2022 年度 3 回生前期学生実験 HW

team02 機能設計仕様書

機能設計仕様書作成者: 伊藤舜一郎
グループメンバー：
伊藤舜一郎 (学籍番号:1029-32-7548)
植田健斗　(学籍番号:1029-32-6498)

提出期限：6 月 9 日 13 時 15 分　提出日: 2022 年 6 月 3 日

1　新たに作成した部品の仕様

1.1　fowadingUnit

1.1.1　回路の仕様

入力・出力

- input [2:0] IDEX_RegRd,EXMEM_RegRd,IFID_RegRs,IFID_RegRd
- input IDEX_RegWrite,EXMEM_RegWrite
- output [1:0] FwdA,FwdB

フォワーディングを行うかどうか判断をするためのユニット。1 ビットの信号 IDEX_RegWrite,EXMEM_RegWrite を受け取り、それが 1 のときでさらに、IFID_RegRs,IFID_RegRd の値が IDEX_RegRd,EXMEM_RegRd と等しいときにフォワーディングを行う。2 ビットの信号 FwdA,FwdB はマルチプレクサに用いられ、FwdA,FwdB の上位 1 ビットが 1 のときは IDEX_RegRd の値が、下位 1 ビットが 1 のときは EXMEM_RegRd の値がフォワーディングされる。

1.1.2　回路の設計

回路の仕様を満たすため、assign 部において、FwdA[1] = (IDEX_RegWrite)&(IDEX_RegRd==IFID_RegRs); とすることで、IDEX_RegWrite が 1 のときかつ IDEX_RegRd と IFID_RegRs の値が等しいとき FwdA の上位 1 ビットが 1 となるようにしている。これと同様の実装を FwdA の下位 1 ビット、FwdB でも行う。fowardingUnit の assign 部は以下のようになる。

ソースコード 1: fowardingUnit の assign 文

```
1 assign FwdA[1] = (IDEX_RegWrite)&(IDEX_RegRd==IFID_RegRs);
2 assign FwdA[0] = (EXMEM_RegWrite)&(EXMEM_RegRd==IFID_RegRs);
3 assign FwdB[1] = (IDEX_RegWrite)&(IDEX_RegRd==IFID_RegRd);
4 assign FwdB[0] = (EXMEM_RegWrite)&(EXMEM_RegRd==IFID_RegRd);
```

1.2　clockCounter

このモジュールは processor フォルダ内にある。

1.2.1　回路の仕様

入力・出力

- input [15:0] instruction,
- input systemRunning,clock,reset,
- output reg [7:0] SEG_X,SEG_Y,
- output reg [3:0] selectX,selectY

このモジュールはソートコンテストにおいてサイクル数を数えて 7 セグメント LED に出力するためのものである。1 ビットの systemRunning を受け取ってそれが 1 のときにカウントを開始して、16 ビットの命令を受け取ってそれが halt 命令でないならカウントを 1 足し続け、halt 命令ならカウントを 4 足してカウントをやめる。7 セグメント LED への出力は拡張ボード上で行わず、MU500-RXSET 上で行う。そのためにセレクタ信号 selectX,selectY と 7 セグメント LED を点灯させるための 8 ビットの信号 SEG_X,SEG_Y を出力するようにした。

1.2.2 回路の設計

下の図 1 の X の部分に出力の selectX,SEG_X が、Y の部分に出力の selectY,SEG_Y が対応するように設計、ピンアサインメントを行う。また、点灯のさせ方は externalOutput と同様に行う。

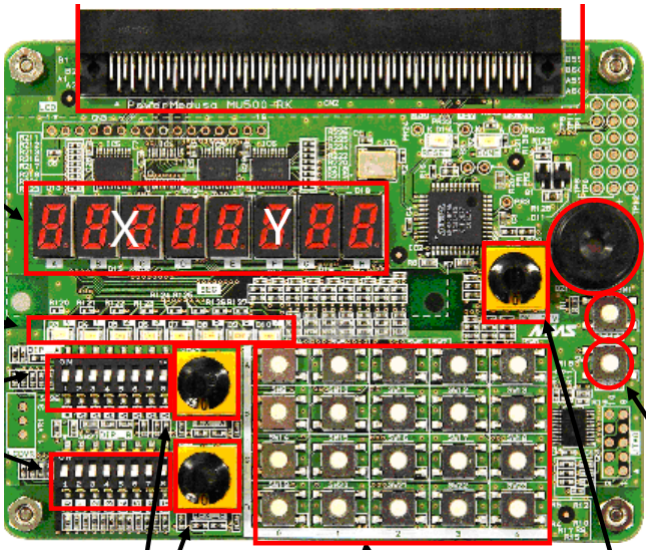

図 1: 点灯させる 7 セグメント LED

また、externalOutput と同様に、outFunc 関数は 4 ビットの数値を受け取って、その値に応じて 16 進数の値として 7SEGLED が点灯するように 8 ビットの数値を返す。

ソースコード 2: outFunc

```
1  function [7:0] outFunc;
2          input [3:0] a;
3          begin
4                  case (a)
5                  4'b0000:outFunc = 8'b1111_1100;
6                  4'b0001:outFunc = 8'b0110_0000;
7                  4'b0010:outFunc = 8'b1101_1010;
8                  4'b0011:outFunc = 8'b1111_0010;
9                  4'b0100:outFunc = 8'b0110_0110;
10                 4'b0101:outFunc = 8'b1011_0110;
11                 4'b0110:outFunc = 8'b1011_1110;
12                 4'b0111:outFunc = 8'b1110_0000;
13                 4'b1000:outFunc = 8'b1111_1110;
14                 4'b1001:outFunc = 8'b1111_0110;
15                 4'b1010:outFunc = 8'b1110_1110;
16                 4'b1011:outFunc = 8'b0011_1110;
17                 4'b1100:outFunc = 8'b0001_1010;
18                 4'b1101:outFunc = 8'b0111_1010;
```

```
19                  4'b1110:outFunc = 8'b1001_1110;
20                  4'b1111:outFunc = 8'b1000_1110;
21                  default:outFunc = 8'b0000_0000;
22 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣endcase
23 ␣␣␣␣␣␣␣␣end
24 endfunction
```

また、always 部は以下のようになっている。

ソースコード 3: always 部

```
1  reg [31:0] counter;
2  reg start;
3  reg stop;
4  reg [2:0] pattern1,pattern2;
5
6  always @(posedge clock) begin
7          if(reset) begin
8                  counter <= 0;
9                  start <= 1'b0;
10 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣stop ␣<= ␣1'b0;
11         end else begin
12                 if(systemRunning==1'b1&~start) ␣begin
13 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣start ␣<= ␣1'b1;
14                         counter <= counter + 1;
15                 end else if(({instruction[15:14],instruction[7:4]}==6'b11_1111)&~stop) ␣begin
16 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣stop ␣<= ␣1'b1;
17                         counter <= counter + 4;
18                 end else if(start&~stop) begin
19                         counter <= counter + 1;
20                 end
21         end
22
23         if (reset) begin
24                 selectX <= 4'b0000;
25 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣pattern1 ␣<= ␣3'b000;
26                 SEG_X <= outFunc(counter[19:16]);
27         end else if( pattern1 == 3'b000) ␣begin
28 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1110;
29                 pattern1 <= pattern1 + 1;
30         end else if( pattern1 == 3'b001) ␣begin
31 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1111;
32                 SEG_X <= outFunc(counter[31:28]);
33                 pattern1 <= pattern1 + 1;
34         end else if( pattern1 == 3'b010) ␣begin
35 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1101;
36                 pattern1 <= pattern1 + 1;
37         end else if( pattern1 == 3'b011) ␣begin
38 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1111;
39                 SEG_X <= outFunc(counter[27:24]);
40                 pattern1 <= pattern1 + 1;
41         end else if( pattern1 == 3'b100) ␣begin
42 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1011;
43                 pattern1 <= pattern1 + 1;
44         end else if( pattern1 == 3'b101) ␣begin
45 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1111;
46                 SEG_X <= outFunc(counter[23:20]);
47                 pattern1 <= pattern1 + 1;
48         end else if( pattern1 == 3'b110) ␣begin
49 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b0111;
50                 pattern1 <= pattern1 + 1;
51         end else if( pattern1 == 3'b111) ␣begin
52 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectX ␣<= ␣4'b1111;
53                 SEG_X <= outFunc(counter[19:16]);
54                 pattern1 <= pattern1 + 1;
55         end
56
57         if (reset) begin
58                 selectY <= 4'b0000;
```

```
59 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣pattern2␣<=␣3'b000;
60                     SEG_Y <= outFunc(counter[3:0]);
61             end else if( pattern2 == 3'b000)␣begin
62 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1110;
63                     pattern2 <= pattern2 + 1;
64             end else if( pattern2 == 3'b001)␣begin
65 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1111;
66                     SEG_Y <= outFunc(counter[15:12]);
67                     pattern2 <= pattern2 + 1;
68             end else if( pattern2 == 3'b010)␣begin
69 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1101;
70                     pattern2 <= pattern2 + 1;
71             end else if( pattern2 == 3'b011)␣begin
72 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1111;
73                     SEG_Y <= outFunc(counter[11:8]);
74                     pattern2 <= pattern2 + 1;
75             end else if( pattern2 == 3'b100)␣begin
76 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1011;
77                     pattern2 <= pattern2 + 1;
78             end else if( pattern2 == 3'b101)␣begin
79 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1111;
80                     SEG_Y <= outFunc(counter[7:4]);
81                     pattern2 <= pattern2 + 1;
82             end else if( pattern2 == 3'b110)␣begin
83 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b0111;
84                     pattern2 <= pattern2 + 1;
85             end else if( pattern2 == 3'b111)␣begin
86 ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣selectY␣<=␣4'b1111;
87                     SEG_Y <= outFunc(counter[3:0]);
88                     pattern2 <= pattern2 + 1;
89             end
90
91 end
```

counter はクロック数を数えるためのカウンタである。7〜21 行目で 1 ビットの systemRunning を受け取ってそれが 1 のときにカウントを開始して、16 ビットの命令を受け取ってそれが halt 命令でないならカウントを 1 足し続け、halt 命令ならカウントを 4 足してカウントをやめるように実装している。クロックが立ち上がったときに、reset 信号を受け取ったらカウンタやその他のレジスタを初期状態に戻す。systemRunnning が 1 になったら start を 1 にしてカウンタを 1 足しはじめ、start が 1 かつ stop が 0 のときはカウンタに 1 を足す。stop が 0 のときに halt 命令を受け取ったら stop を 1 にしてカウンタに 4 を足して、カウンタの増加をやめる。

また、23〜55 行目で X 部の LED の点灯を行っている。点灯の仕組みは以下のようになっている。

SEG_X にカウンタの上位 1〜4 ビットの値を格納→ 1 クロック後に X の上から 1 桁目が点灯するようにセレクタ信号を更新

↓

1 クロック後にセレクタ信号を 1111 にして何も点灯しない。SEG_X にカウンタの上位 5〜8 ビットの値を格納→ 1 クロック後に X の上から 2 桁目が点灯するようにセレクタ信号を更新

↓

1 クロック後にセレクタ信号を 1111 にして何も点灯しない...

これをループすることであたかも同時に各 LED が点灯して見えるようにしている。また、Y 部でも同様に 57〜89 行目で実装している。

2　性能評価

2.1　ALU

2.1.1　回路規模

ALU の回路規模は下図の図 2 のようになった。ここから Total logic elements は 171 で、Total pins は 56 であることが読み取れる。

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Jun 02 16:58:11 2022
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	alu
Top-level Entity Name	alu
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	171 / 28,848 (< 1 %)
Total registers	0
Total pins	56 / 329 (17 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

図 2: ALU の Flow Summary

2.1.2　遅延時間

TimeQuest Timing Analyzer の ReportDatasheet の一部は次の図 3 のようになった。この図から、遅延時間の最大値は 18.479ns であることがわかり、それは inB から SZCV への FR のときのパスであった。

Propagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	inB[0]	SZCV[2]	17.898	17.851	18.479	18.432
2	inB[3]	SZCV[2]	18.060	18.013	18.473	18.426
3	inB[1]	SZCV[2]	17.987	17.940	18.363	18.316
4	inB[4]	SZCV[2]	17.709	17.662	18.353	18.306
5	inB[2]	SZCV[2]	17.635	17.588	18.218	18.171
6	inB[5]	SZCV[2]	17.908	17.846	18.091	18.029
7	inB[6]	SZCV[2]	17.128	17.144	17.683	17.647
8	inB[12]	SZCV[2]	16.952	16.968	17.455	17.416
9	inB[0]	out[13]	16.898	16.738	17.408	17.232
10	inB[3]	out[13]	17.060	16.900	17.402	17.226
11	inB[10]	SZCV[2]	16.872	16.888	17.382	17.343
12	inB[1]	out[13]	16.987	16.827	17.292	17.116
13	inB[4]	out[13]	16.709	16.549	17.282	17.106
14	inB[9]	SZCV[2]	16.999	17.015	17.242	17.203
15	inB[5]	out[13]	17.030	16.870	17.213	17.053

図 3: ALU の遅延時間

2.1.3 考察

計算機の構成の講義で学んだように、ALU の計算時間は長く、遅延時間の最大値は 18.479ns となった。これを周期とする周波数は 54.1MHz となった。作成したプロセッサの動作可能な最大周波数は 60MHz 程度であるため、ALU の動作可能な最大周波数はプロセッサの動作可能な最大周波数に比べ小さくなることがわかるが、この原因はわからなかった。この ALU の遅延時間を短くなるように設計を改善することができればプロセッサの性能の向上や回路規模の縮小が見込まれると自分は考えた。

2.2 shifter

2.2.1 回路規模

shifter の回路規模は下図の図 4 のようになった。ここから Total logic elements は 277 で、Total pins は 44 であることが読み取れる。

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Jun 02 17:11:29 2022
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	shifter
Top-level Entity Name	shifter
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	277 / 28,848 (< 1 %)
Total registers	0
Total pins	44 / 329 (13 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

図 4: shifter の Flow Summary

2.2.2 遅延時間

TimeQuest Timing Analyzer の ReportDatasheet の一部は次の図 5 のようになった。この図から、遅延時間の最大値は 19.776ns であることがわかり、それは BR から SZCV への FR のときのパスであった。

2.2.3 考察

shifter の遅延時間の最大値は ALU の遅延時間の最大値よりも大きくなったのは予想外だった。Total logic elements も ALU より多く、回路規模の大きさに遅延時間は比例するのではないのだろうかと自分は考えた。また、ALU と同様に、shifter の動作可能な最大周波数はプロセッサの動作可能な最大周波数に比べ小さくなることがわかるが、この原因はわからなかった。また、shifter

Propagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	BR[4]	SZCV[2]	19.355	19.201	19.776	19.631
2	d[0]	SZCV[2]	19.014	18.860	19.448	19.303
3	BR[5]	SZCV[2]	18.869	18.715	19.246	19.101
4	BR[2]	SZCV[2]	18.818	18.664	19.231	19.086
5	BR[8]	SZCV[2]	18.405	18.251	18.822	18.668
6	BR[10]	SZCV[2]	18.295	18.141	18.795	18.641
7	BR[3]	SZCV[2]	18.356	18.202	18.754	18.609
8	BR[14]	SZCV[2]	18.242	18.088	18.737	18.583
9	BR[15]	SZCV[2]	18.095	17.941	18.537	18.392
10	d[2]	SZCV[2]	18.009	17.651	18.208	18.385
11	BR[12]	SZCV[2]	18.127	17.973	18.507	18.362
12	BR[9]	SZCV[2]	18.006	17.852	18.473	18.328
13	d[1]	SZCV[2]	18.028	17.883	18.437	18.283
14	BR[11]	SZCV[2]	17.929	17.775	18.420	18.275
15	BR[7]	SZCV[2]	17.875	17.721	18.313	18.168

図 5: shifter の遅延時間

の SZCV の計算はシフト演算と同じ関数で行っているためこれが遅延時間の原因なのではないのだろうかと自分は考えた。また、この shifter の遅延時間を短くなるように設計を改善することができればプロセッサの性能の向上や回路規模の縮小が見込まれると自分は考えた。

2.3 externalOutput

2.3.1 回路規模

externalOutput の回路規模は下図の図 6 のようになった。ここから Total logic elements は 999 で、Total pins は 92、Total registers は 325 であることが読み取れる。

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Jun 02 17:31:26 2022
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	externalOutput
Top-level Entity Name	externalOutput
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	999 / 28,848 (3 %)
Total registers	325
Total pins	92 / 329 (28 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

図 6: externalOutput の Flow Summary

2.3.2　周波数

TimeQuest Timing Analyzer にて Constrains でクロックを設定し、Flow Max Summary を確認する。設定するクロックの名前は clock として 50MHz とした。作成した externalOutput の Flow Max Summary は以下の図 7 のようになった。ここからこの回路が動作可能な最大周波数は 163.24MHz であることがわかる。

Slow 1200mV 100C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	163.24 MHz	163.24 MHz	clock	

図 7: clock の FMax Summary

2.3.3　クリティカルパス

また、Slow 1200mV 100C Model での Worst-Case Timing Paths を調べる。Summary of Paths の結果を 300 個出力し、これを Data Delay が大きい順にソートしたところ、下図の図 8 のようになり、最大値は 6.072ns となった。図からクリティカルパスはレジスタから点灯のためのアウトプット信号で発生していることがわかる。

Slow 1200mV 100C Model

Command Info | Summary of Paths

	Slack	Data Delay	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew
1	13.874	6.072	regM[10]	SEG_F[1]~reg0	clock	clock	20.000	-0.072
2	14.575	5.374	regI[2]	SEG_D[7]~reg0	clock	clock	20.000	-0.069
3	14.585	5.363	regA[14]	SEG_A[4]~reg0	clock	clock	20.000	-0.070
4	14.600	5.349	regI[2]	SEG_D[5]~reg0	clock	clock	20.000	-0.069
5	14.602	5.338	pattern[3]	SEG_G[2]~reg0	clock	clock	20.000	-0.078
6	14.700	5.250	regK[6]	SEG_C[7]~reg0	clock	clock	20.000	-0.068
7	14.702	5.242	pattern[3]	SEG_E[1]~reg0	clock	clock	20.000	-0.074
8	14.711	5.227	regF[8]	SEG_F[6]~reg0	clock	clock	20.000	-0.080
9	14.715	5.223	regF[11]	SEG_F[6]~reg0	clock	clock	20.000	-0.080
10	14.753	5.197	regI[6]	SEG_C[7]~reg0	clock	clock	20.000	-0.068
11	14.801	5.144	regI[11]	SEG_B[3]~reg0	clock	clock	20.000	-0.073
12	14.813	5.112	pattern[3]	SEG_A[6]~reg0	clock	clock	20.000	-0.093
13	14.820	5.126	regI[10]	SEG_B[3]~reg0	clock	clock	20.000	-0.072
14	14.861	5.077	regF[9]	SEG_F[6]~reg0	clock	clock	20.000	-0.080
15	14.867	5.081	regA[14]	SEG_A[1]~reg0	clock	clock	20.000	-0.070

図 8: clock のクリティカルパス

2.3.4　考察

拡張ボードのすべての 7 セグメント LED が点灯するように設計したため、Total registers は 325 ととても多くなっている。また、遅延時間の最大値 6.072ns を周期とする周波数は約 164MHz となる。これは Fmax Summary の値 163.24MHz とほぼ同じため、クリティカルパスを改善すれば

動作可能な最大周波数は大きくなると予測できる。作成したプロセッサの動作可能な最大周波数は60MHz 程度であるため、external の動作可能な最大周波数はプロセッサの動作可能な最大周波数に比べ非常に大きいことがわかる。

2.4 fowardingUnit

2.4.1 回路規模

fowardingUnit の回路規模は下図の図 9 のようになった。ここから Total logic elements は 8 で、Total pins は 18 であることが読み取れる。

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Jun 02 04:46:20 2022
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Editio
Revision Name	forwardingUnit
Top-level Entity Name	forwardingUnit
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	8 / 28,848 (< 1 %)
Total registers	0
Total pins	18 / 329 (5 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

図 9: fowardingUnit の Flow Summary

2.4.2 遅延時間

TimeQuest Timing Analyzer の ReportDatasheet は次の図 10 のようになった。この図から、遅延時間の最大値は 9.111ns であることがわかり、それは IFID_RegRd から FwdB への FF のときのパスであった。

2.4.3 考察

はじめは、if 文を用いて設計を行おうとしていたが、その場合モジュールを作成するとマルチプレクサを用いたような回路が設計されてしまい、この設計より Total logic elements は多くなるだろうと予測されるので、この設計は最適化されているのではないのだろうかと考えた。また、遅延時間の最大値 9.111ns を周期とする周波数は 109MHz となる。作成したプロセッサの動作可能な最大周波数は 60MHz 程度であるため、fowadingUnit の動作可能な最大周波数はプロセッサの動作可能な最大周波数に比べ大きいことがわかる。

Propagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	IFID_RegRd[1]	FwdB[0]	8.739	8.753	9.111	9.054
2	EXMEM_RegRd[0]	FwdB[0]	8.660	8.716	9.097	9.020
3	EXMEM_RegRd[1]	FwdB[0]	8.535	8.502	8.873	8.877
4	IFID_RegRd[0]	FwdB[1]	8.429	8.405	8.843	8.804
5	EXMEM_RegRd[0]	FwdA[0]	8.446	8.422	8.836	8.797
6	IFID_RegRd[0]	FwdB[0]	8.421	8.389	8.802	8.761
7	IFID_RegRs[1]	FwdA[1]	8.364	8.377	8.808	8.688
8	IFID_RegRd[1]	FwdB[1]	8.293	8.219	8.629	8.594
9	EXMEM_RegRd[1]	FwdA[0]	8.285	8.211	8.625	8.590
10	IFID_RegRs[1]	FwdA[0]	8.201	8.130	8.583	8.503
11	IFID_RegRs[0]	FwdA[0]	8.105	8.079	8.493	8.456
12	IDEX_RegRd[0]	FwdB[1]	8.087	8.061	8.478	8.441
13	IDEX_RegWrite	FwdB[1]	8.114			8.427
14	IDEX_RegRd[1]	FwdB[1]	8.143	8.072	8.503	8.423
15	EXMEM_RegRd[2]	FwdA[0]	8.095	8.078	8.507	8.417
16	IDEX_RegWrite	FwdA[1]	8.044			8.351
17	IDEX_RegRd[1]	FwdA[1]	8.074	7.999	8.434	8.350
18	EXMEM_RegRd[2]	FwdB[0]	8.010	7.950	8.347	8.323
19	IFID_RegRs[2]	FwdA[1]	7.978	7.955	8.389	8.309
20	IDEX_RegRd[0]	FwdA[1]	8.015	7.986	8.407	8.307
21	IFID_RegRs[2]	FwdA[0]	7.972	7.907	8.321	8.292
22	IDEX_RegRd[2]	FwdB[1]	7.910	7.846	8.250	8.222
23	IFID_RegRs[0]	FwdA[1]	7.859	7.783	8.213	8.174
24	IDEX_RegRd[2]	FwdA[1]	7.839	7.769	8.179	8.145
25	IFID_RegRd[2]	FwdB[0]	7.718	7.705	8.145	8.075
26	EXMEM_RegWrite	FwdA[0]	7.739			8.067
27	IFID_RegRd[2]	FwdB[1]	7.689	7.672	8.117	8.043
28	EXMEM_RegWrite	FwdB[0]	7.515			7.805

図 10: fowardingUnit の遅延時間

2.5 clockCounter

2.5.1 回路規模

clockCounter の回路規模は下図の図 11 のようになった。ここから Total logic elements は 197 で、Total pins は 43、Total registers は 55 であることが読み取れる。

2.5.2 周波数

TimeQuest Timing Analyzer にて Constrains でクロックを設定し、Flow Max Summary を確認する。設定するクロックの名前は clock として 100MHz とした。作成した clockCounter の Flow Max Summary は以下の図 12 のようになった。ここからこの回路が動作可能な最大周波数は 196.81MHz であることがわかる。

2.5.3 クリティカルパス

また、Slow 1200mV 100C Model での Worst-Case Timing Paths を調べる。Summary of Paths の結果を 300 個出力し、これを Data Delay が大きい順にソートしたところ、下図の図 13 のようになった。図からクリティカルパスは clock の値の足し算で発生していることがわかる。

Flow Summary

<<Filter>>

Flow Status	Successful - Thu Jun 02 05:00:39 2022
Quartus Prime Version	17.1.0 Build 590 10/25/2017 SJ Lite Edition
Revision Name	clockCounter
Top-level Entity Name	clockCounter
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	197 / 28,848 (< 1 %)
Total registers	55
Total pins	43 / 329 (13 %)
Total virtual pins	0
Total memory bits	0 / 608,256 (0 %)
Embedded Multiplier 9-bit elements	0 / 132 (0 %)
Total PLLs	0 / 4 (0 %)

図 11: clock の Flow Summary

clockCounter.v | Compilation Report - cloc

Slow 1200mV 100C Model Fmax Summary

<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	196.81 MHz	196.81 MHz	clock	

図 12: clock の FMax Summary

Slow 1200mV 100C Model

Command Info | Summary of Paths

	Slack	Data Delay	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew
1	4.919	5.026	counter[0]	counter[29]	clock	clock	10.000	-0.073
2	5.103	4.839	counter[3]	counter[29]	clock	clock	10.000	-0.076
3	5.135	4.812	counter[0]	counter[25]	clock	clock	10.000	-0.071
4	5.172	4.773	counter[0]	counter[31]	clock	clock	10.000	-0.073
5	5.186	4.758	counter[1]	counter[29]	clock	clock	10.000	-0.074
6	5.215	4.732	counter[0]	counter[30]	clock	clock	10.000	-0.071
7	5.233	4.709	counter[5]	counter[29]	clock	clock	10.000	-0.076
8	5.235	4.709	counter[3]	counter[30]	clock	clock	10.000	-0.074
9	5.252	4.690	counter[2]	counter[29]	clock	clock	10.000	-0.076
10	5.283	4.662	counter[0]	counter[27]	clock	clock	10.000	-0.073

図 13: clock のクリティカルパス

2.5.4 考察

性能評価報告書で述べたが、作成したプロセッサの動作可能な最大周波数は 60MHz 程度であるため、clockCounter の動作可能な最大周波数はプロセッサの動作可能な最大周波数に比べ非常に大きい。また、プロセッサのクリティカルパスに clockCounter の部分は含まれていなかったため、clockCounter を改善してもプロセッサの機能の向上は見込めないのではないのだろうかと自分は考えた。

また、clockCounter は各 7 セグメント LED の表示の切り替えに clock を用いているために、clock の値が大きくなると同時に表示されているように見えるが、周期も短くなるため表示が暗くなる。

拡張ボードでクロック数を表示すれば明るさの問題は解決するが、externalOutput の仕様やピンアサインメントを変更する必要があり面倒である。拡張ボードを使用せずにこの問題を解決する方法は何かないだろうかと自分は考えたが思いつかなかった。

3　実験の感想

計算機科学概論の講義でアセンブリ言語について学び、計算機の構成・計算機アーキテクチャの講義でプロセッサの仕組みについて説明を受けていたが、講義だけでは理解が進んでいなかった。今回、ハードウェア実験を通してプロセッサの作成、アセンブリプロセッサの作成を行うことでこれらの理解が深まった。また、他の人と実装を行う際にはコミュニケーションや認識のすり合わせがいかに大切であるかを学ぶことができたので良かった。