚛 🥟 🗍	t 🖪 🕀 III 🖭 i 🖥	₣₮₠ጚጟጜ	£-+€ E	E - 📴 Search:		🚽 🚓 🦚	Q Q Q ,	le 🔊 🖉 🗍		1.1.5	J	
🕼 sim - Default 🚃 🕬		😂 Objects		Wave - Default					ww					 + •
▼ Instance	Design unit	▼ Name	Va] 🕈 🗢 29 ns 🔊 🕨	- \$ 2.+	Msgs									
tb_and_gate uut * uut * uut *	tb_and_gate and_gate	◆ a ◆ b ◆ out	1'h1 Regi Inter 1'h1 Regi Inter 1'h1 Net Inter	/tb_and_gate/a /tb_and_gate/b /tb_and_gate/out	1h1 1h1 1h1									
		▲ ∰ Processes (Active) = ♥ Name	■ 100 million filtered) Type (filtered)											

Fig4: Simulation output

Task2: Repeat the above steps by creating or_gate.v and tb_or_gate.v

// name: or_gate *module or_gate(out,a,b);* //you list all inputs and outputs, by convention outputs go first output out; // this tells the compile which lines are inputs and outputs *input a, b;* assign out = $a \mid b$; // output function endmodule //name: tb_or_gate module tb_or_gate(); // Test bench for and_gate.v reg a,b; // a reg, to allow us to assign the input, and a wire to receive the output wire out; *or_gate uut* (*out*,*a*,*b*); // this instantiates a and gate, uut is a label initial begin a = 1'b0;// here we apply inputs to the gate b = 1'b0;#10; a = 1'b0;b = 1'b1;#10; a = 1'b1;b = 1'b1;#10; a = 1'b1;b = 1'b0;#10; end // set up the monitoring initial

begin
\$monitor("a=%b, b=%b, out=%b, time=%t\n", a, b, out, \$time);
end
endmodule.

Exercise: Simulate 3input / input AND gate and OR gate. Write appropriate testbench.

Task 3: Types Modeling in Verilog

In this part we will model a logic function in different ways. Consider the following table

sel	In2	In1	out
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Out put can be simplied as

Out=in1. sel +in2. sel' and the implementation is shown in figure 4



Figure 4

There are various ways of modeling in Verilog.

- (1) Structural
- (2) Behavioural level

Let us call this function as mux and now we model the function in the above 3 methods.

(1) Structural

Structural Style: The circuit is specified in terms of lower level components (in this example logic gates, which are Verilog primitives) connected with internal signals. The translation of such a specification into a physical circuit is straight forward.

// lines start with "//" is a comment
// name: mux_gate.v // gate level model

and a1 (a1_o,in1,sel); // defines the a1 gate, see figure not n1 (inv_sel,sel); // defines the inverter gate, see figure and a2 (a2_o,in2,inv_sel); // defines the a2 gate, see figure or o1 (out, a1_o, a2_o); // defines the 01 gate, see figure

endmodule

(2) Behavioural level

1. Behavioural Style: It specifies the circuit in terms of its expected behaviour. It is the closest to a natural language description of the circuit functionality.

// lines start with "//" is a comment // name: mux_behav.v // Behavior: event-driven behavior description construct

endmodule

2. Behavioural Style:

// lines start with "//" is a comment

module mux_behav2(out, in1,in2,sel); //you list all inputs and outputs, by convention outputs go first
output out; // this tells the compile which lines are inputs and outputs
input in1,in2,sel;

assign out=sel ? in1: in2;
endmodule

module tb mux behav();

reg in1,in2,sel;

wire out;

mux_behav UUT (out,in1,in2,sel);

initial

begin

in1 = 1'b0;	// here we apply inputs to the logic
in2 = 1'b0;	
sel = 1'b0;	
#10;	
in1 = 1'b0;	
in2 = 1'b1;	
sel = 1'b0;	
#10;	
in1 = 1'b1;	
in2 = 1'b0;	
sel = 1'b0;	
#10; end	
// set up the monitor	ing
initial begin	
\$monitor("sel=%b, i	$in1=\%b$, $in2=\%b$, $out=\%b$, time=%t\n", sel, $in1,in2$, out , \$time); end
endmodule	