目 次

第1章　2週目 ツールを用いた回路設計技術

1.1　目的

1週目では紙の上で回路設計を行ったが，巨大なシステム，例えばGPU[1]などを紙上で設計することは困難であり，コンピュータを用いて設計することが一般的である．2週目では，近年の回路設計技術に対する理解を深めるために，Windowsツール，Xilinx ISE WebPACKを用いて回路設計を行う．また，回路シミュレータ上で動作検証を行う．

1.1.1　Xilinx ISE WebPACKについて

一般に集積回路の制作には時間がかかり，また大量生産を行わない限りコストもかかるため，少量生産を行う際はFPGA[2]など論理回路をエミュレートできる回路が使用される．Xilinxは FPGA の大手メーカであり，ISE WebPACK は Xilinx が提供する．

1.2　実験方法

シミュレーションで使用するデバイスは Spartan3E ファミリの XC3S100E の CP132 パッケージである．Xilinxの回路図ベースで実験を進めていく．

1.2.1　課題1 3入力多数決回路

3入力多数決回路の回路を作成する．真理値表を表1.1に示す．設計した回路図を1.1に示す．シミュレーションのテストベンチを付録のソースコード1.1に示す．

1.2.2　課題2 発展課題 10進7セグデコーダの制作

7セグメントデコーダを作成する．I_0 から I_3 の入力で7セグメント LED のセグメントの出力を得るようにする．設計する回路は，正論理を仮定すると，アノードコモンの7セグメント LED は0のときに光るようにする．そのため真理値表は表1.2のとおりである．カルノー図を半自動生成するために Excel を用いて表を生成している (図1.3)．シミュレーションのテストベンチを付録のソースコード1.2に示す．

[1] Graphics Processing Unit の略
[2] Field Programmable Gate Array の略

表 1.1: 多数決回路の真理値表

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

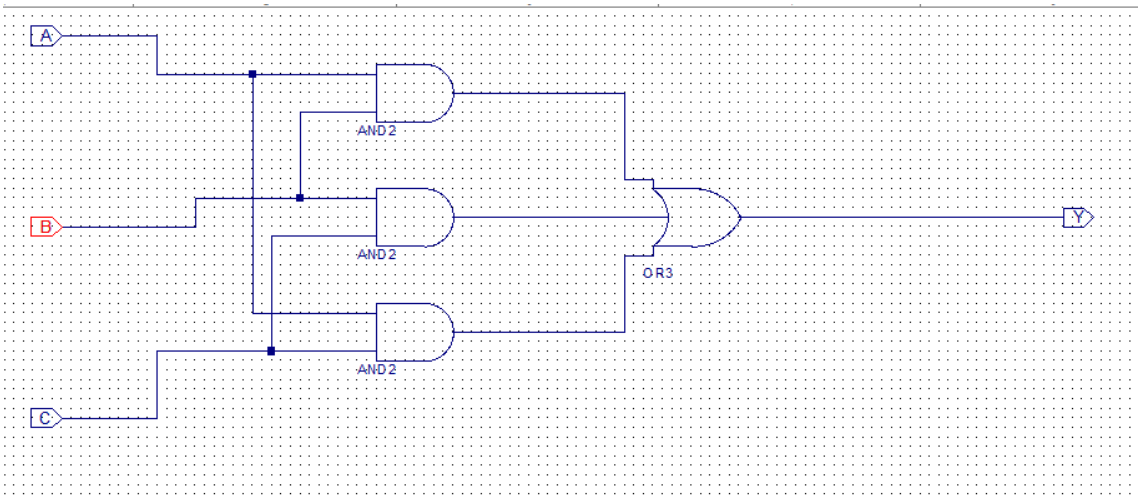


図 1.1: 3 入力多数決回路の回路図

1.2.3 課題 3 2 入力 AND ゲートの FPGA ボードでの動作確認

2 入力 AND ゲートを実際に作成し，その入力を押しボタンスイッチ，出力を LED に接続し，入出力を割り当て，コンフィグレーションデータを作成する．作成したデータを

表 1.2: 7 セグデコーダの真理値表

N	I_3	I_2	I_1	I_0	A	B	C	D	E	F	G
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	0	0	1
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0

FPGA にダウンロードし，動作検証をする．回路図を図 1.4 に示す．ピンの a 割り当てを図 1.5 に示す.

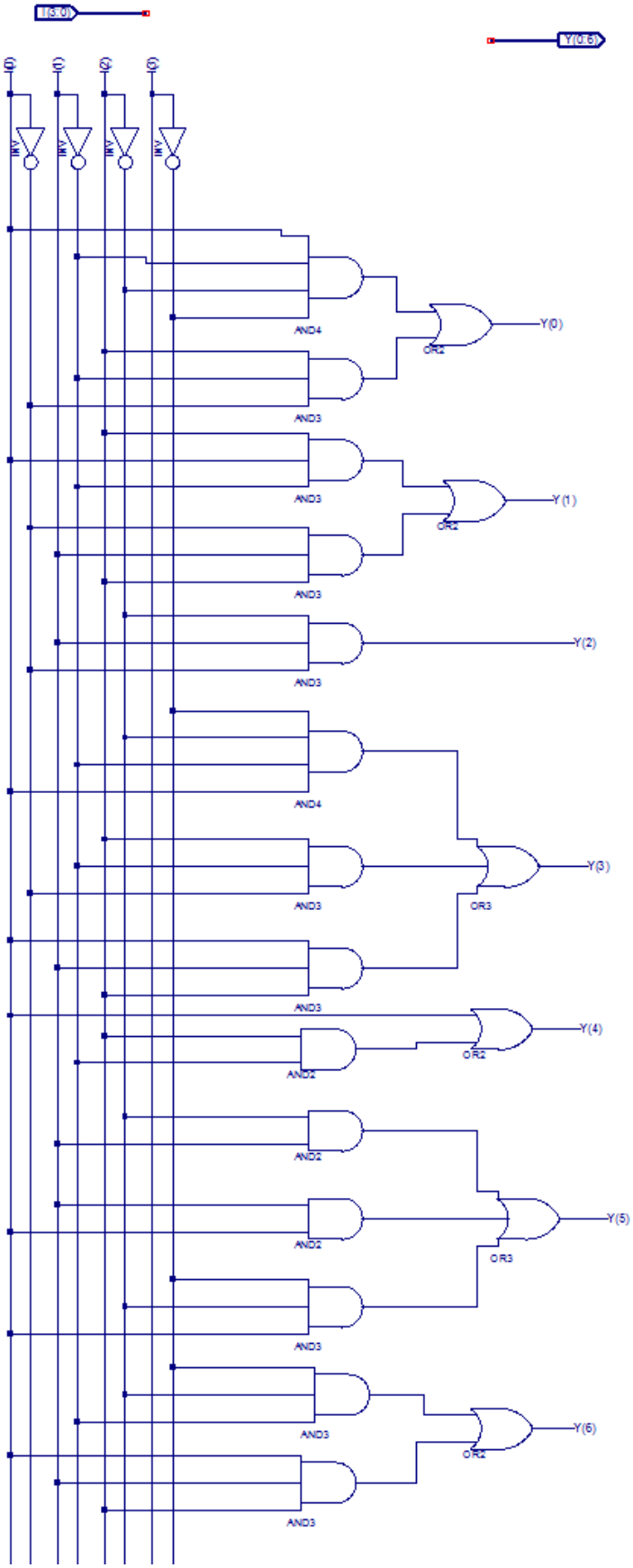
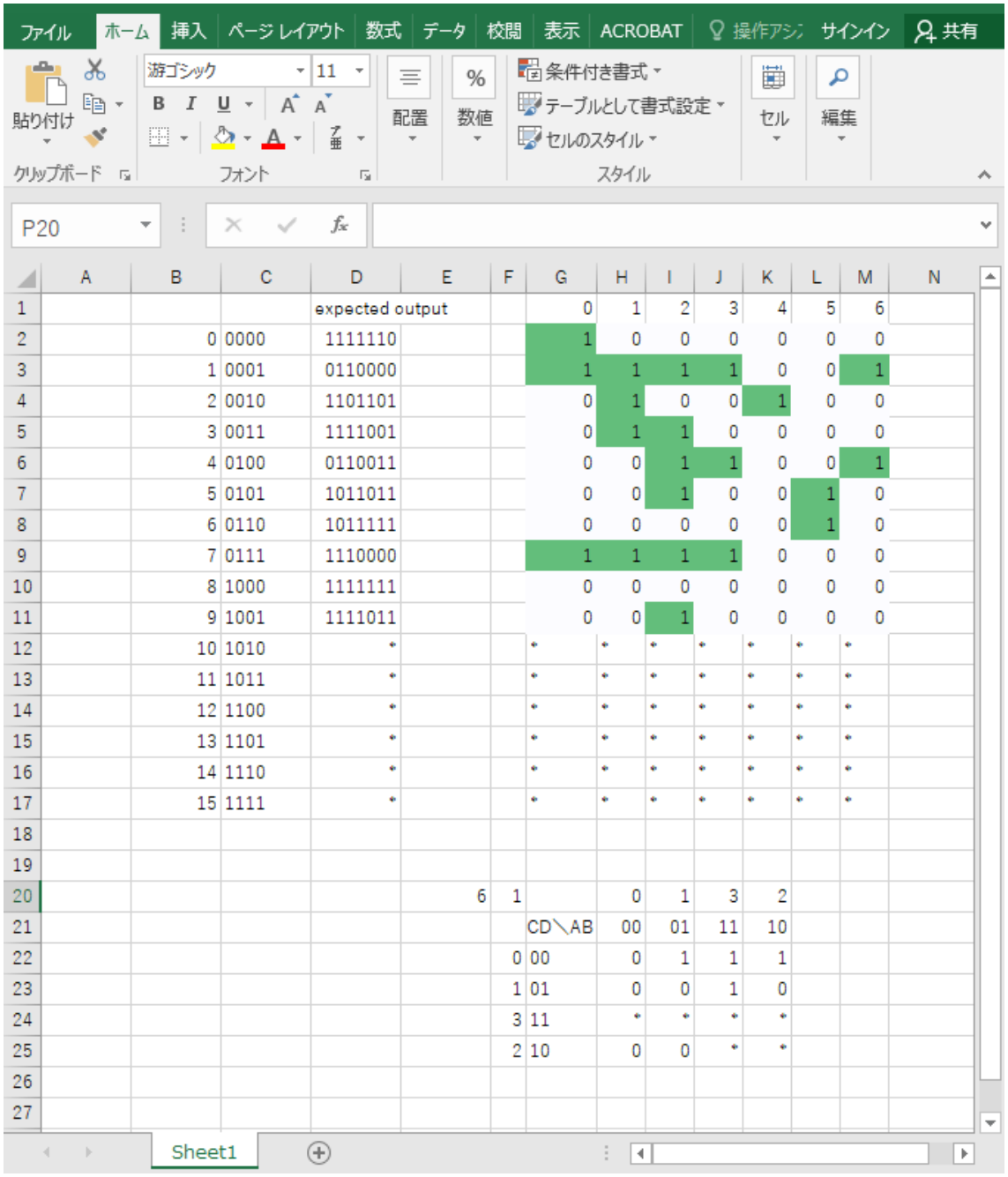


図 1.2: 7 セグデコーダの回路図



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1				expected output			0	1	2	3	4	5	6	
2		0	0000	1111110			1	0	0	0	0	0	0	
3		1	0001	0110000			1	1	1	1	0	0	1	
4		2	0010	1101101			0	1	0	0	1	0	0	
5		3	0011	1111001			0	1	1	0	0	0	0	
6		4	0100	0110011			0	0	1	1	0	0	1	
7		5	0101	1011011			0	0	1	0	0	1	0	
8		6	0110	1011111			0	0	0	0	0	1	0	
9		7	0111	1110000			1	1	1	1	0	0	0	
10		8	1000	1111111			0	0	0	0	0	0	0	
11		9	1001	1111011			0	0	1	0	0	0	0	
12		10	1010	*			*	*	*	*	*	*	*	
13		11	1011	*			*	*	*	*	*	*	*	
14		12	1100	*			*	*	*	*	*	*	*	
15		13	1101	*			*	*	*	*	*	*	*	
16		14	1110	*			*	*	*	*	*	*	*	
17		15	1111	*			*	*	*	*	*	*	*	
18														
19														
20					6	1		0	1	3	2			
21							CD＼AB	00	01	11	10			
22						0	00	0	1	1	1			
23						1	01	0	0	1	0			
24						3	11	*	*	*	*			
25						2	10	0	0	*	*			
26														
27														

図 1.3: 7セグデコーダの設計に用いたシート, 画像では6のカルノー図が出力されている

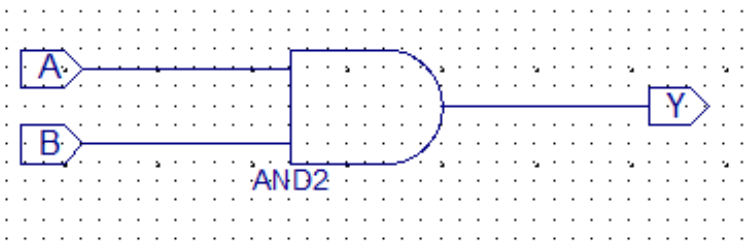


図 1.4: AND ゲートの回路図

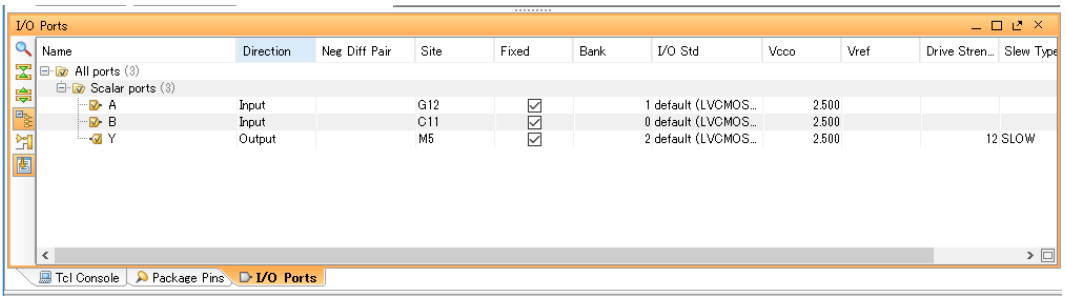


図 1.5: AND ゲートのピンアサイン

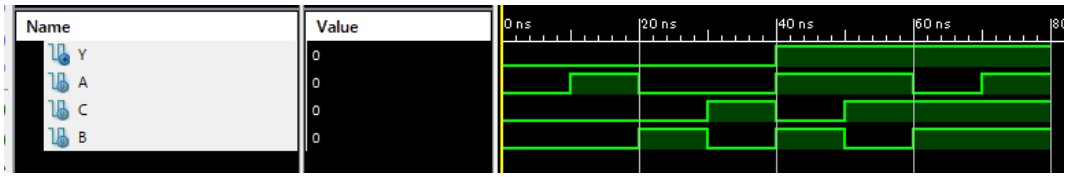


図 1.6: 3 入力多数決回路のシュミュレーション結果

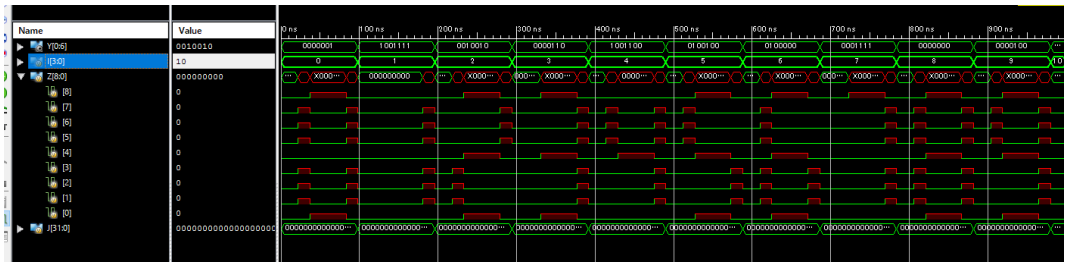


図 1.7: 7 セグメントデコーダのモジュールテスト

1.3 実験結果

1.3.1 課題 1 3 入力多数決回路

シュミュレーション結果を図 1.6 に示す.

1.3.2 課題 2 発展課題 10 進 7 セグデコーダの制作

シュミュレーション結果を図 1.7 に示す．テストベンチの結果の中で，赤で引かれた線に注目すると，7 セグのデコード結果が見える.

1.3.3 課題 3 2 入力 AND ゲートの FPGA ボードでの動作確認

動作確認をしている様子を図 1.8 に示す．2 つのボタンを 0 個または 1 個押したときには LED は点灯せず，両方のボタンを押したときのみ LED が点灯した.

1.4 考察

1.4.1 課題 1 3 入力多数決回路

3 入力多数決回路について，予想通りの動作ができていることがわかる．この回路で，信頼性が求められる計算の正確性が確保できると考えられる.

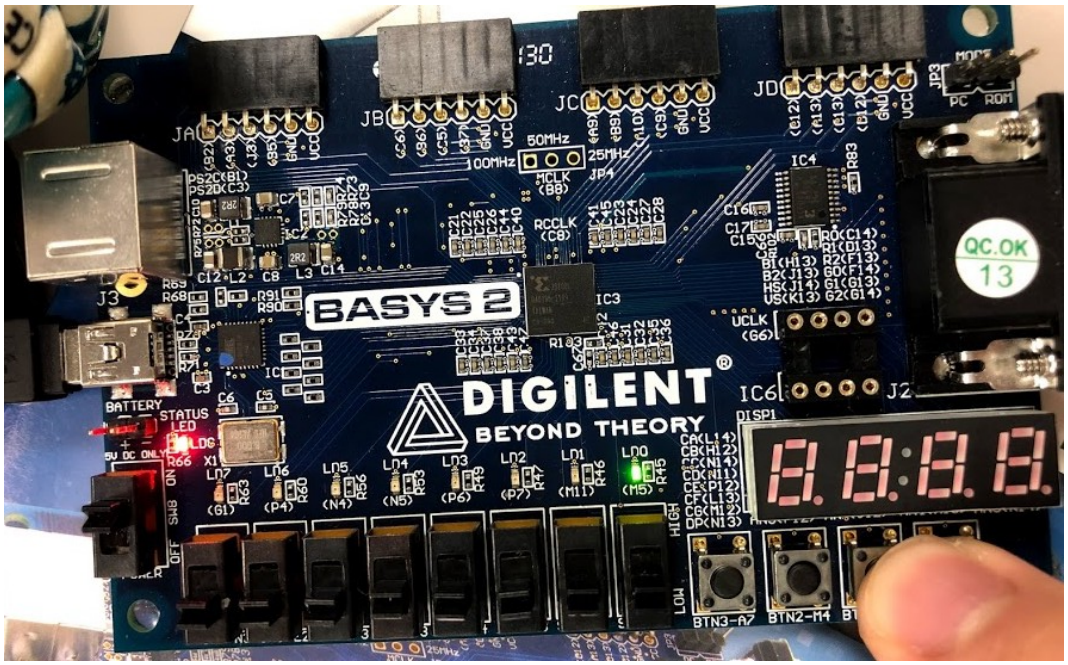


図 1.8: 2 入力 AND の LED への出力テスト

1.4.2 課題 2 発展課題 10 進 7 セグデコーダの制作

7 セグメントデコーダも，波形から正しくデコードできている様子がわかる．0 で光るように設計したため，AND ベースで設計する際に論理素子が少なくて済んだ．これも，光る部分の設計より，光らない部分を考えたほうが回路が簡略化しやすいことが理由としてある．

1.4.3 課題 3 2 入力 AND ゲートの FPGA ボードでの動作確認

2 入力 AND 回路について，想定通り両方のボタンを押したときのみ LED が光った．これは，スイッチが押されたら 1，LED も 1 のときに光るように設計されているためである．

1.5 感想

コンピュータ上での回路設計に触れることができ，基本的な FPGA の特性をつかむことができた．特に実際に回路をブレッドボードと論理ゲートで組むことなく，複雑な回路のエミュレートが実機レベルで開発できる点から，FPGA は有用だと思う．

7 セグメントデコーダの制作では，どのセグを光らせるのか，迷う部分があった．例えば 1, 6, 7, 9 などは光らせ方が複数あり[3]，それぞれの数字において，発光パターンが複数考えられたが，解説に揃えて設計することにした．後の検算のことを考えてのことである．

あと，レポート作成時に Tex で表作るのが意外と大変だった．Excel 等で作った表を変換するソフトを用意すべきだと思った．

[3]数字”1” は右左の違い，6, 7, 9 はそれぞれ A，F，D セグメントの発光の有無に迷う

1.6 付録

ソースコード 1.1: TMR のテストベンチ

```
1  ‘timescale 1ns / 1ps
2
3  //
       //////////////////////////////////////////////////////////////////////////////////
4  // Company:
5  // Engineer:
6  //
7  // Create Date: 13:19:06 04/24/2019
8  // Design Name: ADDER
9  // Module Name: Z:/adder/testbench.v
10 // Project Name: adder
11 // Target Device:
12 // Tool versions:
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: ADDER
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 //
       //////////////////////////////////////////////////////////////////////////////////
24
25 module testbench;
26
27         // Inputs
28         reg A;
29         reg B;
30         reg C;
31
32         // Outputs
33         wire S;
34         wire Co;
35
36         // Instantiate the Unit Under Test (UUT)
37         ADDER uut (
38                 .A(A),
39                 .B(B),
40                 .C(C),
41                 .S(S),
42                 .Co(Co)
43         );
44
45         initial begin
46                 // Initialize Inputs
47                 A = 0;
48                 B = 0;
49                 C = 0;
50
51                 // Wait 100 ns for global reset to finish
52                 #10;
```

```
53                  // Initialize Inputs
54                  A = 0;  B = 0;  C = 0;  #10;
55                  A = 1;  B = 0;  C = 0;  #10;
56                  A = 0;  B = 1;  C = 0;  #10;
57                  A = 1;  B = 1;  C = 0;  #10;
58                  A = 0;  B = 0;  C = 1;  #10;
59                  A = 1;  B = 0;  C = 1;  #10;
60                  A = 0;  B = 1;  C = 1;  #10;
61                  A = 1;  B = 1;  C = 1;  #10;
62
63                  // Add stimulus here
64
65          end
66
67  endmodule
```

ソースコード 1.2: 回路図で設計した 7 セグデコーダのテストベンチ

```
1   // Verilog test fixture created from schematic Z:\sevenseg\sevensegdec.sch - Wed Apr
        17 16:46:36 2019
2
3   `timescale 1ns / 1ps
4
5   module sevensegdec_sevensegdec_sch_tb();
6
7   // Inputs
8       reg [3:0] I;
9
10  // Output
11      wire [0:6] Y;
12          reg [8:0] Z;
13          integer J;
14
15  // Bidirs
16
17  // Instantiate the UUT
18      sevensegdec UUT (
19                  .I(I),
20                  .Y(Y)
21      );
22  // Initialize Inputs
23  initial begin
24          I = 0;
25          for(J=0; J<16; J=J+1) begin
26                  I <= J;
27                  Z <= 0;
28                  #22;
29                  Z <= {1'b0, {3{Y[5] ? 1'b0 : 1'bx}}, 1'b0,
30                                          {3{Y[4] ? 1'b0 : 1'bx}}, 1'b0}; // c (E, F)    \
31                      #15
32
33                  Z <= {Y[0] ? 1'b0 : 1'bx, 3'b000, Y[6] ? 1'b0 : 1'bx,
34                                          3'b000, Y[3] ? 1'b0 : 1'bx}; // (A, D, G)    \
35                      #46
36
37                  Z <= {1'b0, {3{Y[1] ? 1'b0 : 1'bx}}, 1'b0,
38                                          {3{Y[2] ? 1'b0 : 1'bx}}, 1'b0}; // Ec (B, C)
                                                \
39                      #15
40
41                  ;
```

```
42          end
43           Z<= 0;#20
44           $finish;
45  end
46
47  endmodule
```

第2章　3週目 Verilog を用いた回路設計技術

2.1　目的

2 週目の実験では，コンピュータ上で回路図を設計した．しかし大規模な回路の制作に回路図は向いていない．多くの場合，ハードウェア記述言語 (HDL; Hardware Description Language) を用いて記述される．3 週目は HDL の 1 種である Verilog を用いて回路設計を行う．Verilog は C 言語と似た言語であり，C 言語習得者にとっては感覚的に扱いやすい言語である．

2.2　実験方法

2.2.1　課題 1 全加算器

半加算器，全加算器を Verilog で記述する．ただし手続きブロックと算術演算子を用いてはならない．ここで全加算器は，A, B, C_I の入力の和を求める回路である．全加算器の真理値表を 2.1 に示す．

表 2.1: 全加算器の真理値表

A	B	C_I	C_O	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

ソースコードをソースコード 2.1 に示す．テストベンチをソースコード 2.2 に示す．

2.2.2　課題 2 について

課題 2 は選択問題だが，実験時間が余ったため，両方とも実験を行った．両方ともレポートにまとめておく．

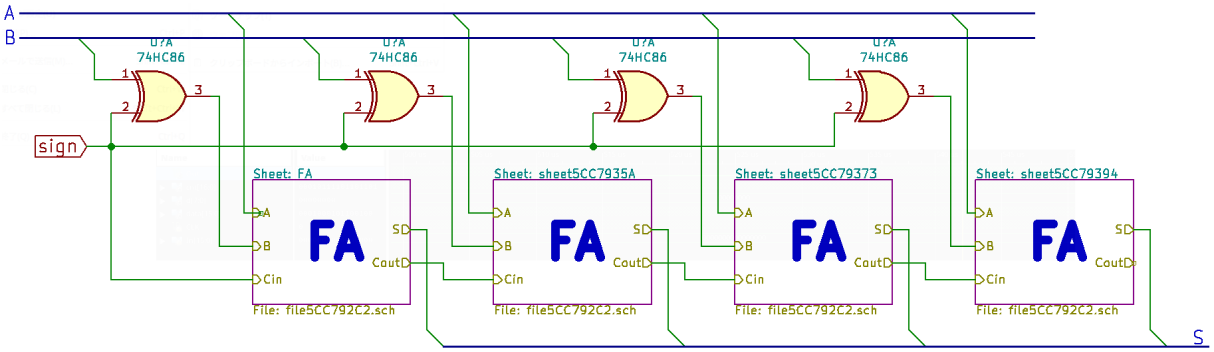


図 2.1: 加減算器の構成

2.2.3 課題 2A 4 ビット加減算器

4 ビット加減算器を作成する．加減算機とは，制御入力の値に応じ，加算器または減算器となる回路である．減算は，$S = A - B$ などという式に対し，2 の補数表現を用いて $C = -B = (B \oplus 4'hF) + 1$ と変形することで，$S = A + C$ という式で記述することができる．そのため，通常の加算器に対し，XOR を通して符号を反転させ，全加算器の 0 ビット目のキャリービットを 1 にすることで，減算器を実現することができる．図 2.1 に回路図を示す．図中，FA は全加算器という意味である．実際には，最終ビットのキャリービットの出力もモジュールから出力させる．

ソースコードをソースコード 2.3 に示す．テストベンチをソースコード 2.4 に示す．

2.2.4 課題 2B 4 ビット乗算器

符号なし 4 ビット乗算器を制作する．4 ビットの掛け算の積は 8 ビットとなる．2 進数での掛け算は，10 進法での筆算と同じように行う (表 2.2)．

表 2.2: 2 進法での筆算例

				1	0	1	0	
			×	0	1	0	1	
				1	0	1	0	(1)
			0	0	0	0		(0)
		1	0	1	0			(1)
+	0	0	0	0				(0)
	0	1	1	0	0	1	0	

具体的には，$C = A \times B$ の演算に対して，A を左にシフトしつつ，B の各ビットで全体をマスクし，最後に合算する．筆算例と HDL で記述されたソースコードを合わせて見ながら理解してもらいたい．ソースコードをソースコード 2.5 に示す．テストベンチをソースコード 2.6 に示す．

2.2.5 課題3 手続きブロックを用いた7セグメントデコーダの作成

2週目の課題にあった7セグメントLED用のデコーダを手続きブロックを用いて記述する．ソースコードをソースコード2.7に示す．テストベンチをソースコード2.8に示す．

2.2.6 課題4 7セグメントLEDへの出力と発光の関係の観察

7セグメントLEDの光り方を調べるため，7セグメントLEDとスイッチをバッファで連結し，光り方を観察する．

ピンの割り当ては，詳細は付録にまとめておく(ソースコード2.9)．ボタンとLEDへの出力の光らせ方も，ソースコード2.10で記述している．

7セグメントLEDはアノードコモンである．カソード側はAセグメントからG,dotのLEDに繋がっていて，4桁すべて同じ端子を共有している．プッシュスイッチ側スイッチはプルダウン抵抗を通して何も押されない状態では0出力となる．IOの割り当てでは，スライドスイッチをLEDのカソード側8ビットに割り当てをした．プッシュスイッチは，アノード側に4桁分配置した．

7セグメントLEDの光り方を調べる前に期待される動作を考える．7セグメントLEDのアノード側はLow状態で光る状態の準備ができる．カソード側はHigh状態になることで光る準備ができ，両方が成立したセグメントのLEDが発光することが期待できる．

そのため，まずはプッシュボタンは何も押さず，すべての桁を選択した状態でテストを開始する．スライドスイッチとLEDの発光の関係を調査する．その次にスライドスイッチはすべてHighの状態にし，スライドスイッチを1つずつ押して，桁が無効になるかを観察する．

2.2.7 課題5 7セグメントLED表示回路の作成

7セグメントLEDへ実際の数値のの出力をテストする．押しボタン3つを3ビットの入力とし，それに応じた数字をすべての桁の7セグメントLEDに出力する回路を作成する．

7セグメントLEDとスイッチのピン割り当てをソースコード2.11に示す．その他の7セグメントLEDの表示ソースコードを2.12に示す．

2.2.8 課題6 乗算における符号について

4ビット乗算器について，符号ありと符号なし両方の演算結果の違いを観測する．ソースコードをソースコード2.13に示す．テストベンチをソースコード2.14に示す．

2.3 実験結果

2.3.1 課題1 全加算器

シミュレーション結果の波形を図2.2に示す．

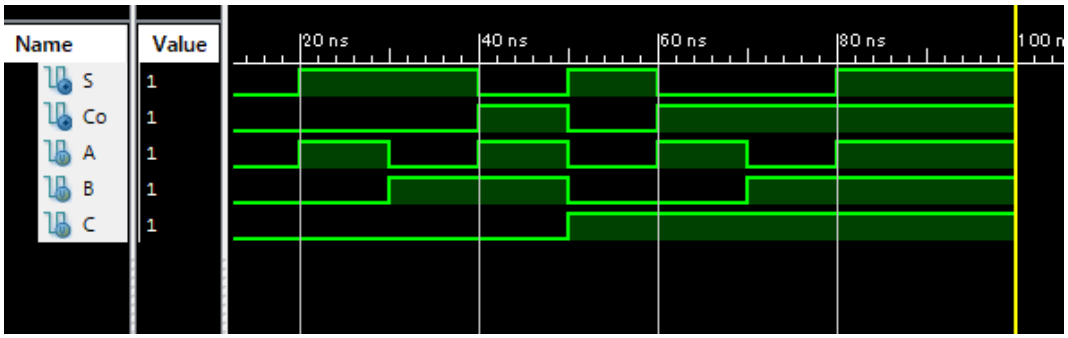


図 2.2: 全加算器のモジュールテスト

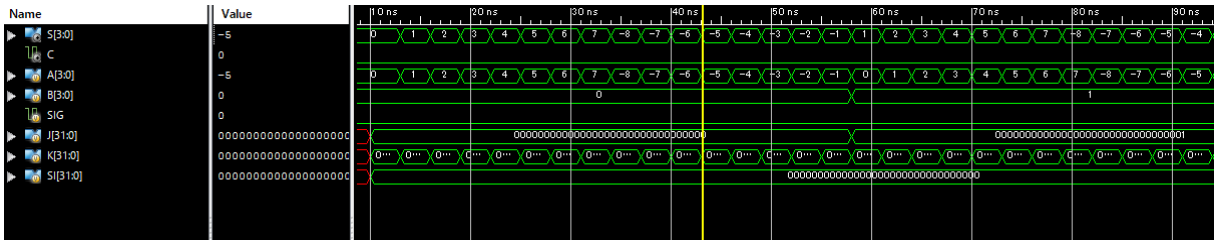


図 2.3: 加減算器の加算時のモジュールテスト

2.3.2 課題 2A 4 ビット加減算器

B の符号を正のまま，つまり加算中のシミュレーション結果の 1 部の波形を図 2.3 に示す．B の符号を負にし，減算中のシミュレーション結果の 1 部の波形を図 2.4 に示す．波形中，SIGN が符号を示しており，すべてのデータは符号ありと仮定して表示している．出力 C は桁溢れを示している．図中，変数 J, K, SI はループ変数で特に意味はない．

2.3.3 課題 2B 4 ビット乗算器

4 ビット乗算器のモジュールテストの結果の 1 部を図 2.5 に示す．図中，変数 J, K はループ変数で特に意味はない．

2.3.4 課題 3 手続きブロックを用いた 7 セグメントデコーダの作成

7 セグデコーダのモジュールテストの結果を図 2.6 に示す．2 週目の課題同様に赤の線に注目するとデコード結果が見える．

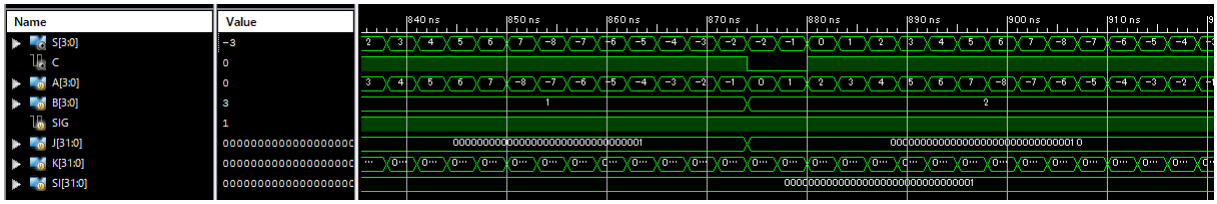


図 2.4: 加減算器の減算時のモジュールテスト

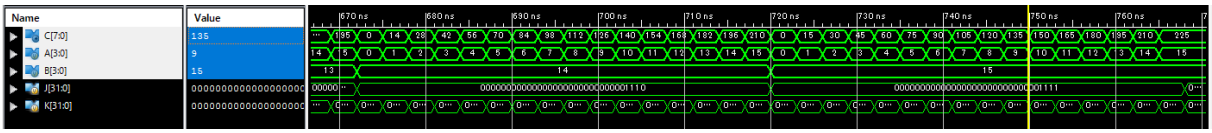


図 2.5: 4 ビット乗算器のモジュールテスト

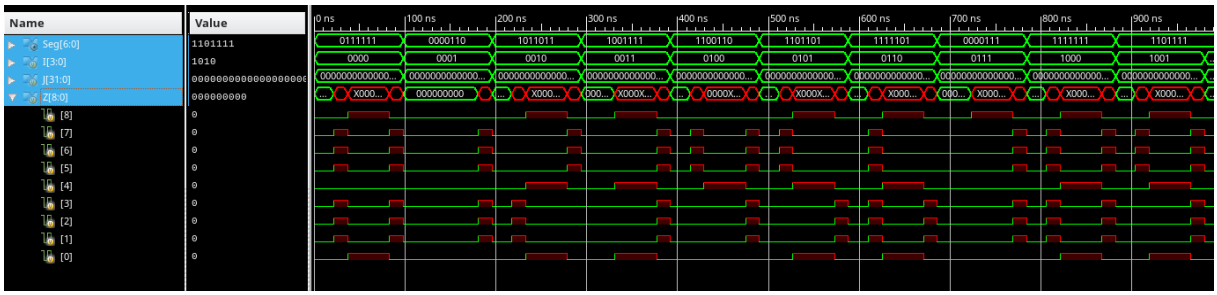


図 2.6: 7 セグデコーダのモジュールテスト

2.3.5 課題 4 7 セグメント LED への出力と発光の関係の観察

出力値と 7 セグメント LED の光り方の調査結果を調べた．実験方法で述べたように，課題 4 の実験は 2 つに分けて行われた．

前者のほうでは，プッシュスイッチを押さずにセグメントの発光パターンを調査する．結果としては，割り当てたセグメントとスライドスイッチの間の関係性が検証できた．dot に関しては High 出力のとき，小数点を表現するときに用いられる”.” が現れた．その他，回路図のセグメントの割り当てと同じようにセグメントが High で点灯した．

次に後者の方では，スライドスイッチを幾つか High にセットした状態で 1 つずつプッシュスイッチを押す．押すとスイッチの出力は High になる．押した桁に対応する桁が消灯した．4 つ全て押すと完全に消灯した．3 つだけ押すと，一つだけ残して消えた．

2.3.6 課題 5 7 セグメント LED 表示回路の作成

7 セグメント LED は目的の動作を果たしていた．0-7 のバイナリ入力に対応する 7 セグメント表示ができていた．

2.3.7 課題 6 乗算における符号について

テストベンチの結果を図 2.7 に示す．符号を考慮した演算結果になっている．符号なしでは単に乗算計算をしている．

2.4 考察

2.4.1 課題 1 全加算器

1 ビットの加算計算ができていることがわかる．

2.4.2 課題 2A 4 ビット加減算器

4 ビットの加減算計算が正しく動作していることがわかる．

2.4.3 課題 2B 4 ビット乗算器

4 ビットの乗算計算が正しく動作していることがわかる．符号は考慮していないので，符号なしの計算のみしかできない．

2.4.4 課題 3 手続きブロックを用いた 7 セグメントデコーダの作成

7 セグメント LED 用のデコードが正しく動作していることがわかる．2 週目で設計した回路図と同様の動作をした．こちらのほうが開発時間がかなり短くなっている．以前は 3 時間くらいはかかってたが，今回は 20 分くらいで同様の動作を実現した．

2.4.5 課題 4 7 セグメント LED への出力と発光の関係の観察

7 セグメント LED の動作として，桁を決めるピンと，7 セグメント LED のセグを制御するポートがあることがわかる．ポートとセグメントの対応は，回路図通りであることが分かった．アノード側は Low で光ることがわかる．カソード側は High で光る．

両方が AND の条件式になっていて，両方成り立つときのみ LED が点灯する．セグメントの制御用ポートがすべての桁で共通していることから，すべての桁を別々に，また同時に制御することはできない．桁を高速に切り替えながら，それぞれの桁を表示するダイナミック制御を行うことが考えらる．

4 桁の 7 セグメント LED のように，なにも工夫しないと $(7+1)\times 4 = 32$ 個の LED を同時に制御するのには多数のポートが必要になる．しかし FPGA ではポートが有限であるため，1 つの LED に 1 つのポートを対応させるのは入出力ポートの資源の面から考えると良くない．この回路で使われているダイナミック点灯の方式では，この場合は $8+4 = 12$ ポートで制御できる．半分以下までポート数を削減できるが，同時には制御できないので，高速に桁を切り替えながら表示することが考えられる．そのためデメリットとしては LED の輝度が $\frac{1}{4}$ まで低下することが考えられる．

2.4.6 課題 5 7 セグメント LED 表示回路の作成

7 セグメント LED 用の出力を正しく動作している．今回は case 文で実装しているが，if 文でも同様の実装ができることが考えられる．

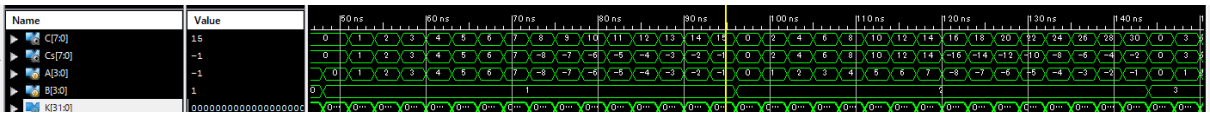


図 2.7: 4 ビット乗算器における符号ありと符号なしの違い

2.4.7 課題 6 乗算における符号について

符号ありと符号なしの乗算の演算結果に違いが出た．符号ありでは，私たちは普段，符号同士の演算を別にしておいて，後で符号をまとめる．乗算器でも同様に，符号なしでは符号の計算は考慮されなかった．すべて符号なしとして処理されている．逆に符号ありでは，符号を考慮しての計算ができていることがわかる．

2.5 感想

以前 7 セグメント LED 用のデコード回路を作成したときよりも Verilog で記述したほうが製作時間が短くった．ハードウェアの最適化は人がやるところと機械に任せるところの選定が必要だと思った．全体の構造は人の手で決めるべきだが，細かい部分はコンピュータの論理合成により行われたほうが簡単にできる．

また，7 セグメント LED の制御回路に工夫が見られた．ポート数は，マイコンでプログラミングする際にも制約として考慮しなければならないことが多い．少ないポートで効率よく制御するためには，回路上の工夫が欠かせないことがわかった．

また，加算器に関しては，このままでは桁が長くなると，計算時間が長くなると思った．桁上がりが入るのが，前の桁を計算しなければ桁上がりがわからないためである．高速化するには桁上がりを前もって計算するための機構などを用意する必要があると思う．

2.6 付録

ソースコード 2.1: 課題 1 ソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 13:00:13 04/24/2019
7  // Design Name:
8  // Module Name: ADDER
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////
21 module HADDER(
22         input A,
23         input B,
24         output S,
25         output Co
26         );
27         assign Co = A & B;
28         assign S = A ^ B;
29 endmodule
30
31 module ADDER(
32     input A,
33     input B,
34     input C,
35     output S,
36     output Co
37     );
38          wire F;
39          wire G;
40          wire H;
41
42         HADDER U1(.A(A), .B(B), .S(F), .Co(G));
43         HADDER U2(.A(F), .B(C), .S(S), .Co(H));
44         assign Co = G | H;
45
46
47 endmodule
```

ソースコード 2.2: 課題 1 テストベンチ

```
1  `timescale 1ns / 1ps
2
```

```
3  //
       //////////////////////////////////////////////////////////////////////////////////
4  // Company:
5  // Engineer:
6  //
7  // Create Date: 13:19:06 04/24/2019
8  // Design Name: ADDER
9  // Module Name: Z:/adder/testbench.v
10 // Project Name: adder
11 // Target Device:
12 // Tool versions:
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: ADDER
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 //
       //////////////////////////////////////////////////////////////////////////////////
24
25 module testbench;
26
27         // Inputs
28         reg A;
29         reg B;
30         reg C;
31
32         // Outputs
33         wire S;
34         wire Co;
35
36         // Instantiate the Unit Under Test (UUT)
37         ADDER uut (
38                 .A(A),
39                 .B(B),
40                 .C(C),
41                 .S(S),
42                 .Co(Co)
43         );
44
45         initial begin
46                 // Initialize Inputs
47                 A = 0;
48                 B = 0;
49                 C = 0;
50
51                 // Wait 100 ns for global reset to finish
52                 #10;
53                 // Initialize Inputs
54                 A = 0;  B = 0;  C = 0;  #10;
55                 A = 1;  B = 0;  C = 0;  #10;
56                 A = 0;  B = 1;  C = 0;  #10;
57                 A = 1;  B = 1;  C = 0;  #10;
58                 A = 0;  B = 0;  C = 1;  #10;
59                 A = 1;  B = 0;  C = 1;  #10;
```

```
60                  A = 0;  B = 1;  C = 1;  #10;
61                  A = 1;  B = 1;  C = 1;  #10;
62
63                  // Add stimulus here
64
65          end
66
67  endmodule
```

ソースコード 2.3: 課題 2A ソースコード

```
1   `timescale 1ns / 1ps
2   //
        //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 13:26:46 04/24/2019
7   // Design Name:
8   // Module Name: adder4
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //
        //////////////////////////////////////////////////////////////////////////////////
21  module HADDER(
22          input A,
23          input B,
24          output S,
25          output Co
26          );
27          assign Co = A & B;
28          assign S = A ^ B;
29  endmodule
30
31  module ADDER(
32      input A,
33      input B,
34      input C,
35      output S,
36      output Co
37      );
38            wire F;
39            wire G;
40            wire H;
41
42          HADDER U1(.A(A), .B(B), .S(F), .Co(G));
43          HADDER U2(.A(F), .B(C), .S(S), .Co(H));
44          assign Co = G | H;
45  endmodule
46
```

```
47
48 module ADDER4(
49     input [3:0] A,
50     input [3:0] B,
51     input SIG,
52     output [3:0] S,
53     output C
54     );
55
56         wire [3:0]Ba;
57         wire [3:0]Co;
58         assign Ba = {4{SIG}}^B;
59
60         ADDER U0(.A(A[0]),.B(Ba[0]),.C(SIG),.S(S[0]),.Co(Co[0]));
61         ADDER U1(.A(A[1]),.B(Ba[1]),.C(Co[0]),.S(S[1]),.Co(Co[1]));
62         ADDER U2(.A(A[2]),.B(Ba[2]),.C(Co[1]),.S(S[2]),.Co(Co[2]));
63         ADDER U3(.A(A[3]),.B(Ba[3]),.C(Co[2]),.S(S[3]),.Co(Co[3]));
64         assign C = Co[3];
65
66 endmodule
```

ソースコード 2.4: 課題 2A テストベンチ

```
1  `timescale 1ns / 1ps
2
3  //
       //////////////////////////////////////////////////////////////////////////////////
4  // Company:
5  // Engineer:
6  //
7  // Create Date: 13:46:17 04/24/2019
8  // Design Name: ADDER4
9  // Module Name: Z:/adder4/testbench.v
10 // Project Name: adder4
11 // Target Device:
12 // Tool versions:
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: ADDER4
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 //
       //////////////////////////////////////////////////////////////////////////////////
24
25 module testbench;
26
27         // Inputs
28         reg [3:0] A;
29         reg [3:0] B;
30         reg SIG;
31
32         // Outputs
33         wire [3:0] S;
34         wire C;
```

```
35
36          // Instantiate the Unit Under Test (UUT)
37          ADDER4 uut (
38                  .A(A),
39                  .B(B),
40                  .SIG(SIG),
41                  .S(S),
42                  .C(C)
43          );
44          integer J;
45          integer K;
46          integer SI;
47
48          initial begin
49                  // Initialize Inputs
50                  A = 0;
51                  B = 0;
52                  SIG = 0;
53
54                  // Wait 100 ns for global reset to finish
55                  #10;
56                  for(SI=0; SI<2; SI=SI+1) begin  // 0     15 ^^dc^^82^^c5^^82
57                          for(J=0; J<16; J=J+1) begin  // 0     15 ^^dc^^82^^c5^^82

58                                  for(K=0; K<16; K=K+1) begin  // 0     15 ^^dc
                                        ^^82^^c5^^82
59                                          A=K;
60                                          B=J;
61                                          SIG = SI;
62                                          #3;
63                                  end
64                          end
65                  end
66                  // Add stimulus here
67
68          end
69
70  endmodule
```

ソースコード 2.5: 課題 2B ソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////

3  // Company:
4  // Engineer:
5  //
6  // Create Date: 15:53:45 04/24/2019
7  // Design Name:
8  // Module Name: multi
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
```

```
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////
21 module multi(
22     input [3:0] A,
23     input [3:0] B,
24     output [7:0] C
25     );
26
27         assign C=(A&{4{B[0]}})+((A&{4{B[1]}})<<1)+((A&{4{B[2]}})<<2)+((A
               &{4{B[3]}})<<3);
28 endmodule
```

ソースコード 2.6: 課題 2B テストベンチ

```
1  `timescale 1ns / 1ps
2
3  //
       //////////////////////////////////////////////////////////////////////////////////
4  // Company:
5  // Engineer:
6  //
7  // Create Date: 16:01:39 04/24/2019
8  // Design Name: multi
9  // Module Name: Z:/multimake/testbench.v
10 // Project Name: multimake
11 // Target Device:
12 // Tool versions:
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: multi
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 //
       //////////////////////////////////////////////////////////////////////////////////
24
25 module testbench;
26         integer J;
27         integer K;
28         // Inputs
29         reg [3:0] A;
30         reg [3:0] B;
31
32         // Outputs
33         wire [7:0] C;
34
35         // Instantiate the Unit Under Test (UUT)
36         multi uut (
37                 .A(A),
38                 .B(B),
39                 .C(C)
40         );
```

```
41
42          initial begin
43                  // Initialize Inputs
44                  A = 0;
45                  B = 0;
46
47                  for(J=0; J<16; J=J+1) begin  // 0     15 ^^dc^^82^^c5^^82
48                          for(K=0; K<16; K=K+1) begin  // 0     15 ^^dc^^82^^c5
                                ^^82
49                                  A=K;
50                                  B=J;
51                                  #3;
52                          end
53                  end
54
55                  // Add stimulus here
56
57          end
58
59  endmodule
```

ソースコード 2.7: 課題 3 ソースコード

```
1   `timescale 1ns / 1ps
2   //
        //////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 14:15:28 04/24/2019
7   // Design Name:
8   // Module Name: seg7proc
9   // Project Name:
10  // Target Devices:
11  // Tool versions:
12  // Description:
13  //
14  // Dependencies:
15  //
16  // Revision:
17  // Revision 0.01 - File Created
18  // Additional Comments:
19  //
20  //
        //////////////////////////////////////////////////////////////////////////////////
21  module seg7proc(
22      input [3:0] I,
23      output reg [6:0] Seg
24      );
25
26          always @(*) begin
27                  case(I)
28                          4'd0: begin Seg <= 7'b0111111; end
29                          4'd1: begin Seg <= 7'b0000110; end
30                          4'd2: begin Seg <= 7'b1011011; end
31                          4'd3: begin Seg <= 7'b1001111; end
32                          4'd4: begin Seg <= 7'b1100110; end
33                          4'd5: begin Seg <= 7'b1101101; end
34                          4'd6: begin Seg <= 7'b1111101; end
```

```
35                          4'd7: begin Seg <= 7'b0000111; end
36                          4'd8: begin Seg <= 7'b1111111; end
37                          4'd9: begin Seg <= 7'b1101111; end
38                  endcase
39          end
40  endmodule
```

ソースコード 2.8: 課題 3 テストベンチ

```
1   `timescale 1ns / 1ps
2
3   //
        //////////////////////////////////////////////////////////////////////////////////
4   // Company:
5   // Engineer:
6   //
7   // Create Date: 14:26:11 04/24/2019
8   // Design Name: seg7proc
9   // Module Name: Z:/seg7proc/testbench.v
10  // Project Name: seg7proc
11  // Target Device:
12  // Tool versions:
13  // Description:
14  //
15  // Verilog Test Fixture created by ISE for module: seg7proc
16  //
17  // Dependencies:
18  //
19  // Revision:
20  // Revision 0.01 - File Created
21  // Additional Comments:
22  //
23  //
        //////////////////////////////////////////////////////////////////////////////////
24
25  module testbench;
26          // Inputs
27          reg [3:0] I;
28
29          // Outputs
30          wire [6:0] Seg;
31          integer J; // [v^^cf^^90 (for^^c9^^97p    )
32          reg [8:0]Z;
33
34          // Instantiate the Unit Under Test (UUT)
35          seg7proc uut (
36                  .I(I),
37                  .Seg(Seg)
38          );
39
40          initial begin
41                  for(J=0; J<16; J=J+1) begin  // 0     15 ^^dc^^82^^c5^^82
42                          I <= J;  // [v^^cf^^90l    fR[_^^c9^^97^
43                          Z <= 0;  // }[W    ю ^^d4^^83}[W    \
44                          #22;
45
46                          Z <= {1'b0, {3{~Seg[5] ? 1'b0 : 1'bx}}, 1'b0,
47                                  {3{~Seg[4] ? 1'b0 : 1'bx}}, 1'b0}; // c (E, F)
```

```
                                                \
48                          #15
49
50                          Z <= {~Seg[0] ? 1'b0 : 1'bx, 3'b000, ~Seg[6] ? 1'b0 : 1'bx,
51                                  3'b000, ~Seg[3] ? 1'b0 : 1'bx};  // (A, D, G)
                                                \
52                          #46
53
54                          Z <= {1'b0, {3{~Seg[1] ? 1'b0 : 1'bx}}, 1'b0,
55                                  {3{~Seg[2] ? 1'b0 : 1'bx}}, 1'b0}; // Ec (B, C)
                                                \
56                          #15
57
58                          ;
59                  end
60                  Z <= 0; #20 // E}[W    \
61                  $finish;
62      end
63  endmodule
```

ソースコード 2.9: 課題 4 ピン割り当て

```
1
2   # PlanAhead Generated physical constraints
3
4   NET "A[7]" LOC = P11;
5   NET "A[6]" LOC = L3;
6   NET "A[5]" LOC = K3;
7   NET "A[4]" LOC = B4;
8   NET "A[3]" LOC = G3;
9   NET "A[2]" LOC = F3;
10  NET "A[1]" LOC = E2;
11  NET "A[0]" LOC = N3;
12  NET "B[3]" LOC = G12;
13  NET "B[2]" LOC = C11;
14  NET "B[1]" LOC = M4;
15  NET "B[0]" LOC = A7;
16  NET "C[7]" LOC = L14;
17  NET "C[6]" LOC = H12;
18  NET "C[5]" LOC = N14;
19  NET "C[4]" LOC = N11;
20  NET "C[3]" LOC = P12;
21  NET "C[2]" LOC = L13;
22  NET "C[0]" LOC = N13;
23  NET "C[1]" LOC = M12;
24  NET "D[3]" LOC = F12;
25  NET "D[2]" LOC = J12;
26  NET "D[1]" LOC = M13;
27  NET "D[0]" LOC = K14;
```

ソースコード 2.10: 課題 4 ソースコード

```
1   ‘timescale 1ns / 1ps
2   //
        //////////////////////////////////////////////////////////////////////////////////

3   // Company:
4   // Engineer:
5   //
6   // Create Date: 14:57:31 04/24/2019
7   // Design Name:
```

```
// Module Name: check7seg
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////

module check7seg(
    input [7:0] A,
    input [3:0] B,
    output [7:0] C,
    output [3:0] D
    );

        assign C=A;
        assign D=B;

endmodule
```

ソースコード 2.11: 課題 5 ピン割り当て

```
1
2
3  NET "I[2]" LOC = C11;
4  NET "I[1]" LOC = M4;
5  NET "I[0]" LOC = A7;
6  NET "Seg[6]" LOC = M12;
7  NET "Seg[5]" LOC = L13;
8  NET "Seg[4]" LOC = P12;
9  NET "Seg[3]" LOC = N11;
10 NET "Seg[2]" LOC = N14;
11 NET "Seg[0]" LOC = L14;
12 NET "Seg[1]" LOC = H12;
```

ソースコード 2.12: 課題 5 ソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////

3  // Company:
4  // Engineer:
5  //
6  // Create Date: 14:15:28 04/24/2019
7  // Design Name:
8  // Module Name: seg7proc
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
```

```
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
    //////////////////////////////////////////////////////////////////////////////////
21 module seg7proc(
22     input [2:0] I,
23     output reg [6:0] Seg
24     );
25
26         always @(*) begin
27                 case(I)
28                         3'd0: begin Seg <= ~7'b0111111; end
29                         3'd1: begin Seg <= ~7'b0000110; end
30                         3'd2: begin Seg <= ~7'b1011011; end
31                         3'd3: begin Seg <= ~7'b1001111; end
32                         3'd4: begin Seg <= ~7'b1100110; end
33                         3'd5: begin Seg <= ~7'b1101101; end
34                         3'd6: begin Seg <= ~7'b1111101; end
35                         3'd7: begin Seg <= ~7'b0000111; end
36                 endcase
37         end
38 endmodule
```

ソースコード 2.13: 課題 6 ソースコード

```
1 `timescale 1ns / 1ps
2 //
    //////////////////////////////////////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 15:29:57 04/24/2019
7 // Design Name:
8 // Module Name: multitest
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
    //////////////////////////////////////////////////////////////////////////////////
21 module multitest(
22     input [3:0] A,
23     input [3:0] B,
24     output [7:0] C,
25     output signed [7:0] Cs
26     );
27
28         wire signed [3:0]As = A;
```

```
29          wire signed [3:0]Bs = B;
30          assign C = A*B;
31          assign Cs = As*Bs;
32
33  endmodule
```

ソースコード 2.14: 課題 6 テストベンチ

```
1   `timescale 1ns / 1ps
2
3   //////////////////////////////////////////////////////////////////////////////////
4   // Company:
5   // Engineer:
6   //
7   // Create Date: 15:33:59 04/24/2019
8   // Design Name: multitest
9   // Module Name: Z:/multest/testbench.v
10  // Project Name: multest
11  // Target Device:
12  // Tool versions:
13  // Description:
14  //
15  // Verilog Test Fixture created by ISE for module: multitest
16  //
17  // Dependencies:
18  //
19  // Revision:
20  // Revision 0.01 - File Created
21  // Additional Comments:
22  //
23  //////////////////////////////////////////////////////////////////////////////////
24
25  module testbench;
26          integer K;
27          integer J;
28
29          // Inputs
30          reg [3:0] A;
31          reg [3:0] B;
32
33          // Outputs
34          wire [7:0] C;
35          wire [7:0] Cs;
36
37          // Instantiate the Unit Under Test (UUT)
38          multitest uut (
39                  .A(A),
40                  .B(B),
41                  .C(C),
42                  .Cs(Cs)
43          );
44
45          initial begin
46                  // Initialize Inputs
47                  A = 0;
48                  B = 0;
49
```

```
50                  for(J=0; J<16; J=J+1) begin  // 0     15 ^^dc^^82^^c5^^82
51                          for(K=0; K<16; K=K+1) begin  // 0     15 ^^dc^^82^^c5
                                ^^82
52                                  A=K;
53                                  B=J;
54                                  #3;
55                          end
56                  end
57
58                  // Add stimulus here
59
60          end
61
62  endmodule
```

第3章　4週目 Verilog を用いた順序回路の設計

3.1　目的

3 週目では組み合わせ回路を設計した．4 週目では順序回路の設計を学ぶ．また複雑な順序回路設計を体験する．

3.2　実験方法

3.2.1　課題 1 10 進数カウンタの作成

10 進カウンタを設計する．

ソースコードを付録のソースコード 3.1 示す．テストベンチを付録のソースコード 3.2 示す．

3.2.2　課題 2

回路図 (図 3.1) の動作を予測し，実際に動作させ確かめてみる．動作予測としては，2 通り考えられる．図 3.2 に，考えられる回路構成を示す．この矢印のどちらに配線が入っているかで結果が変わると考えられる．

パターン 1 では，LED の負荷により High 出力のディレイがかかり，仮に LED に Hi 出力がされていてもその瞬時には Hi と認識できるレベルまで電圧が上がらず，読み出した際に Low として読み込まれる現象が考えられる．そのため，ボタンを離したときに LED が消灯する回数が多くなることが考えられる．

パターン 2 では，バッファやラッチが挿入されて，出力負荷に関係なく論理レベルが決定される例で，ボタンを押している間，LED がルーレットのように 25MHz で点滅し，ボタンを離した瞬間の出力を保持し続けるという予想がある．そのため，押している時間がランダムで十分に長いときに離した瞬間の LED の値もランダムとなる．

これは出力段にラッチの挿入がない場合に考えられるが，実際の出力をもとにした論理合成が行われると起こりうる．多くのマイクロコントローラでも同様の問題が提起されており，PIC コントローラの場合は後続のチップでは D ラッチを搭載することで問題を回避した．

仮にこのような現象が起きた場合，同条件のもとでも High になる確率と Low になる確率が同じにならない．

ソースコードを付録のソースコード 3.3 示す．

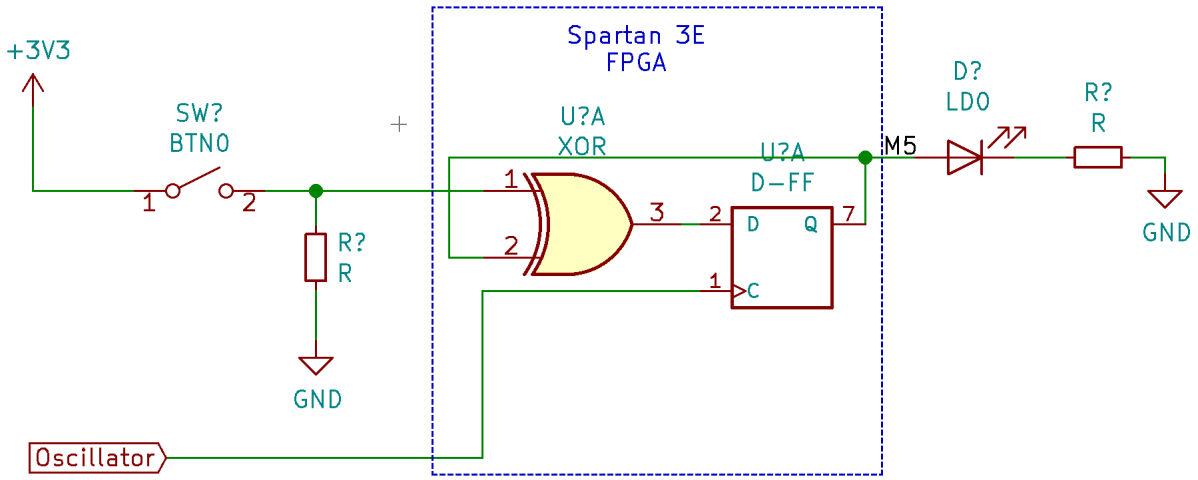


図 3.1: 課題 2 の回路図

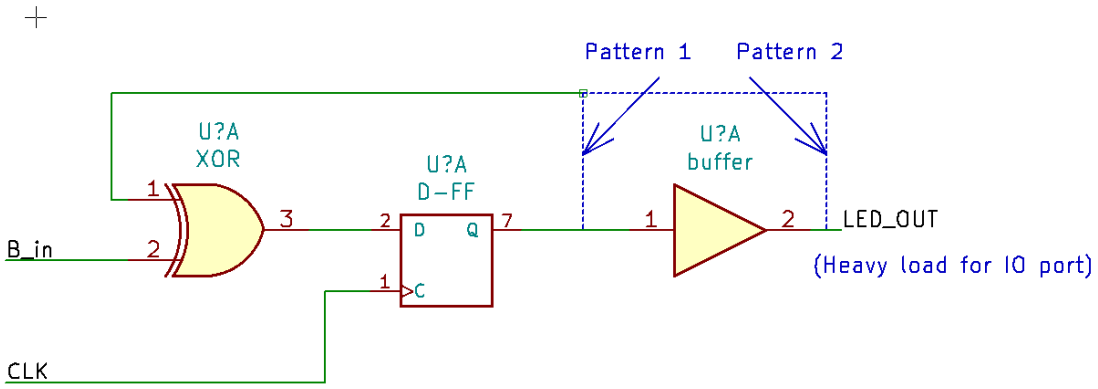


図 3.2: IO ポートの構成によって結果が変わる例

3.2.3 課題 3 LED の点滅

LED1 個を 1Hz で点滅させる．クロック源のクロック周波数は 50MHz なので 25’000’000 回 カウントした段階で LED をトグルさせることで実現できる．

ソースコードを付録のソースコード 3.4 示す．

3.2.4 課題 4 7 セグメント LED の 10 進カウンタ

押しボタンを押すたびに 7 セグメント LED1 桁の値を 0 → 1 → 2 ... → 9 → 0 → ... と変化する回路を作成する．

押しボタンの入力には，チャタリングが含まれている可能性があるので，内部でチャタリング除去する必要がある．また，押しボタンをトリガにイベントを起こすことは許され

ない．クロック源は 50[MHz] の発振器から供給し，それを分周した 10[ms] 程度のサンプリング周期でサンプリングを行い，チャタリングを除去する．

押下を判定するためには以前の状態を保持し，直前は押されず，今押されたことを確認する必要がある．サンプリングした値をもとに，押下判定を行い，カウント動作を行う．

ソースコードを付録のソースコード 3.5 示す．

3.2.5 課題 5 7 セグメント LED に 1234 を表示

7 セグメント LED に「1234」を表示する回路を作成する．

7 セグメント LED は 4 桁表示でそれぞれの桁はアノードコモンで構成されている．カソード側は同じタイプのセグメントが共有されていて，厳密に同時に複数の数字を表示することができない．

そのため高速に表示する桁を切り替えながら数字を一つずつ表示することであたかも同時に数字が表示されているかのように見えるようにする．この方式をダイナミック点灯方式という．

約 1ms ごとに桁を切り替え，4 桁分表示するように設計する．拡張性を考え，(課題 6 のため) モジュールとして汎用性がある形で設計する．

入力はクロックと表示する数値 4 桁分，出力は 7 セグメント LED への出力とする．

ソースコードを付録のソースコード 3.6 示す．

3.2.6 課題 6 4 桁のカウンタの作成

押しボタンを押すたびに 7 セグメント LED の値が 0000 → 0001 → 0002 → ... → 9999 → 0000 → ... と変化する回路を作成する．任意の位の桁上がりはその桁の現在の数値が 9 から 0 に変化するタイミングで発生し，それをもとに次の桁をインクリメントする．残りの機能としては課題 4 と 5 を組み合わせて実装する．

ソースコードを付録のソースコード 3.7 示す．

課題 6 除算器の制作

実験が終わったあとに，解説を見た．10000 進カウンタを作成し，それをもとに割り算回路で商とあまりを求めて表示する方法が記述されていた．

それについても実装テストを行うべきだと判断し，実装することにする．10000 進カウンタを作るためには 14 ビット必要である．この数と 10 での除算を実装することで，それぞれの桁を求めることができる．

10 の割り算を実装する方法はいくつか提唱されている．例えば，単純に除算器を作成し，順に割っていく方法や，逆数をかけて求める方法，整数のシフトを順に繰り返していく方法 [1] など，様々な方法がある．

割り算を行う除算器を作る方法や整数のシフトを順に繰り返していく方法は数サイクルを要するため，効率が悪い．今回は短時間に処理が終わる，逆数を求める方法で計算をする．

解説と同じやり方ではなく，ここでは整数の乗算とシフトを用いた除算法を行う．逆数を用いて乗算で計算する方法である．

まず 10 で割る必要があるので，10 の逆数を求める．しかし単に求めるだけでは整数の丸め誤差が割り算の結果に悪影響があるので，ここでは固定小数点演算で計算する際のフォーマットの選定に注意が必要である．固定小数点のフォーマットの選定は，必要な桁と割る数によって変える必要があるため，この方法は定数の割り算ではよく使われるが定数以外の割り算では使われない．固定小数点のフォーマットは小数点以下 $N + log_2(M)$; N はビット数，M は割る数 で表現できる．ここで切り捨て誤差を抑えるために，逆数で求めた値に 1 を足しておく．

この方法では，整数の丸め誤差が含まれるが，整数の範囲 (ビット数) を限定することで，誤差を小数点以下に追い込むことができる [2].

今回は 14 ビットの数値を 10 で割った商と余りを求めているので，小数点以下 17 ビットの逆数を用いる．つまりある数に 13108 をかけて，17 ビットシフトすると 10 で割った商が求められる．

余りを求めるためには，この数を 10 でかけて差分を取ることで求められるが，多くのビットの計算になり，時間がかかる．そのため，精度検証をしながら別の方法であまりを求める．あまりとは，割ったときの小数部分をオフセットした値に比例する．そのため割ったあまりをつかってある程度表現できる．これが完璧ではないのは，すでに逆数に誤差が存在するためであるが，検証をした上で回路を記述することができる．余りは割った数の小数点以下 5 ビットを 10 倍することで求めることができる[1].

作成したモジュールを挙げる

- 10 の除算器
- プリスケーラ
- 7 セグメント LED 表示
- カウンタ本体

10 の除算器は入力値を 10 で割った商と余りを出力する．

プリスケーラは 50MHz のクロック信号を分周するのに使う．クロック周波数を $\frac{1}{2}$ から $\frac{1}{2^{24}}$ までリニアに調節できるようになっている.[2]

7 セグメント LED 表示モジュールへのクロック周波数を下げるとダイナミック点灯を目でも観測できる．

7 セグメント LED 表示モジュールは，入力値を表示する．入力値は順に 10 で割られて余りは 7 セグメント LED に出力する．割った商は次の割る数にセットされ，次のクロック信号までに計算が完了する．これを繰り返すことで 7 セグメント LED に数値を表示することができる．

カウンタ本体はカウントをする．カウントした数値を 7 セグメント LED 表示モジュールへ出力する．

除算器を用いてた 10000 進カウンタのソースコードを付録のソースコード 3.8 示す.

[1] この方法は完璧ではなく，0 から 11263 までしか誤差なく計算できないが，今回扱う数は $10^4 - 1$ までなのでこのまま続けることにする

[2] 多くの場合クロックは分周ということから分母を整数で変更するが，今回は分子を変更し，クロック周波数を調節する．

3.2.7 課題 7 ΔΣ 変調による LED の調光

デルタシグマ変調

課題 7 は任意の回路を作成ということで，私は LED の調光を行うことにした．

課題 3 ではディジタル的な点滅を行ったが，オンとオフの点滅である．オンとオフの間でゆっくり変化させるためには調光を行う必要がある．LED の輝度をディジタル的に調光する方法として，よく使われる方式として PWM 方式[3]がある．

PWM 方式は安定した調光ができることが特徴である一方で EMC[4]対策としては特定の周波数に分布が偏るため有効ではない．また，調光の分解能を上げると PWM の周波数も下がるため，トレードオフの関係になる．

応答特性も悪い．周期が固定のため，周期ごとにしか値を変更できない．

今回採用した調光の方式は，デルタシグマ変調方式のディジタル変換である．1 ビットのデルタシグマ方式では，長期的に誤差が最小になるように出力が調整される．1 次遅延の場合，直前の出力の誤差成分を次の出力に伝搬させることで実現できる．

デルタシグマ変調方式の大きな特徴は，ダイナミックに変化する周波数で，高い分解能と低い低周波数成分を実現できることが期待できる．EMC 対策として周波数を拡散することが有効であるが，広い周波数に渡って周波数を拡散できるため有効な対策になる．

欠点としては，調光に必要な回路規模が大きくなることと，スイッチング回数の増大によるスイッチング損失の増大である．PWM のように特定の周波数のみに対してのノイズ対策にはならないので，スイッチングの回路設計も煩雑になる．

仕様

デルタシグマ変調を実装するにあたり，必要なモジュールを挙げる．

- 分周器 (プリスケーラ)
- デルタシグマ変調器
- テスト用モジュール

まず，分周器を設計する．クロック信号が高速すぎるため，それを低速なクロックに分周するためのモジュールを用意する．このモジュールでは，クロックを $\frac{(1+N)}{4096}$ 倍にするためのモジュールである．N は 0 2047 までの 11 ビットで，$\frac{1}{2}$ から $\frac{1}{2048}$ までリニアに分周できる．

デルタシグマ変調器では，分解能 16 ビットの変調器を生成する．そのまま変調すると高速なシグナルができるが，LED の出力に FPGA の出力をそのまま使うため，少し周波数を落として使うために，分周器を用いて分周し，クロック周波数を調整する．デルタシグマ変調方式では 17′h10000 を基準とし，現在の出力がそれ以上かそれ以下で論理レベルを決定する．論理レベルを決定したあと，その誤差を現在の出力に加算する．

[3]Pulse Width Modulation の略で，パルス幅変調のこと
[4]Electro Magnetic Interference の略，電磁妨害のこと

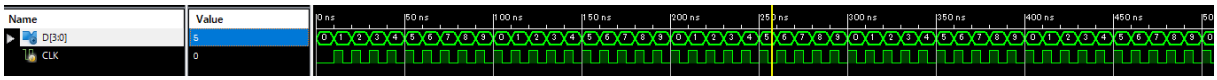


図 3.3: 10 進カウンタのテストベンチの結果

テスト用モジュールは，デルタシグマ変調が期待通りに動作しているかを検証するのに使う．検証するのに 4 つのモードを用意する．スライドスイッチでモードを自由に切り替えられるようにする．

1 つ目はゆっくり LED を点灯，消灯を繰り返すために 0 から 65535 まで出力をカウントアップし，そこから 0 までカウントダウンする．カウントアップのトリガには別の分周器により分周したクロック信号を使う．

2 つ目は出力を 0(消灯) に固定，3 つ目は，約 40%の出力，4 つ目は 100%出力のテストを行う．

実験の手順としては，おおよその動作をシミュレーションする．その後実際に実機で動かし，モードを変化させながらその様子の違いを観察する．

ソースコードを付録のソースコード 3.9 示す．

3.3 実験結果

3.3.1 課題 1 10 進数カウンタの作成

シミュレーション結果を図 3.3 に示す．

3.3.2 課題 2

指で押しているときは LED が半端な明るさになり，離すと LED が点灯することもあれば消灯することもあった．指を離した際にほぼ同じ確率で LED がついたり消えたりした．

3.3.3 課題 3 LED の点滅

LED が 1 秒周期で点滅した．

3.3.4 課題 4 7 セグメント LED の 10 進カウンタ

7 セグメント LED がボタンを押すたびに 0 → 1 → ... → 9 → 0 → ... と変化した．

3.3.5 課題 5 7 セグメント LED に 1234 を表示

7 セグメント LED に 1234 と出力できている．

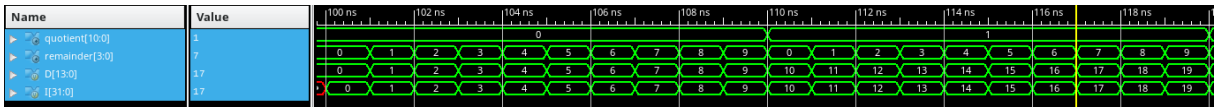


図 3.4: 除算器のテストベンチ 1

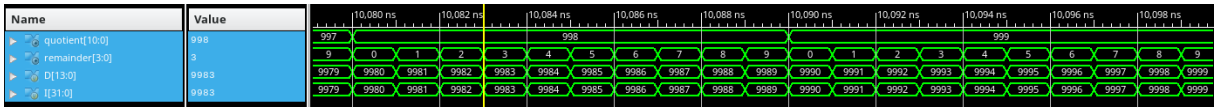


図 3.5: 除算器のテストベンチ 2

3.3.6　課題 6 4 桁のカウンタの作成

0 からカウントして 2000 程度まで手動で入力し，検証して，9999 に初期値を設定して 0 に戻ることを確認した．その他は検証していない．

除算器を用いた方法での表示部のテストベンチ

除算器のテストベンチの結果を図 3.5 と図 3.5 に示す．D が割る数で 10 で割った商 (quotient) と余り (remainder) を出力している．表示器のテストベンチの結果を図 3.6 に示す．num は表示する数が入っていて，I1 から I4 に数字に対応する数が計算され，順に格納されている様子がわかる．例えば 21 だと I4 に 1 が，I3 に 2，それ以外は 0 が格納されている．

3.3.7　課題 7 $\Delta\Sigma$ 変調による LED の調光

テストベンチを図 3.7 と図 3.8 に示す．

図 3.7 では，最初のモードのゆっくり点滅するモードで，1 周期の中から消灯部分付近を拡大した部分を出している．この変調方式の特徴である，パルスの密度が変化している様子がわかる．

図 3.8 では，実際に波形が一定ではない部分を抜き出して示している．

実際に実機で動作させてみると，LED をうまく調光できなかった．画像等ではうまく表せないが，低周波発振を起こし，更に出力が両極端の時に不安定になった．

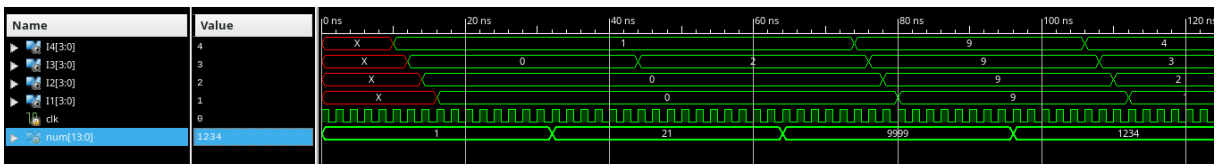


図 3.6: 表示機のテストベンチ

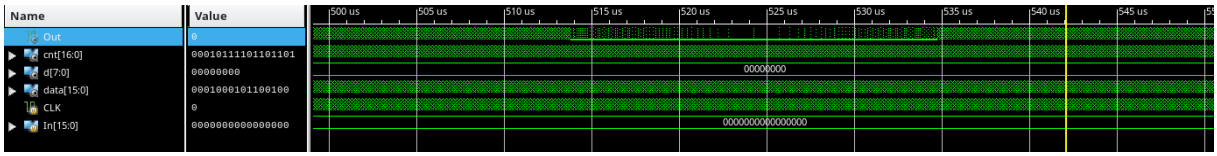


図 3.7: ΔΣ 変調のテストベンチ 1

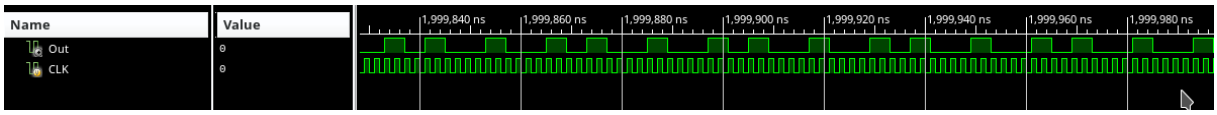


図 3.8: ΔΣ 変調のテストベンチ 2

3.4 考察

3.4.1 課題 1

クロック信号の立ち上がりでカウントできていることが分かる．0 から 9 までカウントし，その後 0 に戻っている．

3.4.2 課題 2

ボタンを押していても放した時の値はランダムに変化するようにみられたことから，図 3.2 中の Pattern 2 の方でコンフィグレーションが行われたのだと考えられる．つまり，ポートの読み出しタイミングに関して LED に出力する前にバッファが挟まっていて，その前段階で，XOR の入力部にフィードバックされていると考えられる．

3.4.3 課題 3 LED の点滅

LED が 1 秒ごとに点滅した．クロック源が正確であるため，安定した点滅のように見えた．

3.4.4 課題 4 7 セグメント LED の 10 進カウンタ

10 進カウンタの動作をした．LED が明く点灯したのは点灯している LED が 1 桁分だけのため，4 桁表示時よりも 1 桁に集中して電流が流れたためだと考えられる．

3.4.5 課題 5 7 セグメント LED に 1234 を表示

7 セグメント LED の表示は，1234 になった．7 セグメント LED の桁を高速に切り替えることであたかも同時に表示されているかのように見えたのは，目が高速な変化に追いついていないためだと考えられる．

3.4.6　課題6 4桁のカウンタの作成

2つの方法共にうまくカウントアップできた．一つずつ考察していく．

10進カウンタを工夫する方法

この方法では，10進カウンタの動作以外に任意の数字を表示する機能は有さない．そのため拡張性が低いと考えられる．

除算器を使う方法

除算器のテストベンチの結果，正しく10で割った商と余りを求められていることがわかる．

数値を各桁で分離するテストにも成功していることがわかる．I1 から I4 に10進のそれぞれの数値が入っていることが読み取れる．

割り算回路は1つだけなので，順番に計算され，順に格納されているが，この回路のダイナミック点灯方式では1桁ずつしか表示できないので，割り算回路を1つ用意して順に割って余りを出せば十分である．

この方法では，回路は少々複雑になったが任意の4桁の数字を表示できた．拡張が容易なため，拡張性が高いと考えられる．

割り算回路について考察していく．同様の計算を割り算回路で行うと，複数サイクル必要なので，1サイクルで終わる乗算器で逆数を求めたほうが効率が高い．ただし，精度の考察が必要である点と，予め逆数を求める必要があるのが欠点である．割る数に応じて回路構成を変更する必要がある．

3.4.7　課題7 ΔΣ変調によるLEDの調光

テストベンチからは誤差がうまく分散している様子がわかる．図3.8でわかるように，このΔΣ変調方式では同じような出力範囲でも波形としてみるとHighとLowの幅が一定の出力にはならない．常に累積誤差が最小になるように，出力が調節されるためである．

図3.7でもその様子がよくわかる.，最初のモードのゆっくり点滅するモードで，1周期の中から消灯部分付近を拡大した部分を出している．この変調方式の特徴である，パルスの密度が変化している様子がわかる．

調光に関して，最初にテストベンチを書いて予想できたが，やはり実際の動作は不安定だった．低周波発振も起きて，少しちらつきも見えた．特に0%や100%付近の出力は，不安定になった．これはΔΣ変調が交流信号向けだからだと思われる．交流等価で見ると，PWMよりも特性が良くなることがあるものの，直流には精度が限定されたPWMが良いことが分かった．

3.5 感想

FPGA による同期回路の設計に関する基本的な知識を身につけられたと思う．ハードウェアにとって実装が容易なタスク，そうでないタスクについて，振り分けを行い，ソフトウェアと共同して処理をすすめることがコンピュータの発展では欠かせないことがわかった．

3.5.1 ΔΣ 変調について

残りは，任意の回路を作成で，どんな回路を作るのか迷った．どうして ΔΣ 変調というマイナーな回路を作ることにしたのかについて説明しよう．

最初私は LED をチカチカさせて終わりにしようかなって思ってた．しかしそれではつまらないと思い，もっと高度なことをしようと思った．

そこで，以前 2 年前だろうか，私は個人で PIC32MK シリーズを用いて 3 ピンで NTSC[5] シグナルを生成する試みを行ったことがあった．そこでは，参考にした文献の中に 1 ピンで NTSC シグナルを生成する試みがあった．FPGA は 200MHz 付近の高速なシグナル生成でビット精度を稼いでいたがマイコンでは 50MHz 付近までしか使えない．代わりに 3 ピンだったらシグナル生成できるのではと考えたのがきっかけだった．3 ビットの DAC の出力ではもちろんカラーコンボジット入力のカラーバースト信号ですらもうまく生成できない．制約をパスするためには 3 ビットの入力をうまく変調して誤差を分散する必要があった．ここでは 1 次の ΔΣ フィルタを用いて誤差分散をかけた．1 波長を 16 回出力に設定した．1 ドットを $\frac{1}{2}$ 波長分，つまり 8 回出力にした．この方法では 4 : 3 スクリーンでは正方ドットにはならないが，ワイドスクリーンではほぼ正方ドットになる．

NTSC シグナルの生成はうまくいった．カラービデオ出力に成功し，3 ビットの DAC の難点であったカラー精度も時間を高速にしたことにより，幾分も改善することができた．この成功の影には，ΔΣ 方式の DA コンバータと NTSC シグナルの相性が良かったことが言えるだろう．NTSC シグナルは $fsc = 3.579545$[MHz] の基本周波数をもち，鋭い遮断特性を持つフィルタによりフィルタリングされる．このおかげでそのこの基本周波数の 16 倍のサンプリング周波数で出力したシグナルは安定してテレビに出力できた．

ただ，今回の実験でもうまく行くかと思った．直流ではそう簡単には行かないようだ．そのことがわかって，PWM 技術の長所を改めて実感することができた．

3.5.2 最後に

FPGA は前から興味を持ちながら，高額なハードウェア，それからソフトウェアライセンスにより，意欲が削がれていた．そういう自分を情けなく思うこともあり，高専時代の卒業研究では，FPGA を用いた研究をしようとしてたのだ．しかし卒業研究をすすめるうちに自分のやるべきこと，やらなくてはならないことが明確になり，FPGA を触るのをやめてしまった．

ハードウェアは可能性が大きい．そのため今後も継続してハードウェアとソフトウェアの共存について研究していきたいと思う．

[5] ビデオコンボジットの規格の一つ，昔のカラーテレビ放送の技術の一部に使われていた

3.6 付録

ソースコード 3.1: 課題 1 ソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 16:41:59 04/24/2019
7  // Design Name:
8  // Module Name: counter
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////
21 module counter(
22     input CLK,
23     output reg[3:0] D = 0
24     );
25         always @(posedge CLK) begin
26                 if(D == 4'd9) begin
27                         D <= 0;
28                 end else begin
29                         D <= D + 4'd1;
30                 end
31         end
32 endmodule
```

ソースコード 3.2: 課題 1 テストベンチ

```
1  `timescale 1ns / 1ps
2
3  //
       //////////////////////////////////////////////////////////////////////////////////
4  // Company:
5  // Engineer:
6  //
7  // Create Date: 16:46:05 04/24/2019
8  // Design Name: counter
9  // Module Name: Z:/counter/testbench.v
10 // Project Name: counter
11 // Target Device:
12 // Tool versions:
13 // Description:
14 //
15 // Verilog Test Fixture created by ISE for module: counter
```

```
16 //
17 // Dependencies:
18 //
19 // Revision:
20 // Revision 0.01 - File Created
21 // Additional Comments:
22 //
23 //
    //////////////////////////////////////////////////////////////////////////////////

24
25 module testbench;
26
27         // Inputs
28         reg CLK;
29
30         // Outputs
31         wire [3:0] D;
32
33         // Instantiate the Unit Under Test (UUT)
34         counter uut (
35                 .CLK(CLK),
36                 .D(D)
37         );
38
39         always begin
40                 CLK = 1;
41                 #5
42                 CLK = 0;
43                 #5;
44         end
45         initial begin
46                 // Initialize Inputs
47                 CLK = 0;
48                 // Wait 100 ns for global reset to finish
49                 #100;
50
51                 // Add stimulus here
52
53         end
54
55 endmodule
```

ソースコード 3.3: 課題 2 ソースコード

```
1 `timescale 1ns / 1ps
2 //
    //////////////////////////////////////////////////////////////////////////////////

3 // Company:
4 // Engineer:
5 //
6 // Create Date: 17:14:48 04/24/2019
7 // Design Name:
8 // Module Name: leddim
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
```

```
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
    //////////////////////////////////////////////////////////////////////////////////
21 module leddim(
22     input CLK,
23     input BTN,
24     output reg LED = 1'b0
25     );
26
27         always @(posedge CLK) begin
28                 LED <= LED ^ BTN;
29         end
30 endmodule
```

ソースコード 3.4: LED の点滅のソースコード

```
1  `timescale 1ns / 1ps
2  //
    //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 18:32:09 04/26/2019
7  // Design Name:
8  // Module Name: ledblink
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
    //////////////////////////////////////////////////////////////////////////////////
21 module ledblink(
22     input CLK,
23     output reg LED=1'b0
24     );
25
26         reg [25:0]TIM;
27         always @(posedge CLK) begin
28                 if(TIM == 26'd25000000) begin
29                         TIM <= 0;
30                         LED <= ~LED;
31                 end else begin
32                         TIM <= TIM + 26'd1;
33                 end
34         end
35 endmodule
```

ソースコード 3.5: 7 セグメント LED10 進カウンタソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 16:41:59 04/24/2019
7  // Design Name:
8  // Module Name: counter
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////
21 `timescale 1ns / 1ps
22 //
       //////////////////////////////////////////////////////////////////////////////////
23 // Company:
24 // Engineer:
25 //
26 // Create Date: 14:15:28 04/24/2019
27 // Design Name:
28 // Module Name: seg7proc
29 // Project Name:
30 // Target Devices:
31 // Tool versions:
32 // Description:
33 //
34 // Dependencies:
35 //
36 // Revision:
37 // Revision 0.01 - File Created
38 // Additional Comments:
39 //
40 //
       //////////////////////////////////////////////////////////////////////////////////
41 module seg7proc(
42     input [3:0] I,
43     output reg [6:0] Seg
44     );
45
46         always @(*) begin
47                 case(I)
48                         4'd0: begin Seg <= ~7'b0111111; end
49                         4'd1: begin Seg <= ~7'b0000110; end
50                         4'd2: begin Seg <= ~7'b1011011; end
51                         4'd3: begin Seg <= ~7'b1001111; end
```

```
52                          4'd4: begin Seg <= ~7'b1100110; end
53                          4'd5: begin Seg <= ~7'b1101101; end
54                          4'd6: begin Seg <= ~7'b1111101; end
55                          4'd7: begin Seg <= ~7'b0000111; end
56                          4'd8: begin Seg <= ~7'b1111111; end
57                          4'd9: begin Seg <= ~7'b1101111; end
58                  endcase
59          end
60  endmodule
61
62  module counter(
63      input BTN,
64          input CLK,
65      output [6:0]seg,
66          output [3:0]K
67      );
68          reg[32:0] TIM = 0;
69          reg pv;
70          reg [3:0]D;
71
72          seg7proc u1(.I(D),.Seg(seg));
73
74          assign K=4'b0111;
75
76          always @(posedge CLK) begin
77                  if(TIM == 32'd250000) begin
78                          TIM <= 0;
79                          if(pv == 0 && BTN != pv) begin
80                                  if(D != 9)begin
81                                          D <= D + 1;
82                                  end else begin
83                                          D <= 0;
84                                  end
85                          end
86                          pv <= BTN;
87                  end else begin
88                          TIM <= TIM + 32'd1;
89                  end
90          end
91  endmodule
```

ソースコード 3.6: 7 セグメント LED に 1234 表示ソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////

3  // Company:
4  // Engineer:
5  //
6  // Create Date: 16:41:59 04/24/2019
7  // Design Name:
8  // Module Name: counter
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
```

```
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14:15:28 04/24/2019
// Design Name:
// Module Name: seg7proc
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module seg7proc(
    input [3:0] I,
    output reg [6:0] Seg
    );

        always @(*) begin
                case(I)
                        4'd0: begin Seg <= ~7'b0111111; end
                        4'd1: begin Seg <= ~7'b0000110; end
                        4'd2: begin Seg <= ~7'b1011011; end
                        4'd3: begin Seg <= ~7'b1001111; end
                        4'd4: begin Seg <= ~7'b1100110; end
                        4'd5: begin Seg <= ~7'b1101101; end
                        4'd6: begin Seg <= ~7'b1111101; end
                        4'd7: begin Seg <= ~7'b0000111; end
                        4'd8: begin Seg <= ~7'b1111111; end
                        4'd9: begin Seg <= ~7'b1101111; end
                endcase
        end
endmodule

module seg7(
        output [6:0]seg,
        input [3:0]I1,
        input [3:0]I2,
        input [3:0]I3,
        input [3:0]I4,
        output [3:0]K,
        input CLK);

        reg [3:0]D;
```

```
72          reg [15:0] TIM;
73          seg7proc u1(.I(D),.Seg(seg));
74
75          reg [1:0]cnt;
76          assign K = ~(4'b1 << cnt);
77          always @(posedge CLK) begin
78                  if(TIM == 32'd50000) begin
79                          cnt <= cnt + 1;
80                          case(cnt)
81                                  2'd3: begin D <= I1; end
82                                  2'd0: begin D <= I2; end
83                                  2'd1: begin D <= I3; end
84                                  2'd2: begin D <= I4; end
85                          endcase
86                          TIM <= 0;
87                  end else begin
88                          TIM <= TIM + 1;
89                  end
90          end
91  endmodule
92
93  module counter(
94           input CLK,
95           input BTN,
96       output [6:0]seg,
97           output [3:0]K
98       );
99          seg7 s(.seg(seg),.I1(4'd1),.I2(4'd2),.I3(4'd3),.I4(4'd4),.K(K),.CLK(CLK));
100 endmodule
```

ソースコード 3.7: 4 桁のカウンタ，10 進カウンタを組み合わせる方法のソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 16:41:59 04/24/2019
7  // Design Name:
8  // Module Name: counter
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////
21 `timescale 1ns / 1ps
22 //
       //////////////////////////////////////////////////////////////////////////////////
23 // Company:
```

```
// Engineer:
//
// Create Date: 14:15:28 04/24/2019
// Design Name:
// Module Name: seg7proc
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
    //////////////////////////////////////////////////////////////////////////////////

module seg7proc(
    input [3:0] I,
    output reg [6:0] Seg
    );

        always @(*) begin
                case(I)
                        4'd0: begin Seg <= ~7'b0111111; end
                        4'd1: begin Seg <= ~7'b0000110; end
                        4'd2: begin Seg <= ~7'b1011011; end
                        4'd3: begin Seg <= ~7'b1001111; end
                        4'd4: begin Seg <= ~7'b1100110; end
                        4'd5: begin Seg <= ~7'b1101101; end
                        4'd6: begin Seg <= ~7'b1111101; end
                        4'd7: begin Seg <= ~7'b0000111; end
                        4'd8: begin Seg <= ~7'b1111111; end
                        4'd9: begin Seg <= ~7'b1101111; end
                endcase
        end
endmodule

module seg7(
        output [6:0]seg,
        input [3:0]I1,
        input [3:0]I2,
        input [3:0]I3,
        input [3:0]I4,
        output [3:0]K,
        input CLK);

        reg [3:0]D;
        reg [15:0] TIM;
        seg7proc u1(.I(D),.Seg(seg));

        reg [1:0]cnt;
        assign K = ~(4'b1 << cnt);
        always @(posedge CLK) begin
                if(TIM == 32'd50000) begin
                        cnt <= cnt + 1;
                        case(cnt)
                                2'd3: begin D <= I1; end
                                2'd0: begin D <= I2; end
```

```
83                                  2'd1: begin D <= I3; end
84                                  2'd2: begin D <= I4; end
85                          endcase
86                          TIM <= 0;
87                  end else begin
88                          TIM <= TIM + 1;
89                  end
90          end
91  endmodule
92
93  module counter(
94          input CLK,
95          input BTN,
96      output [6:0]seg,
97          output [3:0]K
98      );
99
100         reg [3:0]D1=3'd0;
101         reg [3:0]D2=3'd0;
102         reg [3:0]D3=3'd0;
103         reg [3:0]D4=3'd0;
104         reg [15:0]TIM=15'd0;
105         reg pv;
106
107         seg7 s(.seg(seg),.I1(D4),.I2(D3),.I3(D2),.I4(D1),.K(K),.CLK(CLK));
108
109         always @(posedge CLK) begin
110                 if(TIM == 32'd50000) begin
111                         if(pv==0&&BTN==1) begin
112                                 if(D1 != 9)begin  D1 <= D1 + 1; end
113                                 else begin
114                                         D1 <= 0;
115                                         if(D2 != 9)begin D2 <= D2 + 1;end
116                                         else begin
117                                                 D2 <= 0;
118                                                 if(D3 != 9)begin D3 <= D3 + 1;end
119                                                 else begin
120                                                         D3 <= 0;
121                                                         if(D4 != 9) begin D4 <= D4
                                                                + 1;end
122                                                         else begin D4 <= 0;end
123                                                 end
124                                         end
125                                 end
126                         end
127                         pv <= BTN;
128                 end else begin
129                         TIM <= TIM + 1;
130                 end
131         end
132 endmodule
```

ソースコード 3.8: 4 桁のカウンタ，10000 進カウンタと除算器を組み合わせる方法のソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////

3  // Company:
4  // Engineer:
```

```
5  //
6  // Create Date: 18:23:15 05/03/2019
7  // Design Name:
8  // Module Name: counter
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////

21
22 // this module input valid in region of 0-11263
23 module div10(
24         input [13:0]D,
25         output [10:0]quotient,
26         output [3:0]remainder
27         );
28         wire [30:0]t;
29
30         assign t = D*3277; //2^12/10+1~13108, 13108>>2=3277
31         assign quotient = t[28:15];
32         assign remainder = ((t[14:10]+1)*5)>>4;//10=5*2
33 endmodule
34
35 module seg7proc(
36     input [3:0] I,
37     output reg [6:0] Seg
38     );
39
40         always @(*) begin
41                 case(I)
42                         4'd0: begin Seg <= ~7'b0111111; end
43                         4'd1: begin Seg <= ~7'b0000110; end
44                         4'd2: begin Seg <= ~7'b1011011; end
45                         4'd3: begin Seg <= ~7'b1001111; end
46                         4'd4: begin Seg <= ~7'b1100110; end
47                         4'd5: begin Seg <= ~7'b1101101; end
48                         4'd6: begin Seg <= ~7'b1111101; end
49                         4'd7: begin Seg <= ~7'b0000111; end
50                         4'd8: begin Seg <= ~7'b1111111; end
51                         4'd9: begin Seg <= ~7'b1101111; end
52                 endcase
53         end
54 endmodule
55
56 module prescaler(
57         input clk,
58         input [22:0]div,
59         output cout);
60
61         reg [23:0]d;
62         assign cout = d[23];
63         always @(posedge clk) begin
```

```
64                      d = d + div;
65              end
66      endmodule
67
68      module disp7seg(
69          input clk,
70                  input [13:0]num,
71                  output [6:0]led7segk,
72                  output reg [3:0]led7sega
73          );
74              reg [2:0]state=0;
75              reg [13:0]r1=0;
76              wire [10:0]quo;
77              wire [3:0]rem;
78
79         reg[3:0] I;
80
81              div10 d(.D(r1),.quotient(quo),.remainder(rem));
82              prescaler p(.clk(clk),.div(1000),.cout(pclk));
83              seg7proc s(.I(I),.Seg(led7segk));
84
85              always @(posedge pclk) begin
86                      if(state == 3'd0) begin //initial state
87                              r1 <= num;
88                              state <= 3'd1;
89                      end else if(state == 3'd1)begin
90                              I = rem;
91                              r1 <= quo;
92                              state <= state+1;
93                              led7sega <= ~(4'b1 << 3);
94                      end else if(state == 3'd2)begin
95                              I = rem;
96                              r1 <= quo;
97                              state <= state+1;
98                              led7sega <= ~(4'b1 << 2);
99                      end else if(state == 3'd3)begin
100                             I = rem;
101                             r1 <= quo;
102                             state <= state+1;
103                             led7sega <= ~(4'b1 << 1);
104                     end else begin
105                             I = rem;
106                             r1 <= num;
107                             state <= 3'd1;
108                             led7sega <= ~(4'b1 << 0);
109                     end
110             end
111     endmodule
112
113     module counter10000(
114             input CLK,
115             input BTN,
116             output [6:0]led7segk,
117             output [3:0]led7sega);
118
119             reg pbtn=0;
120             reg [13:0]num=0;
121             wire pclk;
122
123             disp7seg d(.clk(CLK),.num(num),.led7sega(led7sega),.led7segk(led7segk));
124             prescaler p(.clk(CLK),.div(1000),.cout(pclk));
```

```
125
126         always @(posedge(pclk)) begin
127                 if(pbtn == 0 && BTN == 1) begin
128                         if(num != 9999) begin
129                                 num <= num + 1;
130                         end else begin
131                                 num <= 0;
132                         end
133                 end
134                 pbtn <= BTN;
135         end
136
137 endmodule
```

ソースコード 3.9: ΔΣ 方式の調光ソースコード

```
1  `timescale 1ns / 1ps
2  //
       //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 17:04:52 04/29/2019
7  // Design Name:
8  // Module Name: deltashigma
9  // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //
       //////////////////////////////////////////////////////////////////////////////////
21
22
23
24 module prescaler(
25         input CLK,
26         input [10:0]MLn,
27         output Cout
28         );
29         reg[11:0]DA=0;
30
31         assign Cout = DA[11];
32         always @(posedge CLK) begin
33                 DA <= DA+MLn+12'd1;
34         end
35 endmodule
36
37 module deltasigma(
38     input CLK,
39     input [15:0] In,
40     output reg Out=0,
41          output reg [16:0]cnt = 0
```

```
42      );
43
44          wire pclk;
45          prescaler p(.CLK(CLK),.MLn(11'd500),.Cout(pclk));
46
47          always @(posedge CLK) begin
48                  if(cnt[16]) begin
49                          cnt <= cnt-17'h10000+In;
50                          Out <= 1;
51                  end else begin
52                          cnt <= cnt+In;
53                          Out <= 0;
54                  end
55          end
56  endmodule
57
58
59  module testpattern(
60          input CLK,
61          input [1:0]pattern,
62          output out
63          );
64          wire pclk;
65          reg [15:0]count=16'd0;
66
67          deltasigma d(.CLK(CLK),.In(count),.Out(out));
68
69          reg mode;
70          prescaler p(.CLK(CLK),.MLn(/*speed,0~2047*/11'd4),.Cout(pclk));
71                  always @(posedge pclk) begin
72                  if(pattern == 2'b00) begin
73                          if(mode) begin
74                                  if(count != 16'hFFFF) begin
75                                          count <= count + 1;
76                                  end else begin
77                                          mode <= 0;
78                                  end
79                          end else begin
80                                  if(count != 16'h0000) begin
81                                          count <= count - 1;
82                                  end else begin
83                                          mode <= 1;
84                                  end
85                          end
86                  end else if(pattern == 2'b01) begin
87                          count <= 16'h0;
88                  end else if(pattern == 2'b10) begin
89                          count <= 16'h6000;
90                  end else if(pattern == 2'b11) begin
91                          count <= 16'hFFFF;
92                  end
93          end
94  endmodule
```

参考文献

[1] Henry S. Warren, INTEGER DIVISION BY CONSTANT, Chapter 10, Hacker's Delight (2nd Edition) , 2003 年

[2] Hasselstrom, Karl, Fast Division of Large Integers: A Comparison of Algorithms, Master's in Computer Science thesis, Royal Institute of Technology, 1994 年