

論理回路とVerilog-HDL

八巻隼人

(yamaki@uec.ac.jp)

西10号館-519

この実験資料について

- ・ダウンロード用 URL
 - <u>http://www.hpc.is.uec.ac.jp/yamaki_lab/teaching-jp.html</u>
 - 資料と共にsampleファイルが置いてあります。
 授業で出てくるvファイルはここに入っているので参考にしてください。(./~.scというスクリプトを実行するとiverilogコンパイルから実行まで一気に行ってくれます)
- ・佐藤証先生の「J2課題解説スライド」がベース
 - ・佐藤証先生のオリジナル→ <u>http://satoh.cs.uec.ac.jp/ja</u>





- ・赤い本:
 - ・Verilog-HDL の勉強用
- ・青い本:
 - ・ツールの使い方の勉強用





J2課題の目的

・論理回路の特性 / 設計技術を習得する

Q Q Q Q I <th>120 ns 120 ns 120 ns 0</th>	120 ns 120 ns 120 ns 0
Name Value 0 ns 20 ns 40 ns 80 ns 80 ns 100 ns	120 ns 140 ns
Index O Index Ind	
Name Nam Name Name	0
> Mikuwa/1:0] 0 0 3	0
> Mash(7:0) 00 00 00 00 Whit(flag) 0 <td></td>	
We hit flag 0 <th0< th=""> 0 <th0< td=""><td></td></th0<></th0<>	
lè miss_flag 0	
> M state_updated[7:0] d2 e4 e1 d2 93 cf X 93	d2
> Mf state_now(7:0] d2 d2 e4 4 e1 d2 93	d2
> Minit_priority(1:0) 0 x 1 x 2 x 8 x 0 x 3 x 0 x 3	0
> M neighbor_way(1:0) 2 X 3 X 2 X 1 X 0 X 2 X 3 X 3 X 2 X 3 X 3 X 2 X 3 X 3 X 3	2
> WWAY_WIDTH[31:0] 00000002 00000002	
> W STATE_WIDTH[31:0] 00000008 00000008 000000000000000000	
> M ADDR_WIDTH[31:0] 00000008 00000008 000000000000000000	
> 1 d2 c4 c1 d2 93 ct X 93	d2
> ¶ addr[7:0] 00 00 00 00	
> M dout[7.0] d2 c4 c1 d2 33	d2
> 🕷 mem [255.0][7:0] 🕹 4, 64, 64, 64, 64, 64, 64, 64, 64, 64,	e4, e4, e4, c4, 🗆 👘 👘
> * #ht_way(1:0] 0 0 0	0
> ¹ state[7:0] d2 <u>e4</u> <u>e1</u> <u>d2</u> <u>93</u>	d2
miss_flag 0	
> Mint_priority[1:0] 0 0 1 2 3 0 3 0 3	0
> M spit_wire[10] 3 3 1 3 2	3
> M WAY_WIDTH[31:0] 00000002 00000002	
> M STATE_WIDTH[31:0] 00000008	
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	0
> \mathfrac{\mathrac{\matrir\}\matrir\} \mathrac{\mathrac{\matrir\} \matrir\} \matrir\} \matrir\} \matrir\} \matrir\} \matrir\} \matrir\} \matrir\} \matrir\} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	d2
$ \sum_{n=1}^{\infty} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{j=1}^{n} \sum_{$	2
> \mt cmp_way(1:0] 0 0	
> Micmp_way1[1:0] 0 0 1 1 0 0	
> Mcmp_way2[1:0] 2 0 2 0 2 0 2 0 2 0	2
> M cmp_way3[1:0] 0 X 3 X	0
> WWAY_WIDTH[31:0] 00000002 00000002	
M WAY_WIDTH[31:0] 00000002 00000002 > M STATE_WIDTH[31:0] 00000008 00000008	
> MWAY_WIDTH[31:0] 00000002 00000002 > MSTATE_WIDTH[31:0] 00000008 00000008 > MINL_way(1:0) 0 0 3	0
> MWAY_WIDTH[31:0] 00000002 00000002 > M STATE_WIDTH[31:0] 00000008 00000008 > M ML_way(1:0) 0 0 3 Is top_flag 1 0 0 0	0
> MWAY_WIDTH[31:0] 00000002 00000002 > MSTATE_WIDTH[31:0] 00000008 00000008 > Mikt_way(1:0) 0 0 1 top_flag 1 0 > Mikt_way(1:0) 0 3	0

J2課題の実施方法

・今年は対面実施となります.

- ・実施内容や課題は本資料に載っています
 - それとは別に、問題のヒントを以下のリンクに随時upします. http://www.hpc.is.uec.ac.jp/yamaki_lab/teaching-jp.html
 - わからないことがあったら対応するので私にメールください

- 八巻 (<u>yamaki@uec.ac.jp</u>)



- ・実験初日(~p43)
 ガイダンス
 簡単な組み合わせ回路の設計
 問題1~3
- 実験2日目(~p60)
 順序回路の設計
 問題4
- 実験3日目(~p71)
 加減算回路の設計
 問題5~7

レポート

<u>記載内容と採点基準</u>

- この資料の問題1~7を全て行い、その内容に考察を加えレポートとする
- 問題1~7全て回答できていることが修了の条件
- 更に出来る人は問題8を解くと加点
- 例年,問題7まで解けずに単位を落とす人が結構います. わからない部分は恐れず質問してください.

<u>提出方法</u>

- レポートPDF ファイルを WebClass 上で提出.
- 受付期間内なら何度でも再提出可.

<u>受付期間</u>

- 授業終了から3週間後の23:00まで
- 前半組: 7/30(水)の23:00まで
- 後半組: 8/11(月)の23:00まで

目次

- 本実験の内容とその意義(論理回路とハードウェア設計)
- verilogの使い方
- ・簡単な組み合わせ回路(問題1~3)
- •順序回路(問題4)
- ·加減算回路(問題5~7)



論理回路とは

・デジタル回路の数学的モデル

 ・デジタル回路:1と0のデジタル信号のみを用いて様々な計算をする回路 (一般的には電圧のHighとLowを1と0にデジタル化)



・アナログ回路:連続するアナログ信号を用いて様々な計算をする回路



デジタル回路の利点



・ある機能を実現したい場合、アナログ/デジタル回路どちらでも可だが



ハードウェア?ソフトウェア?

• 文脈によりそれぞれが指す意味は多少異なる

一般的には

- ハードウェア:物理的実体を持つ機械部品. 例. CPU,メモリ,マウス,キーボード,ディスプレイ

- ソフトウェア:物理的実体を持たない構成要素. 主にプログラムのこと. 例. OS, アプリケーションプログラム, ドライバ

~処理において使われる時(ハードウェア処理,ソフトウェア処理)

- ハードウェア:専用の論理回路による処理
- ソフトウェア: CPUによる (プログラムでの) 処理



ハードウェアとソフトウェア 何が違う? ~ソフトウェア(CPU)の場合~

CPUの強みは何でも (プログラムで記述すれば) 処理できる = 汎用的
 ⇒ 何かの処理に特化しているわけではない

例えば…

• 次のような計算をCPUで実行する場合

 $x = (a \times 10 + 20) + (b \times 100 - 50)$

CPUでは以下のように実行に 5 命令(5 サイクル)を要する
 乗算命令 → 加算命令 → 乗算命令 → 減算命令 → 加算命令

ハードウェアとソフトウェア 何が違う? ~ハードウェアの場合~

• その処理専用の回路(ASIC)を作れば高速に処理できる

例えば…

• 先ほどの例↓

 $x = (a \times 10 + 20) + (b \times 100 - 50)$

ハードウェアでは1サイクルで計算が完了する



x = (a×10 + 20) + (b×100-50) を計算する回路 (この回路は適当…)

Tips ソート処理を考えてみると より差がわかりやすい。 ソート処理はCPUの場合, 値の大小で入れ替えを繰り 返すので結構なサイクルを 要する 一方でハードウェアの場合. ソート回路に値を入れると 1サイクルで結果が得られる 招高速

ハードウェアとソフトウェア 何が違う? ~まとめると~

- ハードウェア処理は<mark>超高速</mark>
- また、(あまりにも複雑な処理でなければ) CPUやメモリを実装するより
 専用論理回路を実装したほうが小規模



小型とか超高速を重視する処理において有用

例えば…

·動画像処理

・組み込み機器系(監視カメラ、loTセンサとか)の処理

・ルータのパケット処理

ハードウェア設計(この実験)

• ハードウェア(専用論理回路)はどうやって作る?

① まず仕様を満たす論理回路の設計図を作成する

- 仕様から状態遷移図・真理値表を描き,入出力の論理式を求める
- 導いた論理式をもとに回路図を作成

論理回路設計の講義でやった

② テストベンチを用いて設計した回路の動作を検証

- 回路作成には膨大なお金がかかるので失敗はできない
- シミュレーションにより回路が所望の動作をするか徹底的に検証

ハードウェア記述言語 verilog

③ 論理回路(実物)を作成
 膨大なお金をかけて外注する

ハードウェア設計の③について

- 実物の回路を作ってもらうにはとにかくお金がかかる(数百万円~)
- そこで最近は FPGA (Field Programmable Gate Array) が出てきた
- FPGAは…
 - 論理回路を(外注せずとも) 自作できるハードウェア
 - FPGAのお値段は数万円~
 - 回路を作る費用はタダ
 - しかも設計した論理回路を
 自分でPC上で焼ける
 - 何度でも焼き直し可能という
 夢のようなデバイス



FPGA(Oイメージ)

Field Programmable Gate Array

- ゲート (トランジスタ) がアレイ (格子) 状に配置されている
- Verilog で作成した回路となるようにFPGAのゲートを配線
 → 所望の論理回路を作成できる(配線しなおすことも可能)





Verilog-HDL の例: AND回路のモジュール (sample1.v)

まずは以下のverilogプログラムを自分で作ってみましょう
 コマンドラインで \$ vi sample1.v または \$ emacs sample1.vとし
 以下を作成

module	<pre>sample1(s, a, b);</pre>	/* モジュール名(入出力端子)
input	a, b;	/* 入力端子a,bの宣言
output	S;	/* 出力端子sの宣言
assign endmodule	s = a & b;	/* s に a (and) b を接続



Verilog-HDL の例: wireの使い方 (sample2.v)

- ・モジュール内で必要になる配線はwireで宣言
- assign文は出力や配線に接続するための構文(後述するregはassignを使わない)

module	<pre>sample2(s, a, b);</pre>	/* モジュール名(入出力端子)
input	a, b;	/* 入力端子a,bの宣言
output	S;	/* 出力端子sの宣言
wire	w1, w2;	/* 配線w1, w2の宣言
assign	w1 = a & b;	/* w1 に a (and) b を接続
assign	w2 = a b;	/* w2 に a (or) b を接続
assign	s = w1 & w2;	/* s に w1 (and) w2 を接続
endmodule		



Verilog-HDL の例:モジュールの入れ子 (sample3.v)

• モジュール内で別モジュールを呼び出すことも可能





モジュールの実装→動作確認 (テストベンチ)

- 以上で簡単な回路ならもう記述できるようになりました
- モジュールが完成したら、その動作確認が必要
- ・以下の回路なら, aとbにテスト信号を流し, 正しいsが得られるか確認 a, bが(0,0), (0,1), (1,0), (1,1)の時, sが 0, 0, 0, 1 になるかどうか
- 作ったモジュールに手を加えて動作確認するのはよろしくない
 - そこで動作確認用のテストベンチモジュールを作る



Verilog-HDL の例:sample1のテストベンチ (sample1_test.v)





iverilogを使った動作確認

・コンパイル

iverilog -o 実行ファイル名 -s トップモジュール名 コンパイルしたい.vファイル群

@blue99 sample]\$ iverilog -o sample1 -s sample1_test sample1_test.v sample1.v

→ エラーが出なければsample1というファイルが作成される

・シミュレーションの実行

vvp 実行ファイル名 または 実行ファイルを直接指定 ./sample1

Γ			@blue99	sample]\$ vvp sample1
а	b	S	time	\$displayの内容
0	0	0	0	
0	1	0	50	Augusticesの中京
1	0	0	100	Şmonitorの內容
1	1	1	150	

→ 自分が意図した通りの結果となっているか確認

一致検出回路(eq.v)









ModelSimの使い方

すみません,今年度はCEDの環境変更に伴いModelSimが使えません

代わりに gtkwave というツールを使います 使い方は Webclass の実験ページに書いてありますので そちらを参照してください

ModelSimの起動とプロジェクト作成

- 1. コンソールで"mkdir ~/mics_J2"と入力
- 2. コンソールで"vsim" (今年は"vsim2"?) と入力
- 3. File→New→Proeject... で"Create Project"ウィンドウが開く
- 4. Project Name: eq
- 5. Project Location : / *path_to_homedir*/mics_J2

ModelSim ALTERA S	STARTER E mpile <u>S</u> ir	DITION 10.1d - Custom Altera Version nulate A <u>d</u> d Library T <u>o</u> ols Layo <u>u</u> t Boo <u>k</u> marks <u>M</u>	indow Help		Create Project
■ • ⊭ ∎≉®	<u>X</u> 🖻 🕯	8 12 12 (> - M 🗄 🗖 🕹 🕮 🚑 🕅 ‡ A	🛊 🍰 🌨 🏦 🛛 Layout NoDesign		Project Name
ColumnLayout AllCol	umns] ≪ - ⊷ - 🥵 - 🥰			led
Library				+ # X	
▼ Name	Type	Path		•	Project Location
work (empty)	Library	C:/J2/work			Cr. (T2
+ 1 220model	Library	\$MODEL_TECH//altera/vhdl/220model			Drowse
+ 220model_ver	Library	\$MODEL_TECH//altera/verilog/220m			
	Library	\$MODEL_TECH//altera/vhdl/altera			Default Library Name
	Library	\$MODEL_TECH//altera/vhdl/altera_l			
→ altera_Insim_ver	Library	\$MODEL_TECH//altera/verilog/altera			work
	Library	\$MODEL_TECH//altera/vhdl/altera_mf			
	Library	\$MODEL_TECH//altera/verilog/altera			
	Library	\$MODEL_TECH//altera/verilog/altera			Copy Settings From
+- altgxb	Library	\$MODEL_TECH//altera/vhdl/altgxb			O (madalain and (madalain ini
	Library	\$MODEL_TECH//altera/vhdl/altgxb			0/modelsim_ase/modelsim.ini browse
	Library	\$MODEL_TECH//altera/verilog/altgxb			Copy Library Mappings O Reference Library Mappings
⊕- arriagx	Library	\$MODEL_TECH//altera/vhdl/arriagx			torony rispy constraints and the constrainty rispyings
+ arriagx_hssi	Library	\$MODEL_TECH//altera/vhdl/arriagx			
⊕arriagx_hssi_ver	Library	\$MODEL_TECH//altera/verilog/arriag			OK Cancel
III arriany ver	Library	MODEL TECH/ /altera/verilog/arriagy			
Q Transcript					
# Loading project	ea				
# reading C:\alter	-ч a\13.0∖mo	delsim ase\win32aloem//modelsim.ini			
		=			
ModelSim>				-	
		<no design="" loaded=""></no>	<no context=""></no>		

Verilog-HDLファイルの追加

- 1. "Add items to the Project"ウィンドウ: Add Existing File またはメインウィンドウから Project→Add to Project→Existing File…
- 2. "Add file to Project"ウィンドウ: Browse
- 3. "Select files to add to project"ウィンドウ:
- 4. eq.v と eqSim.v を選択して開く.
- 5. "Add file to Project"ウィンドウ: OK
- 6. "Add items to the Project"ウィンドウ: Close

Add items to the P	Project
Click on the icon to a	dd items of that type:
Create New File	Add Existing Pile
Create Simulation	Create New Folder
	Close

File Name	
I	Browse
Add file as type	Folder Top Level
• Reference from current location C	Copy to project directory

コンパイル

- ・メインウィンドウ:Compile→Compile All
- Verilog-HDLファイルのStatusが?からレに変わり、"Compile of eqSim.v was successful." "Compile of eq.v was successful"と出る.

ModelSim AL	TERA STARTER EDITION	N 10.4d								- 0) ×
<u>F</u> ile <u>E</u> dit <u>v</u> iev	v <u>C</u> ompile <u>S</u> imulate	e A <u>d</u> d <u>P</u> roject T <u>o</u> ols L	ayo <u>u</u> t Boo <u>k</u> marks <u>W</u> indow	<u>H</u> elp							
Compile	ntions	≥ ⊘ - M 🗄 🖬] 🦠 🗱 🕮 🙀 🕺 🛛 🧣	b 👚 🦇 🛸	토루 100 ps 🔶	1 1 1 1 1 🕅 🍩 🛛	💁 🚇 🕚	🕇 🏵 🏌 🛔	h 🔝 🏦		
	ink	ColumnLayout AllColumn	3	🚽 🛛 🖏 - 🚜	- 💕 🕰 - 🤹 📗		2				
							.]		A Objects		+ @\>
Complie A	atus Type	Order Modified							* Name	Valu 1€	Now 🕈
Compil <u>e</u> St	rder Verilog	1 06/01/2016 03:43:42							🔶 s	StX	Net In
Compile R	eport Verilog	0 06/03/2016 06:40:05							🥎 x		Regi Int
Compile S	Immary.								🔷 у		Regi In
Complic og	2										
									•		
									Active)		+ @1>
									* Name	Type (fi	iltered)
									#INITIAL#5	Initial	
📕 Library 🛛 🛗	Project 🗙 🛺 sim 🛛]						< >	•		
											:+ @ >
# Start time:	: 16:24:26 on Jun	03,2016									
# Loading wor	rk.eqSim	,									-
# Loading wor	rk.eq										
VICTA 45											
JV51M 4>											32
		Proj	ect : eq_Now: 0 ps_Delta: 0		sim:/eaSim						

コンパイルエラー

- エラーがあるとTranscriptウィンドウに赤く表示されるので、そこ
 をダブルクリックするとエラーの詳細が表示される
- ・さらに"** Error: "の赤い行をクリックするとソースコードウィンド ウが開いてエラーの行がオレンジで表示される

M2/Verilog-HDL/eq.v Unsuccessful Compile	
vlog -work work C:/J2/Verilog-HDL/eq.v Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov 2	Version
2012	Layo <u>u</u> t Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp
Compiling module eq	
** Error: C:/J2/Verilog-HDL/eq.v(5): (vlog-2730) Undefined variable	
^{*_**} 5行目の'_a'に文法エラー	w 03 + 23 + 23 + 24
Compiling module eqSim	
Close	
Project - C:/J2/eq	C:/J2/Verilog-HDL/eq.v - Default 🖬 🛋 🗙
Name Status Type Order Modified	Ln#
eg.v 🗙 Verilog 0 05/18/14 11:36:38 PM	1 module eg(s, a, b); /* êvoñH
	2 input a, b;
コンハイル天敗	3 output s;
	4 wire na, nb, s1, s2;
	5 assign #5 na = _a, nb = ~b;
	6 assign #10 s1 = a & b, s2 = 💌
Library × 🛗 Project ×	<u> ∼aが_aになっている</u>
A Transcript	»====================================
# Compile of eq.v failed with 1 errors.	
ModelSim> 1つのコンパイルエラー	
	_
Ln: 5 Col: 0 Proje	ct : eq <no design="" loaded=""> <no context=""> 3</no></no>



- Simulate→Start Simulation
- Start Simulationウィンドウでwork/eqSimを選択

Start Simulation	10.00	Confightion - Solution	×
Design VHDL Verilog	Libraries	SDF Others	«»
▼ Name	Туре	Path	
- work	Library	work	
—M eq	Module	C:/J2/Verilog-HDL/eq.v	
🗆 🕅 eqSim	Module	C:/J2/Verilog-HDL/eq.v	
+ 220model	Library	\$MODEL_TECH//altera/vhdl/220model	
	Library	\$MODEL_TECH//altera/verilog/220m	
+	Library	\$MODEL_TECH//altera/vhdl/altera	
⊕altera_Insim	Library	\$MODEL_TECH//altera/vhdl/altera_l	
⊕→ altera_Insim_ver	Library	\$MODEL_TECH//altera/verilog/altera	
⊕_ <mark>_</mark> altera_mf	Library	\$MODEL_TECH//altera/vhdl/altera_mf	
		·····	
•			-
Design Unit(s)		Resolution	
work.eqSim		default	
Optimization			
Enable optimization		Optimization Opti	ons
		OK	Cancel

シミュレーション設定

- シミュレーションウィンドウが開く
- Add→add to wave→all items in regionで波形ウィンドウが開く
- "All items in region and below"と"All items in design"は下位レベ ルやソースの全ての信号線が表示される

ModelSim ALTERA STARTER EDITION 10.1d		x				
File Edit View Compile Simulate Add Structure Tools Layout Bookn	rmarks Window Help					
Image: Simulate Image: Simulate <td< td=""><td></td><td>*</td></td<>		*				
To Dataflow		₫ X				
Instance Design of the Desig	Image: Solution of the second of the seco	5				
Ibrary × I Project × Sim ×	Image: Ware with the second	7: 				
Transcript ::::::::::::::::::::::::::::::::::::						
Project : eq Now: 0 ps Del	elta: 0 sim:/eqSim	11.				







- •メインウィンドウで, Simulate→Run→All
- Finsh Vsimウィンドウで「いいえ(N)」を選択
 - ・「はい(Y)」を選択するとModelSimが終了してしまうので注意




- ・Waveウィンドウに結果が表示される
- ・ 画面を虫眼鏡で拡大して表示

	x
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>A</u> dd F <u>o</u> rmat <u>T</u> ools Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp	
Wave - Default	★ # ×
E + 🗃 🖶 % 🖨 🌡 🖻 🛍 😂 🔄 🔕 + 🏘 🖺 🦘 🕮 🚑 🌋 💁 🕇 🖛 🛶 Eff 100 ps 뉮 El El El Aŭ 🍩 🌇 🎕	
▋ ┇ ╺み ┇ ! ဵ Ѧ ♠ ┃ ╺┛╸ ╶╝ ╡ - ╡ ┃ ヽ ҧ Ѧ ュュ ! ┣ ┃ ┇ ┟ ┶ ┶ ┶ ┶ ┶ ┶ ┰ ज	
→ + +€ + 📴 Search: 🔽 🖉 🔊 🖏 👘 🛛 🞯 🔍 🔍 🔍 🔍 🗍 🔟 📗 📗 🚛 👘 🗊 丁	
Msgs	
VeqSim/s -No Data-	-
File Ed	
Now 200 ps 50 ps 100 ps 250 ps 250 ps 300 ps 350 ps	
O ps to 380 ps Now: 200 ps Delta: 0	
→ + + + E + B+ Search: → M = M = M = M = M = M = M = M = M = M	
Sa - Msgs	
A /eqSim/s -No Data-	
✓ /eqSim/x -No Data-	
▲ I T O D D D D D D D D D D D D D D D D D D	
Cursor 1 680 ps 680 ps	
	38



シミュレーションモジュールに\$monitorコマンドを記述しているのでTranscriptウィンドウにテキストでも表示される

ModelSim ALTERA STARTER EDITION 10.1d							
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d T <u>r</u> anscript T <u>o</u> ols Layo <u>u</u> t Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp							
🖹 • 😂 🔛 🦈 🚭 🐰 🖮 🏙 🏠 ① ② • 🗰 🏗 🗰 🦠 🎬 🏭 🔀							
🖓 🛧 🛶 🐘 🗄 🖺 TOO ps 🗣 🖺 El El El 🗱 🏔 🕴 🥎 🥎 🕀 🕴 🛊 🗞 🏦 💧 Layout Simulate 🗸 🗸							
😨 sim - Default C:/J2/Verilog-HDL/eq.v (/eqSim) - Default 土 西 エ							
TInstance Design unit Design unit type Visibility Total co 🛆 Name 🔶 Ln#							
eqSim eqSim Module +acc=<							
$+ g_1 \qquad eq \qquad Module \qquad +acc=< \qquad 18 \qquad x=0; y=0;$							
= 20							
→ ▼Name → 20 #50 x=1.							
Library × 🕮 Project × 🛺 sim ×							
Transcript 🔤 🛃 🛃							
VSIM 5> run -all							
# x y s time							
# 0 0 X 0							
# 0 1 0 75							
# 1 0 0 100							
# 1 1 0 150							
# 1 1 1 170							
<pre># ** Note: \$finish : C:/J2/Verilog-HDL/eq.v(22) # Time: 200 ps_Iteration: 0_Instance: (eqSim)</pre>							
# 1							
Project : eq Now: 200 ps Delta: 0 sim:/eqSim/#INITIAL#14							

再シミュレーション

- メインウィンドウで Simulate→Restart
- Restartウィンドウが出るのでOKをクリック
- •「シミュレーションの実行」に戻って再シミュレーション
- ・全てのコマンドはTranscriptウィンドウで実行されているので, ここに手入力した方が早い ■ Restart ■ ■

ModelSim ALTERA STARTER EDITION 10.1d	Keep:
<u>File E</u> dit <u>V</u> iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d T <u>r</u> anscript T <u>o</u> ols Layo <u>u</u> t Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp	✓ List Format
B + 🗃 🖓 🚭 X № 🏶 🖄 🖄 ⊘ - M 😫 🗖 🦠 🕮 🚑 🔀	Wave Format
🖥 🛉 🖛 🛶 Eff 100 ps 🕈 El El El M 🚳 🧯 🧊 🐨 🕈 📩 🚓 🏦 Layout Simulate 🗨	Breakpoints
	✓ Logged Signals
	Virtual Definitions
	Assertions
	Cover Directives
🖉 sim - Default : H 🗹 🗶 🎓 :ts ::::::: H 🗹 🗶 🗋 C:/J2/Verilog HDL/eq.v (/eqSim) - Default::::::::::::::::::::::::::::::::	ATV Format
eqSim eqSim Module +acc=< ↓ s 18 x=0; y=0;	
Image: transmission of the second	OK Cancel
X #Normalize Indecess	
(♣ e) :::::: ± ₫ × 22 #50 \$FINISN; 23 end	
#Name 24 endmodule	-
A ASSIGN A ASSIGN	
Image: All project Image: All project <td>< ></td>	< >
R Transcript	
VSIM 10> restart -f	-
Project : eq [Now: 0 ps_Delta: 0]sm:/eqSim	40

内部信号の表示

- ・Simウィンドウでモジュールを指定し,内部信号を表示
- ・\$monitor関数では "g1.s1"のようにモジュール内の信号やさらに"."でつ ないでモジュール内のモジュールを指定

ModelSim ALTERA STARTER EDITION 10.1d							
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d T <u>r</u> anscript T <u>o</u> ols Layo <u>u</u> t Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp							
Layout Simulate		ColumnLay	out AllColum	ns			⊻] ≪ - ≪ - ≪ ∰ - ≪] I O 100 [] AL ≫] N 52 🛧 10 I I 🕸
🔊 sim - Default		=±∎×	💊 Objects 🚃	= ::::: ± :	ð 🗙 🗋 🕻	C:/J2/Ve	erilog-HDL/eq2.v (/eqSim2/g1) - Default 👬 🗹 🔟
▼ Instance	Design unit	Design ur	▼ Name	1	Value Lr	n#	Now ≑ →
eqSim2	eqSim2	Module	🥠 а	5	St1 5	5	assign $s1 = \sim (a \& b)$, $s2 = a b;$
📮 🗾 g1	eq2	Module	🥠 Ь		St1 🧧 🧯	5	assign s = ~(s1 & s2);
ASSIGN#5	eq2	Process	🔷 s		St1 7	7	endmodule
— #ASSIGN#5.	eq2	Process	🧇 s1	5	St0 E	3	
ASSIGN#6	eq2	Process	🔶 s2		st1	9 1	module eqSim2;
j ⊒ g2	eq2	Module			10		wire s1, s2, s;
ASSIGN#5	eq2	Process			12	, ,	$reg x_1, y_1, x_2, y_2;$
	eq2	Process			13	3	$cq_2 = g_1(s_1, x_1, y_1), g_2(s_2, x_2, y_2),$
ASSIGN#6	eq2	Process			14	1	initial
ASSIGN#13	eqSim2	Process			15	5	begin
└── #INITIAL#14	eqSim2	Process			16	5	\$monitor(" %b %b %b %b %d",
😤 #vsim_capacity#		Capacity			17	7	g1.s1, g1.s2, g2.s1, g2.s2, s,\$stime);
					18	3	<pre>\$display("g1.s1 g1.s2 g2.s1 g2.s2 s time");</pre>
					19	э 🛛	x1=0; y1=0; x2=0; y2=0;
					20)	#50 x1=1;
					21	L	#50 y1=1; y2=1;
					22	2	#50 x2=1;
]			23	3	#50 \$finish;
Project 🗶 🛄 Library	🗶 🏭 sim 🛛		•			4	<u>۱</u>
							:+ax

# g1.31 g1.32 g2.31	g2.32 3		time				
	o ô		50				
# 0 1 1	1 0		100				
# 0 1 0	1 1		150				
# ** Note: \$finish	: C:/J2	/Verilog-	HDL/eq2.v(23)			
# Time: 200 ps	Iteration:	0 Insta	nce: /eqSim2				
			Dr	piect : en2 No	w: 200 ps - F)elta: 0	Module
U			Internet	ojecci eqe (140	200 p3 1	icitai 0	, resource //

別回路のシミュレーション

- Simulate→End Simulation
- Projectウィンドウで右クリックしてポップアップメニューを出し, Add to Project→Existing File...
- ・eq2.vと eqSim2.v を追加

ModelSim ALTERA STARTER EDITION 10.1d							
File Edit View Compile Simulate Add Project Tools Layout Bookmarks Window Help							
E - ≥ E ≉ & X ^I	i 📙 🕈 🌳 🏌 🛔 🏔	Layout NoDesign					
ColumnLayout AllColumns	🥰 🛛 X4 🕅 🕅 🕅	🐐 🛛 🗆 🗤 🖉 🕺 🔁 🔝 i 🕪					
♣ ♣ ♠ ♠ ڲ ڲ <i>\$</i> ↓ \$* • • \$* Search:		X ≪ 🕸 🛐 🔲 📕 I 📕 I					
Project - C:/J2/eq	+ 제 × 🗋 c	:/J2/Verilog-HDL/eq.v (/eqSim) - Default 👬 🛃 🔟					
Name Status Type Orde Modified	Ln	#					
eq.v 🗸 Verilog 0 05/19/14 02:10:09 AM	1	<pre>module eq(s, a, b); /* êvoñH */</pre>					
	2	input a, b;					
	3	output s;					
	T III	write ha, hb, sr, sz; assign ± 5 ha = $\sim a$ hb = $\sim b$.					
	Edit	assign #10 s1 = a $\&$ b, s2 = na $\&$ nb					
	Execute	assign #10 s = s1 s2;					
	Compile	endmodule					
	Add to Project	New File					
	Remove from Project	Existing File /* êvoñH */					
Library × 🔛 Project ×	Close Project	Optimization Configuration					
	Update	Simulation Configuration					
VSIM 105 restart _f		Eolder					
VSIM 11> guit -sim	Properties						
	Project Settings						
ModelSim> run -all							
Project : eq <no design="" loaded=""> < <no context=""></no></no>							

別回路のシミュレーション





別回路のシミュレーション

```
module eqSim2;
  wire s;
 reg x, y;
eq2 g(s, x, y);
  initial
    begin
       $monitor( " %b %b", g.s1, g.s2, $stime);
      $display("g.s1 g.s2 time");
         x=0; y=0;
      #50 x=1;
      #50 y=1;
      #50 x=0;
      #50 $finish;
    end
endmodule
```



別回路のシミュレーション

Window内の追加したファイルを選択して、
 Compile→Compile Selected(選択したファイルが対象)または
 Compile All(すべてのファイルが対象)でコンパイル

Moc	delSim AL	TERA STARTER EDITION	10.1d	
File	Edit Vie	w Compile Simulate	Add Source Tools Layout Bookmarks Window Help	
🖹 🗸	🖻 🗑 4	<u>C</u> ompile	② - ₩ 🕮 🖬 ♦ 🕮 🗿 🕺 🕈 🐎 🛊 ≵ 🏊 🏦 Layout NoDesign 💌	
Colun	nnLayout	<u>SystemC Link</u>	·	,
d` .	t t d	Compile <u>A</u> ll	· · · · · · · · · · · · · · · · · · ·	
🔛 Proj	ect - C:/J2/	eq Compil <u>e</u> Selected	🔚 🔜 📩 📩 C:/J2/Verilog-HDL/eq2.v - Default ======= ::::::::====== ::::::::====== ::::::	×
▼Name		Compile Order	Iodified Ln#	
	eq.v	Compile Report	15/19/14 02:10:09 AM 1 module eq2(s, a, b);	
	eq2.v	Compile Summary	15/19/14 02:43:14 AM 2 input a, b;	
			4 wire s1 s2.	
			5 assign $#10 \ s1 = ~(a \ b), \ s2 = ~(a \ b)$	
			$6 \qquad assign #10 s = ~(s1 \& s2);$	
			7 endmodule	
				Ť
	22	Designation of		
	ary × 🔛	Project ×		
A Tran	script ===			×
ModelSi	m> cls			
# 🗆				
ModelSi	m>			Ţ
 			Colo 0 Desirativas (Ale Desira Landada (Ale Castuda)	
		Ln: 1	Coi: U Project : eq <no design="" loaded=""> <no context=""></no></no>	_//.

別回路のシミュレーション

- ・Libraryタグのworkの下に、eq2とeqSim2ができている
- eqSimと同様にeqSim2をシミュレーション

ModelSim ALTERA S	TARTER E	DITION 10.1d					
<u>F</u> ile <u>E</u> dit <u>V</u> iew <u>C</u> ompile <u>S</u> imulate A <u>d</u> d L <u>i</u> brary T <u>o</u> ols Layo <u>u</u> t Boo <u>k</u> marks <u>W</u> indow <u>H</u> elp							
📄 - 🚅 🖬 🛸 🍈	🖹 - 🚰 🎧 参 小 🤱 階 🏶 公 요 ② - 內 臣 🗖 🧇 部 🎧 🛐 🏌 🀎 🏌 🏦 🌦 🤱 Layout NoDesign 🔍						
ColumnLayout AllCol	umns		🥩 🚭 - 🤹 📔 X* 🕅 🔝	🍇 🖬 💿 📧 🛲 🥔 📐 🔩 🕹 ! 🐌			
╡┙┶┙┶╕		े≪ - ≫ि - क्रि• Search:	 \$\$				
🖺 Library 🚃			+ 🗗 🗙 📄 (C:/J2/Verilog-HDL/eq2.v (/eqSim2) - Default 📶 🚊			
▼ Name	Type	Path	- Ln	n#			
- work	Library	work	1	<pre>module eq2(s, a, b);</pre>			
-M eq	Module	C:/J2/Verilog-HDL/eq.v	2	input a, b;			
-M eq2	Module	C:/J2/Verilog-HDL/eq2.v	3	output s;			
eqSim	Module	C:/J2/Verilog-HDL/eq.v	4	wire s1, s2;			
eqSim2	Module	C:/J2/Verilog-HDL/eg2.v	5	assign s1 = ~(a & b), s2 = a b;			
	Library	\$MODEL_TECH//altera/vhdl/220model	6	assign s = ~(s1 & s2);			
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Library	\$MODEL_TECH//altera/verilog/220m	7	endmodule			
	Library	\$MODEL_TECH//altera/vhdl/altera	8				
+	Library	\$MODEL TECH//altera/vhdl/altera I		modulo oggiml.			
		+					
Library × Project	×		<u> «»</u>	eq.v × eq2.v ×			
A Transcript				번 ₫ .			
VSIM 26> quit -sim							
# Compile of eq2.v	was succ	essful.					
ModelSim>				-			
		"eq2.v" was m	odified after it was compiled Ln: 21 Co	Col: 0 Project : eq <no design="" loaded=""></no>			

プロジェクトの変更

- File→Close Projectでプロジェクトを閉じる
- File→New Projectで新しいプロジェクトを作る
 - 前のプロジェクトを閉じていない時は閉じるかどうか聞かれる
- Project Name: eq2で新しいプロジェクトを開く
 - 他のプロジェクトとモジュールを共有するときや、名前が同じ異なるモジュールがなければDefault Library Name: workでよい

	Create Project
	Project Name
M Close Project	Project Location
	C:/J2 Browse
This operation will close the current project. Continue?	Default Library Name
	Work
(はい(Y) いいえ(N)	Copy Settings From 0/modelsim_ase/modelsim.ini Browse
	Copy Library Mappings C Reference Library Mappings
	OK Cancel



 ・プロジェクトeqに戻るときはFile→Resent Projectから選ぶか、 File→Openでファイル選択のポップアップウィンドウの右下 を"Project Files"に切り替えてmpf (Modelsim Project File) ファイルを表示して選択







問題 1

- モジュール eq の動作検証を eqSim を用いて行い,出力結果が 正しくなるかどうか確認せよ。なお,検証には\$monitorを用い, x, y, s をCUI上に表示して確認すること。
- ・また、eqSimを改良し、eqモジュールの内部信号であるs1,s2 を\$monitorでCUI上に表示できるようにせよ。
 \$monitor文において、入れ子となっているモジュールの内部信 号を観測するには"*モジュール名*.*信号名*"を使う、例として、
 p.35のeqSim2の\$monitor文、\$display文を参考にせよ。



入力信号 *a*, *b*, *c*, *d*を受け取り、*a* = *b*と*c* = *d*がともに成り立
 つとき出力信号 *s*を1に、それ以外のとき*s*を0にする回路の
 モジュールを、モジュールeqを2個使って作れ。

問題3

・問題2で設計したモジュールの動作検証を行うテストベンチを 作成せよ.また、作成したテストベンチモジュールを用いて、 動作検証せよ.動作検証は、ModelSimを用いて、入出力信号 の波形をGUI表示して確認すること.



組み合わせ回路と順序回路

・組み合わせ回路:現在の入力のみによって出力が一意に決まる
 ここまで作成してきた回路は全て組み合わせ回路

・順序回路:過去の入力により出力が変化する
 → すなわちデータを記憶している





Verilogでの順序回路作成

- ・レジスタ(reg)とalways文を使うことで、順序回路を作成できる
 - ・レジスタ…値を保存できる極小のメモリ

wireが配線だったのに対し、レジスタ(regとして宣言)は データを格納する箱のイメージ

always文…always@(イベント) 処理内容

"イベントが起こる時,毎回 *処理内容*を実行する"という構文



50サイクル経過する度に ck の値を反転

Ckが下がった瞬間の入力Dが出力Q



Ckが下がる度にその時のDをQ1に保存



Dフリップフロップのテストベンチ

module	dffnSim;
reg	i;
wire	0;
clk	<pre>clk1(ck);</pre>
dffn	dffn1(o, i, ck);
initial	
begin	
\$moi	nitor(" %b %b %b",
	ck,i,o,\$stime);
\$di	<pre>splay("ck i o time");</pre>
	i = 0;
#100	0 i = 1;
#200	0 i = 0;
#100	ð \$finish;
end	-
endmodule	



もう少し複雑な順序回路:1ビットの記憶回路

・制御信号 | でD-FFへの書き込みを制御

```
module r1(q, l, d, ck);

input l, d, ck;

output q;

wire nl, s1, s2, d1;

dffn f(q, d1, ck);

assign nl = ~l;

assign s1 = nl & q, s2 = l & d;

assign d1 = s1 | s2;

endmodule
```



l = 0 のとき q を保持



l = 1 のとき d を書込み



仕様からverilogで回路を作成するには?

- ・前頁のように回路図が与えられれば、もうverilogで書けるはず
- それでは与えられた仕様から回路図を作るには?
 (論理回路の講義の復習)
 - 1. まず仕様から状態遷移図・真理値表を作成
 - 出力と次遷移状態を、入力と現遷移状態の論理式で表す (真理値表からカルノー図を用いて求める)
 - 3. 得られた論理式から回路図を作成
 - 4. 回路図をverilogで書き直す
- ・次ページでは例として、"スイッチaが押された状態でclkが立ち下がると、3クロックの間、1を出力する回路"をverilogで作成する

逐次制御回路1(仕様→状態遷移表・遷移図)

• 仕様:

スイッチaが押された状態でclkが立ち下がると、3クロックの間、1を出力

- ・パラメータ:
 - 入力a
 - 出力b
 - 現在の状態s1s0
 - 次の状態s1's0'



逐次制御回路1(状態遷移図→真理値表)

・得られた状態遷移表,状態遷移図を基に真理値表を描く





真理値表



xはdon't careと言って, 0でも1でもどちらでも良いという意味

don't care でまとめるのが難しいなら a=0, a=1の2行に分けて描いても良い

逐次制御回路1 (真理值表→論理式化)

- ・b, s1', s0' それぞれに関してカルノー図を作成し論理式化
- ・状態s1s0の00 01 11 10という順番に注意
- カルノー図でまとめられる単位は1,2,4…
 なので1×1や1×2,2×2でまとめる



カルノー図(s1″)								
		状態s1s0						
	\searrow	00	01	11	10			
入力	0	0	1	0				
а	1	0	1	0	1			



真理値表



b = s0 + s1

 $s1' = s1 \cdot s0 + s1 \cdot s0$

 $s0' = a \cdot s0 + s1 \cdot s0$

逐次制御回路1 (論理式→回路図・実装)



逐次制御回路2(回路図→状態遷移表→仕様)

- ・順序回路を状態遷移図で表記して動作を解析,あるいは状態 遷移図から順序回路を設計するときにはFFの値が「状態」を 表す
- 入力と出力によってその状態間がどう遷移するかを調べる
 - CS実験第一J1課題の回路の動作を調べる
 - 入力はE, 出力はQ₁Q₀
 - 状態はQ₁Q₀(D-FFから直接なので出力と一致)
 - 2bitなので最大で4状態だが、全ての組合せを取らない場合もある



逐次制御回路2(状態遷移の確認)

<u>状態Q₁Q₀=00の場合</u>

・E=0の時, $Q'_1=0 Q'_0=0$ となる ・E=0の時, $Q'_1=0 Q'_0=1$ となる

<u>状態Q₁Q₀=01の場合</u>

- ・E=1の時, $Q'_1=0 Q'_0=1$ となる ・E=1の時, $Q'_1=1 Q'_0=0$ となる





逐次制御回路2(状態遷移の確認)

<u>状態Q₁Q₀=10の場合</u>

・E=0の時, $Q'_1=1 Q'_0=0$ となる ・E=0の時, $Q'_1=1 Q'_0=1$ となる

<u>状態Q1Q0=11の場合</u>

- ・E=1の時, $Q'_1=1 Q'_0=1$ となる ・E=1の時, $Q'_1=0 Q'_0=0$ となる





逐次制御回路2(真理値表・状態遷移図)



1が入力されるたびに、カウントアップされた値(<4)を出力する回路 だとわかる

逐次制御回路2(論理式の確認)

カルノー図からQ₁, Q₀の式が求められる

真理値表





Е	Q ₁	Q_0	Q' 1	Q' ₀
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

 $Q'_1 = E \cdot \overline{Q}_1 \cdot Q_0 + Q_1 \cdot \overline{Q}_0 + \overline{E} \cdot Q_1$

$$Q'_0 = \overline{E} \cdot Q_0 + E \cdot \overline{Q_0}$$

→ 回路図と照らし合わせて 一致しているかどうか



- ・以下の状態遷移表m2に従って動く回路m2の状態遷移図を書き, 回路を設計せよ.
- ・なお、回路m2は、初期状態0において入力aに1が入力された状態でクロックが立ち下がると、状態1に遷移する。
 以降、同様の操作が行われる度に状態2、3と遷移し、最終的に出力 b から1を出力し、初期状態0に戻る。
- ・回路m2とそのテストベンチを実装し、シミュレーションで動作を確かめよ、ただし、回路m2の状態は、reg ではなく、2個のdffn を用いて実装すること。
- ・上記のシミュレーション結果の何をもって動作確認としたのかも 説明せよ。



状態遷移表 m2





多ビットデータの表現

- ・多ビットのデータは、x[3:0]のように配列で表す(=x[3]x[2]x[1]x[0]の4bit値)
- Parameter 宣言を使うことでビット数を容易に変更できる



ビットベクトル

- •{}で信号を囲むことで複数の信号をまとめることが可能
 - ・例えば, x[3:0]というのは{x[3:2], x[1:0]} {x[3], x[2], x[1], x[0]} と同じ
 - ビットベクトルを使うとシフト操作等が実現できる
 左シフト: {x[2:0], 1'b0} 左巡回シフト: {x[2:0], x[3]}
 右シフト: {1'b0, x[3:1]} 右算術シフト: {x[3], x[3:1]}
 - ※ 1'b0というのは1桁のb(2進数)の0という意味
 - 例えば2進数の11も2'b11と書かないと10進数の11扱いされてしまう

```
module lshift( y, x );
input [3:0] x;
output [3:0] y;
assign y = {x[2:0], 1' b0};
endmodule
```

左1bitシフトのモジュール例

問題5の前提知識1(加算回路の設計)

桁上げ cu=a&b|b&c|c&a 和 s=a&~b&~c|~a&b&~c|~a&~b&c|a&b&c =a^b^c


問題5の前提知識2(加算回路の設計)

・1ビット全加算器の場合(前頁を参考に)

module	fa(cu, s, a, b, ci);
input	a, b, ci;
output	cu, s;
assign	s = a ^ b ^ ci;
assign	cu = a & b b & ci a & ci;
endmodule	





問題5(加算回路の設計)

- 前頁による実装ではnビットの加算器は作れない.なぜなら、faモジュールをn個宣言することがverilogではできないからである.では、 nビット加算器はどうすれば実装できるか?faモジュールを用いずに ビットベクトルを使ってnビット加算器を作成せよ.
- ・このモジュールはadd4 のように、最上位の桁上げcu と、最下位に 加えるci を持つこと。
- さらに、シミュレーションにより評価せよ。

(ヒント)

問題を解く肝になるのは, add4の"fa a3(~), a2(~)…a0(~);"の部分を いかにビットベクトルを用いて書き直すかである. 次ページを参考に 考えてみてください.

- 例:4 bit加算器の場合
- 入力

被加数 x[3:0],加数 y[3:0],最下位の桁上げ ci

• 出力

「加算結果 s[3:0],最上位の桁上げ cu

途中の桁上げ c[3:0] c[3]…3桁目からの桁上げ, c[2]…2桁目からの桁上げ c[1]…1桁目からの桁上げ, c[0]…0桁目からの桁上げ (=ci)



・4つの式をビットベクトルでまとめると?

・4桁からn桁に拡張した場合どういう式になる?



assign 変数= 条件? 条件が1(真) のときの値: 条件が0(偽) のときの値;

•条件はC言語と同じ >,<, >=,<=, ==, !=, <=などが, またその論理積(&&)や論理和(||)などが利用できる





for (制御変数の初期設定; 継続条件; 制御変数の変更) 文;

```
module sel( z, x, c );
    parameter n = 4;
    input [n-1: 0] x;
    input c;
    output [n-1: 0] z;
    assign z = c? x: 0;
endmodule
```

```
module selSim;
reg [3:0] x;
reg c;
wire [3:0] z;
sel sel1(z, x, c);
initial
begin
    $monitor("%b %b %b", x, c, z);
    c = 0;
    for(x=0; x<8; x=x+1) #50 c=~c;
    #50 $finish;
end
endmodule
```

問題6 (n ビットレジスタ)

- クロック同期して動作し、以下に示す仕様を満たすnビットレジスタのモジュールrを設計し、シミュレーションにより評価せよ。
- ・ちなみに1ビットレジスタの例はP.49に載せている.
- ・この問題では、P.49の1ビットレジスタに対して
 ①データのビット幅をnビットに拡張
 ②セレクタを用いて回路をより簡単化 すること
 - ・データ入力: *d, ck* ・制御信号入力: *load*
 - ・データ出力:
 - q

•動作:

 $load = 0 \text{ } \mathcal{O} \mathcal{E} \mathfrak{F} q = q$ $load = 1 \text{ } \mathcal{O} \mathcal{E} \mathfrak{F} q = d$

問題6 (nビットレジスタ)



問題6 (nビットレジスタ)



load=1の時 q = d load=0の時 q = q

問題6 (nビットレジスタ)



load=1の時 q = d load=0の時 q = q

問題7(加減算回路の設計)

- ・問題5の加算回路をベースに、以下の仕様のn桁の加減算回路の モジュールを設計し、シミュレーションにより評価せよ.ただ し負の数は2の補数表現とする.
 - ・数の入力:
 *n*桁の整数*x, y*
 - ・制御信号入力: k
 - ・出力:
 桁上げ*cu*とn桁の和*s*
 - •動作:

k = 0のときs = x + yk = 1のときs = x - y

負数を2の補数で表したとき、x - y = x + y + 1である.した がって, kの値によって $y \ge y$ の何れかを選ぶ選択回路の出 カとxを加算すればよく, 1の加算には、桁上げ入力を使う.





Xilinx VivadoとNEXYS 4を用いて, 加算器(問題5のadd4モジュール)を FPGA上で動作させよ.

以下のURL からファイルをダウンロードし,作成したディレクトリに保存 http://www.hpc.is.uec.ac.jp/yamaki_lab/experiment/add4_nexys4.xdc (コンソールでwget http://www.hpc.is.uec.ac.jp/yamaki_lab/experiment/add4_nexys4.xdc ~/mics_J2/)



Figure 1. Nexys 4 board features

[出典] https://reference.digilentinc.com/_media/nexys:nexys4:nexys4_rm.pdf

add4の実行

・SW7-0の切り替えに応じてLEDが点灯する



FPGAの用途

- FPGA
 - Field Programmable Gate Array
 - ・ユーザが論理機能を何度でも書き換え可能
 - ・ 少量多品種 ・ 短ライフサイクル品
- ASIC
 - Application Specific Integrated Circuit
 - ・回路の論理が固定の特定アプリケーション用のカスタム品
 - ・ プロセッサ等の大量生産品



xdcファイル:FPGAのピン配置を決めるファイル まずはFPGAボードを確認

#LED vivado 計算結果を出力するLED(1だと点灯、0だと無灯)

set_property PACKAGE_PIN R18 [get_ports {cu}] set_property IOSTANDARD LVCMOS33 [get_ports {cu}] set_property PACKAGE_PIN N14 [get_ports {s[3]}] set_property IOSTANDARD LVCMOS33 [get_ports {s[3]}] set_property PACKAGE_PIN J13 [get_ports {s[2]}] set_property IOSTANDARD LVCMOS33 [get_ports {s[2]}] set_property PACKAGE_PIN K15 [get_ports {s[1]}] set_property IOSTANDARD LVCMOS33 [get_ports {s[1]}] set_property IOSTANDARD LVCMOS33 [get_ports {s[1]}] set_property PACKAGE_PIN H17 [get_ports {s[0]}] set_property IOSTANDARD LVCMOS33 [get_ports {s[0]}]

#PUSH BUTTON
set_property PACKAGE_PIN N17 [get_ports {ci}]
set_property IOSTANDARD LVCMOS33 [get_ports {ci}]

xdcファイル:FPGAのピン配置を決めるファイル まずはFPGAボードを確認

#SWITCH vivado 4ビットの値を入力するスイッチ

set_property PACKAGE_PIN R13 [get_ports {x[3]}] set_property IOSTANDARD LVCMOS33 [get_ports {x[3]}] set property PACKAGE PIN U18 [get ports {x[2]}] set property IOSTANDARD LVCMOS33 [get ports {x[2]}] set property PACKAGE PIN T18 [get ports {x[1]}] set property IOSTANDARD LVCMOS33 [get ports {x[1]}] set property PACKAGE PIN R17 [get ports {x[0]}] set property IOSTANDARD LVCMOS33 [get ports {x[0]}] set property PACKAGE PIN R15 [get ports {y[3]}] set property IOSTANDARD LVCMOS33 [get ports {y[3]}] set property PACKAGE PIN M13 [get ports {y[2]}] set property IOSTANDARD LVCMOS33 [get ports {y[2]}] set_property PACKAGE_PIN L16 [get_ports {y[1]}] set property IOSTANDARD LVCMOS33 [get ports {y[1]}] set property PACKAGE PIN J15 [get ports {y[0]}] set property IOSTANDARD LVCMOS33 [get ports {y[0]}]

Vivadoを起動

- Terminal で vivado と入力
- Create New Projectをクリック



Quick Start

Create Project > Open Project > Open Example Project >

Tasks

Manage IP > Open Hardware Manager > Xilinx Tcl Store >

Learning Center

Documentation and Tutorials > Quick Take Videos > Release Notes Guide >



プロジェクトディレクトリを指定

- Project name: add4
- Project location: 任意のディレクトリ
- Create project Subdirectoryにチェック

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.



<u>P</u> roject name:	add4				\otimes
Project <u>l</u> ocation	: /home3/staff/sa002394/.ced_centos/mic	s_J2/			···
🗸 Create projec	ct subdirectory				
Project will be c	reated at: /home3/staff/sa002394/.ced_cen	tos/mics_J2/add4			
?		< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

プロジェクトタイプの指定

RTLプロジェクトにチェック

Project Type

Specify the type of project to create.



<u>R</u>TL Project

?

You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

<u>D</u>o not specify sources at this time

<u>Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.</u>

Do not specify sources at this time

- Imported Project
 Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project Create a new Vivado project from a predefined template.

< <u>B</u>ack

Next >

<u>F</u>inish

Cancel

ソースファイルの指定

Add filesで次のファイルを追加 add4.v, fa.v

	Index	Name	Li rary	HDL Source For	Location
Э	1	add4.v	xi defaultlib	Synthesis & Simulation	 /home3/staff/sa002394/.ced_centos/mics_J2/samp
9	2	fax	xi defaultlib	Synthesis & Simulation	 /home3/staff/sa002394/.ced_centos/mics_J2/sample
			<u>A</u> d	ld Files Add Direct	ries
5	an and a	eld RTL jnc		ld Files Add Direct	ries
See	an and a	edd RTL jnc	ad lude files into pro oject	ld Files Add Direct oject	oriesCreate File
) Se) Co) Ad	can and a opy <u>sour</u> dd sogre	edd RTL jne ces into pr es from sul	Lude files into pro oject odirectories	ld Files Add Direct oject	rries
Sector Adv	can and a opy gour dd sogro et langu	dd RTL (no ces into pr es from su age: Veri	lude files into pro oject bdirectories log ❤ Si	id Files Add Direct oject mulator language: Mixed	ories ⊆reate File

制約ファイルを追加

Add filesで次のファイルを追加 add4_nexys4.xdc

+ - +	\$
Constraint File	Location
add4_nexys4.xdc	/ced-home/staff/MICS1_J2
	Add Files Create File
Cogy constraints	i files into project
(v)	< Back Next > Einish Cancel

デバイスの指定: XC7A100T-CSG324を選択

Family: Artix-7 Package: csg324 Speed grade: -3

	Et category:	AJI	Ý		Spee	e <u>d</u> grade:	-3		~	
Eamily:		Artix-7		✓ Iemp grad		p grade:	All Re	maini	×	
Packa	ge:	csg324								
				Reset A	All Filters					
earch:	Q		×							
Part		I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transce
🕒 xc7a1	Stcsg324-3	324	210	10400	20800	25	0	45	0	0
🕖 xc7a3	55tcsg324-3	324	210	20800	41600	50	0	90	0	0
🕒 xc7a5	50tcsg324-3	324	210	32600	65200	75	0	120	0	0
🕖 xc7a7	75tcsg324-3	324	210	47200	94400	105	0	180	0	0
🕒 xc7a1	100tcsg324-3	324	210	63400	126800	135	0	240	0	0

設定内容を確認

VIVADO.	New Project Summary
HLx Editions	1 A new RTL project named 'add4' will be created.
	1 2 source files will be added.
	1 constraints file will be added.
	The default part and product family for the new project: Default Part: xc7a100tcsg324-3 Product: Artix-7 Family: Artix-7 Package: csg324 Speed Grade: - 3
E XILINX	To create the project, click Finish
3	Einish Cancel

Vivadoプロジェクトトップページ

<u>F</u> ile <u>E</u> dit F <u>l</u> ow <u>T</u> ools <u>W</u> indow	Layout View Help Qr Quick Access									Read	iy
🖻 🛧 🔺 🖬 🖿 🗙 🕨 I	¤ ✿ Σ ≤≤ ∅ ¥									😬 Default Layout	~
Flow Navigator — — — — — — — — — — — — — — — — — — —	PROJECT MANAGER - add4										?:
V PROJECT MANAGER	Sources 2 – 🗆 🖞 🗙	Project Summary								200	1 X
 Settings Add Sources Language Templates PI P Catalog IP INTEGRATOR Create Block Design Open Block Design 	Q ₹ ↓ ↓ 0 ↓ Q ₹ ↓ ↓ 0 ↓ V Design Sources (4) ↓ ↓ Ø @dd4 (add4.v) ↓ @ eor4 (add4.v) @ eor4 (add4.v) @ eor1 (add4.v) Ø noth (add4.v) Ø noth (add4.v) > Constraints (1) > Simulation Sources (4)	Project summary Settings Edit Project location: Project location: Project family: Project part: Top module name: Target language:	add4 /home3/staff/sa0023 Artix-7 xc7a100tcsg324-3 add4 Verilog	94/.ced_centos/mic	s_J2/add4						
Generate Block Design		Simulator language	e: Mixed								
✓ SIMULATION		Synthesis					1	mplementation			
Run Simulation • RTL ANALYSIS • Open Elaborated Design	Hierarchy Libraries Compile Order Source File Properties ? _ □ ◻ ×	Status: Messages: Part: Strategy: Report Strategy:	Not started No errors or warnings xc7a100tcsg324-3 Vivado Synthesis Default Vivado Synthesis Default	s Reports				Status: Messages: Part: Strategy: Report Strategy:	Not started No errors or warnings xc7a100tcsg324-3 Vivado Implementation Defaults Vivado Implementation Default Reports		
✓ SYNTHESIS▶ Run Synthesis	veradid4.v v veradio							Incremental compile:	None		
> Open Synthesized Design	Location: /home3/staff/sa002394/.ced_centos/mics_J	DRC Violations						Timing			
 IMPLEMENTATION Run Implementation 	Type: Verilog Library: xil_defaultlib		Run Implementati	on to see DRC result	5				Run Implementation to see timing r	results	
> Open Implemented Design	Size: 0.8 KB	Utilization					1	Power			
PROGRAM AND DEBUG Generate Bitstream Open Hardware Manager	Modified: Wednesday 16/06/01 03:46:22 PM		Run Synthesis to	see utilization result	S				Run Implementation to see power r	esults	
7 Open hai dware Manager	Tcl Console Messages Log Reports Design Runs	×								? _ [0 0
	Q ★ ♦ ♦ >> + % Name Constraints Status WNS TNS WH ∨ > synth_1 constrs_1 Not started ▷ impl_1 constrs_1 Not started	S THS TPWS To	tal Power Failed Routes	LUT FF BRAMs	URAM	DSP Star	t Elapsed	Run Strategy Vivado Synthesis De Vivado Implementati	faults (Vivado Synthesis 2017) on Defaults (Vivado Implementation 2017)	Report Strategy Vivado Synthesis Default Reports Vivado Implementation Default Rep	(Viva

コンパイルの実行

Sources ウィンドウで add4 を選んで Generate Bitstream を実行し Yes をクリック

Flow Navigator — — — — — — — — — — — — — — — — — — —	PROJECT MANAGER - add4	
✓ PROJECT MANAGER	Sources 2	Project Summary
🔅 Settings		
Add Sources		Settings Edit
Language Templates	Design Sources (4) Add4 (add4 v) (4)	Project name: add4
👎 IP Catalog		Project location: /home3/staff/sa002394/.ced_centos/mics_J2/add4
	eorn (add4.v)	Product family: Artix-7
✓ IP INTEGRATOR	😡 notn(add4.v)	Project part: xc7a100tcsg324-3
Create Block Design	Constraints (1)	Top module name: add4
Open Block Design	Simulation Sources (4)	Target language: Verilog
Generate Block Design		Simulator language: Mixed
		for the state of the
✓ SIMULATION		Synthesis
Run Simulation		Status: Not started St
	Hierarchy Libraries Compile Order	No Implementation Results Available × Me
Open Elaborated Design		Pa
y open claborated besign	Source File Properties ? _ D	There are no implementation results available. OK to launch synthesis and implementation? Generate Bitstream' will automatically start when synthesis and implementation completes. Between the synthesis and implementation completes.
✓ SYNTHESIS	œ add4.v ← ⇒	Don't show this dialog again
Run Synthesis	✓ Enabled	
> Open Synthesized Design	Location: /home3/staff/sa002394/.ced_centos/m	<u>Y</u> es <u>N</u> o Ti
	Turner Monthead	
✓ IMPLEMENTATION	Type: verilog	Run Implementation to see DRC results
Run Implementation	Library: xil_defaultlib	
> Open Implemented Design	Size: 0.8 KB	Utilization Po
	Modified: Wednesday 16/06/01 03:46:22 PM	Run Synthesis to see utilization results
✓ PROGRAM AND DEBUG		* >
Generate Bitstream	General Properties	
> Open Hardware Manager		

コンパイルの実行

OKをクリック

Launch Runs						
Launch the selected synthesis or impl	ementation runs.	À				
Launch <u>d</u> irectory: 🛜 <default lau<="" th=""><th>unch Directory></th><th>~</th></default>	unch Directory>	~				
Options						
● <u>L</u> aunch runs on local host:	Number of jobs: 4	*				
○ Launch <u>r</u> uns on remote hosts	Configure <u>H</u> osts					
Caunch runs using LSF	Con <u>f</u> igure LSF					
○ <u>G</u> enerate scripts only						
Don't show this dialog again	ОК	Cancel				



Bitstream Generation successfully completed が でたら成功



FPGAの電源を入れる



FPGAへ書き込み









Program Device

×

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstre <u>a</u> m file:	aff/sa002394/.ce	d_centos/r	nics_J2/add4/a	dd4.runs/imp	_1/add4.I🚫
Debug probes file:					
✓ Enable end of s	tartup check	_			
			Program を	クリック	
?				<u>P</u> rogram	Cancel

add4の動作確認 例:0011+0001=0100

