CALIFORNIA STATE UNIVERSITY LOS ANGELES

Department of Electrical and Computer Engineering

EE-2449 Digital Logic Lab

EXPERIMENT 5 BOOLEAN ALGEBRA AND VERILOG SIMULATION

Text: Mano and Ciletti, Digital Design, 5th Edition, Chapter 2

Required software: Xilinx ISE 14.7

5.1 Boolean Algebra is a mathematical approach to deriving equivalent functions. It is desirable to be able to simplify a function in order to design a circuit that uses fewer gates which can result in a more compact, lower-power design or to reduce the propagation delay of a signal through the circuit in order to speed up the circuit.

The table below shows the various identities that can be used to simplify a function.

Identity Name Involution (Double Negative) A'' = ALaw of Intersection A + 0 = A $\mathbf{A} \bullet \mathbf{1} = \mathbf{A}$ Complementary Law A + A' = 1 $\mathbf{A} \bullet \mathbf{A}' = \mathbf{0}$ Idempotent Law A + A = A $\mathbf{A} \bullet \mathbf{A} = \mathbf{A}$ Null Law A + 1 = 1 $\mathbf{A} \bullet \mathbf{0} = \mathbf{0}$ Identity Law $\mathbf{A} + \mathbf{0} = \mathbf{A}$ $\mathbf{A} \bullet \mathbf{1} = \mathbf{A}$ Commutative Law $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$ AB = BAA + (B + C) = (A + B) + CA(BC) = (AB)CAssociative Law Distributive Law A(B+C) = AB + ACA + BC = (A + B)(A + C)(A+B)' = A'B'(AB)' = A' + B'DeMorgan's Law Law of Absorption A + AB = AA(A + B) = A

Boolean Algebra Identities:

The easiest to derive form of a function is the canonical representation, either Sum of Minterms or Product of Maxterms. A minterm is a product term that contains every input variable and a maxterm is a sum term that contains every input variable. Minterm *i* will evaluate to true (1) for input combination *i*. Maxterm *i* will evaluate to false (0) for input combination *i*. The table below shows the minterms (products) and maxterms (sums) for all possible input combinations for a 3-variable function with input variables A, B, and C.

Α	B	С	minterm	Maxterm
0	0	0	m0 = A'B'C'	M0 = A + B + C
0	0	1	m1 = A'B'C	M1 = A + B + C'
0	1	0	m2 = A'BC'	M2 = A + B' + C
0	1	1	m3 = A'BC	M3 = A + B' + C'
1	0	0	m4 = AB'C'	$M4 = A'\!\!+\!\!B\!\!+\!\!C$
1	0	1	m5 = AB'C	M 5 = A' + B + C'
1	1	0	m6 = ABC'	M6 = A' + B' + C
1	1	1	m7 = ABC	M7 = A' + B' + C'

A function can easily be expressed using a Sum of Minterms expression by taking the logical sum (OR) of the function's minterms (ANDs); i.e., for every input combination that produces a one at the output. Likewise, a function can easily be expressed using a Product of Maxterms expression by taking the logical product (AND) of the function's Maxterms (ORs); i.e., for every input combination that produces a zero at the output.

Consider the XOR and XNOR functions shown in this table.

Α	В	$\mathbf{A} \oplus \mathbf{B}$	$(A \oplus B)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

The exclusive-or of A and B, $A \oplus B$, can be expressed as the sum of minterms 1 and 2. That is,

$$A \oplus B = \Sigma m(1, 2) = A'B + AB'$$

The exclusive-nor of A and B, $(A \oplus B)'$ can be expressed as the sum of minterms 0 and 3. That is,

$$(\mathbf{A} \oplus \mathbf{B})' = \sum m(0, 3) = \mathbf{A}'\mathbf{B}' + \mathbf{A}\mathbf{B}$$

The exclusive-or of A and B, A \oplus B, can be expressed as the products of Maxterms 0 and 3. That is, A \oplus B = $\prod M(0, 3) = (A + B)(A' + B')$

The exclusive-nor of A and B, $(A \oplus B)'$ can be expressed as the product of Maxterms 1 and 2. That is,

$$(A \oplus B)' = \Pi M(1, 2) = (A + B')(A' + B)$$

In this part of the experiment you should do the following (in your lab notebook):

- Explore three different functions for implementing the majority voter, V, described in Experiment
 4. The first, referred to as F1, is the function for V from Experiment 4: F1 = AB + BC + AC.
- Write the truth table for the majority voter, V, with inputs A, B, and C (this was derived in Experiment 4).
- Express V using the sum of minterms notation. Call this function F2.
- Express V using the product of maxterms notation. Call this function F3.
- Use the Boolean Algebra identities (see table, pg.1) to prove that F1 = F2. Show the steps and label each step with the appropriate identity/Law. Hint, you can use the Idempotent law to add additional redundant minterms.
- Use the Boolean Algebra identities to prove that F1 = F3. Show the steps and label each step with the appropriate identity/Law.

5.2* In this part of the experiment you will verify the Consensus Theorem (AB + A'C + BC = AB + A'C) by implementing the following two functions using the Verilog Hardware Description Language (HDL), simulating the Verilog functions using the ISim simulator, and verifying that they both produce the same simulation waveform. (That F1 = F2 can also be verified by drawing a K-map for each.)

F1 = AB + A'C + BCF2 = AB + A'C

The purpose of this Experiment is to help you learn the basics of designing combinational logic circuits using a hardware description language (HDL), to learn the basics of how to model a circuit using structural (gate-level) modeling in Verilog, and how to verify that a circuit works properly using a simulator.

Follow along with the instructions provided to implement F1 and F2 in Verilog and to simulate the circuits to verify that F1 is equivalent to F2. Record notes and observations in your manual to assist you in the future when designing circuits in Verilog.

Turn on the lab computer. Open up the Xilinx ISE Design Suite 14.7 (located on the Desktop).



Click ok after reading the tip of the day (don't worry if you don't understand the tip). Click on **New Project**.

>	SE Proje	ct Navi	igator (P.2	0131013)				
File	Edit	View	Project	Source	Process	Tools	Window	/ Layo
	è 🖥	(S 14	D D	×ら	(ଧ) »	P P	ø
Start					↔□₽>	<		
						-		
V Pro	Velcom	e to i mands	the ISE®) Design	Suite			
Pro Op	Velcom ject comr pen Proje	e to f mands	the ISE®) Design	Suite			

Add your username in the path for Location after "C:\Users" (ex: "C:\Users\nwarter). Don't click the ... to navigate as there appears to be a bug in the installation and ISE will crash¹. Then give the project a name (ex: concensus²).

> New Project Wizard	1	\times
Create New Project Specify project log	t cation and type.	
Enter a name, locati	ons, and comment for the project	
Name:	concensus	
Location:	C:\Users\nwarter\concensus	
Working Directory:	C:\Users\nwarter\concensus	
Description:		
Select the type of to	pp-level source for the project	
Top-level source typ	pe:	
HDL	~	
More Info	Next > Cance	I

In this experiment we will be simulating the results in software. It is also possible to synthesize the code to run on hardware. However, we will not be doing that in lab this semester. Therefore, you do not need to specify the device or any of the information in the top half of the **Project settings** window.

¹ Also don't use the Add Source feature in ISE at this point. This will also crash the tool. These bugs have been reported to the IT consultants.

² Note, in the project and filenames used in this design, consensus was misspelled as concensus.

You do need to specify (if not already specified) that the **Simulator** is **ISim** and the **Preferred Language** is **Verilog**. Then click **Next** and **Finish**.

elect the device and design flow for the p	roject	
Property Name	Value	
Evaluation Development Board	None Specified	
Product Category	All	
Family	Artix7	·
Device	XC7A100T	
Package	CSG324	·
Speed	-3	·
Top-Level Source Type	HDL	-
Synthesis Tool	XST (VHDL/Verilog)	·
Simulator	lSim (VHDL/Verilog)	·
Preferred Language	Verilog	
Property Specification in Project File	Store all values	
Manual Compile Order		
VHDL Source Analysis Standard	VHDL-93	
Enable Message Filtering		

To add a source file to the project, under **Project**, select **New Source** (we'll abbreviate this instruction as **Project -> New Source**). In the **Select Source Type** window, select **Verilog Module** and specify the File name (ex: concensus). Also make sure **Add to project** is checked.

> New Source Wizard	X
← Select Source Type Select source type, file name and its location.	
 IP (CORE Generator & Architecture Wizard) Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Package VHDL Test Bench Embedded Processor 	File name: concensus Location: C:\Users\nwarter\concensus
More Info	Next > Cancel

Click **Next**. Specify the names of the input and output ports. In our example, A, B, and C are inputs and F1 and F2 are outputs. Make sure to change the **Direction** of F1 and F2 to **output**.

> New Source Wizard													
Define Module Specify ports for module.													
Module name	concensus												
	Port Name	Direction	n	Bus	MSB	LSB	^						
A		input	\sim										
В		input	\sim										
С		input	\sim										
F1		output	\sim										
F2		output	\sim										
		input	\sim										
		input	\sim										
		input	\sim										
		input	\sim										
		input	\sim				~						
More Info			< Ba	ack	Next >	Cance	el						

Click **Next**, review the summary to make sure it is correct and then click **Next** again. If there you notice a mistake in the summary, click **Back** and go back to correct it.

≽ New	Source Wizard					×
←Summ Pr	n ary oject Navigator wi	ill create a new skeleto	on source with the follow	wing specificatio	ons.	
Add to F Source I Source I Source I	Project: Yes Directory: C:\User Type: Verilog Modu Name: concensus.	s \nwarter \concensus ule v				
Module Port Def	name: concensus finitions:					
	A	Pin	inpu	t		
	В	Pin	inpu	t		
	С	Pin	inpu	rt (
	F1	Pin	outp	but		
	F2	Pin	outp	out		
More I	Info			< Back	Finish	Cancel

A template for your Verilog design should appear. Since we are simulating the circuit, click on the **Simulation** radial button under **Design** next to **View**. If the Verilog code is not showing in the large window, select the Verilog file (concensus.v) in the **Hierarchy** window.

Notice in the template below for concensus.v, there is a module named *concensus*. The module has three input ports, A, B, and C, and two output ports F1 and F2.



Implement the functions F1 and F2 in the module concensus using Verilog HDL. The following logic primitives are available to be used:

not, and, or, nand, nor, xor, xnor

These primitives are the same as the basic gates you tested in Experiment 3. Notice that they are all lower case (case matters in Verilog) and that they turn blue when you type them, indicating that they are reserved keywords. When using a primitive, after the name of the primitive there is a set of parentheses () and between the parentheses are the ports. The ports are the inputs and outputs to the function. When using a primitive, the output port is always first because the basic logic functions (gates) only have one output. However, they can have multiple inputs. For example, we have seen both a two input NAND and a three input NAND. There is no need to indicate the number of inputs in Verilog, instead all ports after the output port are assumed to be inputs).

Consider the Verilog statement: and (AB, A, B);

Notice that the output of the **and** primitive (gate) is given the name **AB** and that its inputs are **A** and **B**. Since the Verilog language is case sensitive, input **A** would be different from **a**. An input or output can have any name provided it consists only of letters, numbers and underscores. Also, it can start only with a letter or underscore and it cannot be a reserved keyword like input or and, etc.. **AB** was the output name chosen here because it signifies the product term **AB**. Notice that the statement ends with a semicolon (;). Before proceeding, refer to the following Verilog file and convince yourself that the code actually implements the functions: F1 = AB + BC + A'C and F2 = AB + A'C.

Notice that the underscore $(_)$ is often used to indicate "not"; e.g., **A**_ is **A not** or **not A**. Also notice that the code for the design is indented (tabbed over) so that it is easy to see what is contained in the module. Extra white space is ignored by the HDL Compiler but you should use tab to make your code more readable.

Type the code for F1 and F2 into your source file (ex: concensus.v). Also modify the comments as shown but include your and your partners names under Engineer.



After you finish entering the design, save the file. If there are any syntax errors indicated in the console window, fix them. For example, delete the semicolon (;) after the **not** primitive and click save. You'll notice that there is an error message indicating that there is a syntax error (Syntax error near "and"). When the HDL Compiler tries to analyze the code it encounters the **and** primitives and it notices that there is an error because there should have been a semicolon (;) encountered before the **and**. Replace the semicolon after the **not** statement and save the file again. The syntax error message should go away.

Under Processes, expand the ISim Simulator dropdown and click on Behavioral Check Syntax. Again, if there are any mistakes, fix them, save the file, and run Behavioral Check Syntax again (click on it again).



In the console window you should see the following message if there are no errors.



To simulate the circuit, we need to provide test inputs just as we use switches or the counter output to provide test inputs to our circuits in Experiments 3 and 4.

Under **Project**, select **New Source** (or right click in the **Hierarchy** window and select **New Source**). Select **Verilog Test Fixture** and name the file (ex: test). Make sure **Add to project** is selected. Click **Next**, **Next**, and **Finish**. Note, that you do want it associated with concensus.v since it will be providing the inputs for that module.

> New Source Wizard	×
←Select Source Type Select source type, file name and its location.	
BMM File ChipScope Definition and Connection File Implementation Constraints File IP (CORE Generator & Architecture Wizard) MEM File Schematic User Document Verilog Module Verilog Test Fixture VHDL Module VHDL Library VHDL Test Bench Embedded Processor	Elle name: test Location: C:\Users\nwarter\concensus
More Info	Next > Cancel

Notice in the following screen shot that the test fixture provides a template for you with the inputs for the "uut" (uut means unit-under-test; otherwise use "dut" for device-under-test); these are already inserted (ex: A, B, C). To test the circuits, values have to be loaded into registers (components that hold values) A, B, and C. This is similar to what we have done in hardware on the breadboard where we have driven the inputs from the counter (which can hold values) or from the switches which can be mechanically/physically set to a particular value. The outputs will be driven on the wires F1 and F2. Under the comment **Add stimulus here**, you should assign various test input combinations to A, B, and C. Notice they have already been initialized to 0. In between each test case, insert a delay using the statement #10 (the value of the delay doesn't matter in this case because we are only simulating for behavior and not for timing).

> I	SE Project	Navigato	or (P.20131	1013) - C:\	Users\nw	arter\	concens	us\con	census.xis	e - [test.v*]
	<u>E</u> ile <u>E</u> dit	<u>V</u> iew	P <u>r</u> oject	Source	Process	<u>T</u> o	ols <u>W</u>	indow	Layout	Help
) 🏓 🖥	# 🕹	1 X G		50		» 🎾 🏓	<i>P</i> 2) @ /	2 🔊 🖻 🗆 📼 🥬 🥓 🕨 🗴 🛠
Desig	jn			↔	□∂×	∎	17	11	Depend	encies:
*	View: 🔘	🔯 Imple	mentation	🔘 🎆 Sir	mulation	ÞE	18		D	
a.	Hierarch	у				_	19		Revisi	on:
Co.	- 🖻 d	oncensus	;			=	20	11	Additi	onel Commente:
a	🗄 🛄 🗴	c7a100t-3	csg324			<u> </u>	21	11	Addici	Shar Commentes.
<u>_</u>	i 1	or 🖥	icensus (c	oncensus.	.v)		23	111	//////	
-						10	24			
<u></u>							25	mod	ule tea	st;
6 23						∧	26			
						%	27		// Inp	its
						126	28		reg A;	
						2	29		reg B;	
						20-	30		reg C;	
						G	32		// Outi	nuts
						0	33		wire F	L;
						-	34		wire F	2;
							35			
							36		// Inst	tantiate the Unit Under Test (UUT)
							37		concent	sus uut (
							38		.A()	A),
							39		. B (1	3),
	🏹 No P	Processes P	Running				40		.C(\ F1	-/, (F1)
ENH.							42		. F2	(F2)
<u>т</u> #1	Processe	s: concen	sus	_			43);	()
Ψ.	<u> </u>	Design	Summary	/Reports			44			
EY		Design	Utilities				45		initia	l begin
~44		Synthe	rize - XST				46		11	Initialize Inputs
	i∰ (2/2)	Implem	nent Desig	In			47		A =	0;
	6	Genera	te Progran	nming File	e		48		B =	0;
	🛓 🙀	Config	ure Target	Device	-		49		C =	0;
	l õn	Analyze	e Design U	Jsing Chip	Scope		50		// 1	Jait 100 mg for global reget to finish
							52		#10):
							53			
							54		// 1	Add stimulus here
							55			
							56		end	
							57			
							58	end	module	
							59			

The various test cases have been inserted into the code below. Note, that this will test every combination from ABC = 000 to ABC = 111. The last statements sets the inputs back to 000 (similar to the counter rolling over). The last test case (ABC = 000) is redundant and was inserted so that you can easily see the test cases in the simulation waveform. Our test cases will start after a delay of 100 (indicated by #100). Note that one unit of delay has been set to 1 ns by default.

EXP.5	(pg.12)
-------	---------

> 1	SE Project	Navigator	(P.20131	013) - C:\l	Jsers\nw	arter\	concensi	us\concens	us.xise	e - [test.v]							
	<u>E</u> ile <u>E</u> dit	<u>V</u> iew	P <u>r</u> oject	<u>S</u> ource	Process	; <u>T</u> o	ols <u>W</u> i	ndow La	yout	<u>H</u> elp							
	1 🏓 🗐	j 🕹	ЖC	B×	50	• :	» 🏓	P 👂 🖉	P	2 🔊	5 E		•	×	\?	▶ 2	ت 🖈 🗈
Desig	jn			↔	08×	€	23	//////	////	///////////////////////////////////////	11111	/////	1111	1111	1111	1111	11111
*	View: 🔘	🔯 Implem	entation	💿 🔜 Sin	nulation	ÞE	24										
	Behavioral				\sim	_	25	module	tes	st;							
Ca l	Hierarchy	/				=	26	11	Inpu	its							
-	- 🗐 o	oncensus				1	28	reg	А;								
00	🖻 🛄 🗴	7a100t-3cs	sg324			=	29	reg	в;								
a	<u>+</u> <u>۷</u>	test (test	.v)			2	30	reg	C;								
						1	32	11	Outr	outs							
EX.d						26	33	wir	e F1	;							
						126	34	wir	e F2	2;							
						×	35	11	Inst	antiate	the	Unit	Und	er T	est	(11117	5
						~	37	con	cens	sus uut	(01110	ona			(001	1
						G	38		.A(A	A),							
						\odot	39		.B(E	3),							
							40		.C(C	。), (〒1)							
						-	42		.F2 ((F2)							
							43);									
						-	44										
						-	45	111	// 1	L begin Initiali	ze In	nuts					
	No Pi	rocesses Ru	nning			-	47		A =	0;		puod					
P 1	Processes	: test					48		в =	0;							
91	in 🎾	ISim Sim	ulator				49		C =	0;							
~~~~	5	Behav	vioral Ch	eck Synta	c l	-	50		// 14	Jait 100	ns f	or al	loba	l re	set	to f	inish
70		🛛 Simu	late Beha	ivioral Mo	del		52		#100	);		- g.					
							53										
							54		// 2	Add stim	ulus	here					
							55		#10 #10	B = 1;	C = 0						
							57		#10	C = 1;							
							58		#10	A = 1;	B = 0	; C =	= 0;				
							59		#10 #10	C = 1;	c - 0						
							61		#10 #10	C = 1;	C - 0	·					
							62		#10	A = 0;	B = 0	; c =	= 0;				
							63										
							64 65	end									
							66	endmod	ule								
							<										

Save the file and make sure that there are no syntax errors. Make sure that your Design window still has the Simulation View clicked. Next, click on test.v in the hierarchy and then click on **Simulate Behavioral Model** under the **ISim Simulator** drop down in the **Processes** window.

The simulator window will pop up.

Sim (P 20131013) - [Default work]									_	пх
File Edit View Circulation	Wedaw Issued IIs	1-								
File Edit View Simulation	Window Layout He			1						
🗋 🏓 🖥   👹    🐰 🛍 🖄 🗙	🔊 🖄 🗠 🗢	110 5		<i>₽</i> <b>№</b> ?	<i>&gt; &gt;</i>	<u>8 / 2   2   1</u>	±±± ‡	^ ^i   🖸 🕨	× 1.00	us 🗸 🔙 💠
Instances and Processes ↔ □ ₽	Objects	↔ 🗆	8 × 🏓						1,000,	,000 ps
📳 🛞 🖃 🗲 블 F(x) 📃 🗲	Simulation Objects f	or test	<i>&gt;</i>			1000,007	1000,009	1000,000		1.000.000 pc
Instance and Process Name		L L 🔛	<b>1</b>	Name	Value	999,997 ps	1 966'666			1,000,000 ps
test	te Object Name	Value	, , , , , , , , , , , , , , , , , , , ,	L@ F1	0					
b glbl	gl La F1	0		10 1						
	16 F2	0		10 4						
	16 A	0								
		0		-0	ľ					
		-	<u> </u>							
			1 🛉 🛉							
			1							
			-							
			1154							
			121							
			- XI							
			2							
						X1: 1,000,000 p	os			
<	>			$\langle \rangle$	< >	<				>
🛃 Instanc 🛗 Memory 🔰 Source	2 🗶 👘		> 🕅		Default.w	cfg				
Console										↔□₽;
WARNING: A WEBPACK license was found.										,
WARNING: Please use Xilinx License Config	uration Manager to check	out a full ISim license.								
WARNING: ISim will run in Lite mode. Please This is a Lite version of ISim	e refer to the ISim docume	entation for more inform	nation on the o	lifferences be	tween the	Lite and the Full ve	ersion.			
Time resolution is 1 ps										
Simulator is doing circuit initialization proces	ss.									
ISim>										
Console Compilation Log	🕨 Breakpoints  🕅 Fi	ind in Files Results	Search Re	sults						
								[	Sim Time:	1,000,000 ps

Click on the Zoom to full view button **F**. You should now see the following waveform. If you do not, you need to go back and check your code to see if there are any mistakes.

P (			1,000.000 n
2	Name	Value	0 ns 500 ns
20	16 F1	0	
~	16 F2	0	
(F)	1 <u>6</u> A	0	
$\bigcirc$	16 в	0	
1	1 <u>6</u> C	0	
<b>⊉</b> r			

To expand you can click on the + magnifier and move the slider at the bottom over (this is similar to adjusting the settings on the oscilloscope).

								20	3.7	30 n	s	
Name	Value	 100	ns		150	ns		200	ns			250 n
🗓 F1	0											
🗓 F2	0											
16 А	0											
Ць в	0											
16 с	0											

In the waveform, note that the starting at 100 ns, the values of ABC count from 000 to 111 and at 180 ns go back to 000. Reading the simulation waveform is similar to reading the oscilloscope waveforms except in this case we can see all inputs and outputs together. This is a very useful function and in fact, there are logical analyzers that are similar to oscilloscopes that capture and display multiple inputs and outputs in a digital hardware design.

Note that F1 and F2 have the same waveform which proves that they are the same function. If you do not obtain the correct results, there must be an error in your design that you need to fix. In this example, if your results do not show that F1 and F2 are the same, there is an error in your design in the consensus module. Go back and fix it and repeat the steps to simulate the code.

The yellow bar can be moved over the waveforms to check the values of the inputs and outputs at different times during the simulation. For example, at time 135.23 ns, ABC are 011 and F1 and F2 are 1.

Æ					135.230 ns							
P	Name	Value	10	0 ns		150 ns			200 r			
~	🗓 F1	1										
~	🗓 F2	1										
R	16 А	0										
$\odot$	ЦЬ В	1										
1	ЦЬ с	1										
<b>≁</b> r												

Show your instructor your code and simulation results. Also, print out and paste the code and simulation results into your lab notebook.

Answer the following question in your lab notebook.

Question: Describe intuitively why the term **BC** is redundant in **F1**. Hint: consider the other two terms **AB** and **A'C** and the possible values of **A**. (Example: if A=1, then F1 = BC + B. Is BC needed?).

**5.3*** Use Xilinx ISE to implement the three functions for the majority voter in Verilog and run a behavioral simulation to verify that the three functions are the same.

F1 = AB + BC + AC
F2 = Sum of Minterms (determined in part 5.1)
F3 = Product of Maxterms (determined in part 5.1).

Follow the process outlined in section 5.2 to design your circuit in Verilog using structural modeling and to simulate your design. Show your instructor your code and simulation results. Also, print out and paste the code and simulation results into your lab notebook.

Answer the following question in your lab notebook.

What is the benefit to designing (and simulating) a circuit in a hardware description language (HDL) such as Verilog before building it in hardware?