

・全体設計図のまとめ

・設計した命令や機能の説明(ブロック図、回路図)

まず、設計した命令セットアーキテクチャを表 1 に示す。減算命令および乗算命令まで実装した。

表 1 設計した命令

opcode	命令	機能の説明	データフロー
0000	MOV A,Imm	Aレジスタに即値を転送	$A \leftarrow Imm$
0001	ADD A,Imm	Aレジスタに即値を累積, キャリーフラグを更新	$A \leftarrow A + Imm$ , set C
0010	MOV A,B	AレジスタにBレジスタの値を転送	$A \leftarrow B$
0011	MOV A,IN	Aレジスタに入力ポートの値を転送	$A \leftarrow \text{入力ポート}$
0100	MOV B,Imm	Bレジスタに即値を転送	$B \leftarrow Imm$
0101	ADD B,Imm	Bレジスタに即値を累積, キャリーフラグを更新	$B \leftarrow B + Imm$ , set C
0110	MOV B,A	BレジスタにAレジスタの値を転送	$B \leftarrow A$
0111	MOV B,IN	Bレジスタに入力ポートの値を転送	$B \leftarrow \text{入力ポート}$
1000	MOV OUT,Imm	出力ポートに即値を転送	出力ポート $\leftarrow Imm$
1001	MOV OUT,B	出力ポートにBレジスタの値を転送	出力ポート $\leftarrow B$
1010	MOV B,GPR	GPR[Imm]の値をBレジスタに転送	$B \leftarrow GPR[Imm]$
1011	MOV GPR,B	Bレジスタの値をGPR[Imm]に転送	$GPR[Imm] \leftarrow B$
1100	SUB A,B	AレジスタからBレジスタの値を減算し, 結果をAレジスタに格納	$A \leftarrow A - B$
1101	MUL A,B	AレジスタとBレジスタの値を乗算し, 乗算結果の下位4ビットをBレジスタに, 上位4ビットをGPRの最上位アドレスに格納	$GPR[1111], B \leftarrow A \times B$
1110	JNC Imm	キャリーフラグが0のとき、即値の示す番地へ ジャンプ	If C=0, $PC \leftarrow Imm$ If C=1, No operation
1111	JMP Imm	即値の示す番地へジャンプ	$PC \leftarrow Imm$

表 1 にしたがって、PCU や ALU, GPR の設計を行った。

次に設計したブロック図および回路図を示す。

## PCU

### 1. データパス部

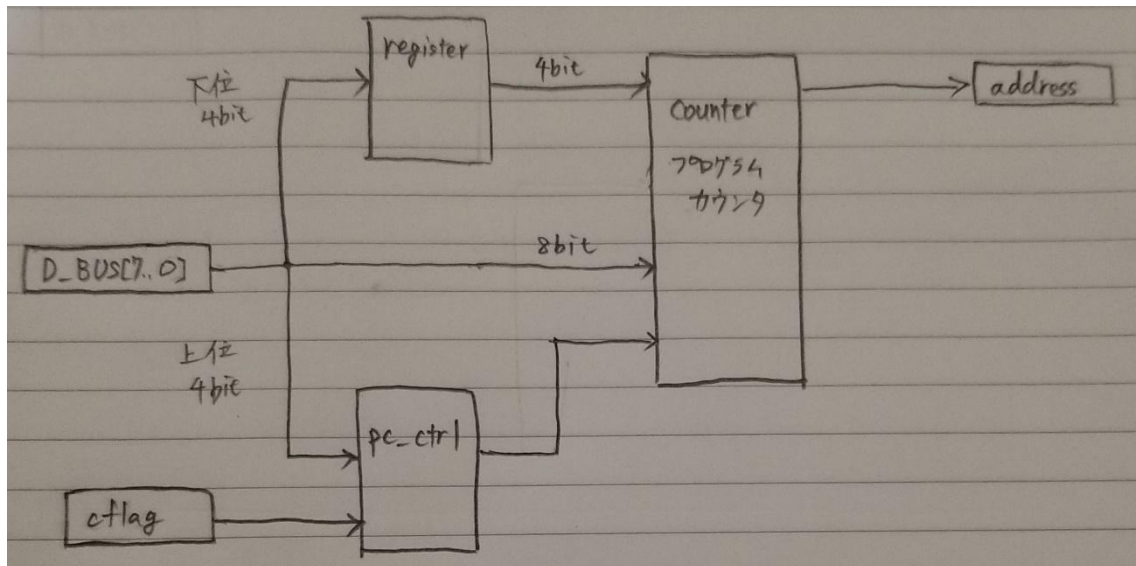


図 1 PCU のデータバス部のブロック図

図 1 に PCU のデータバス部のブロック図を示した。

それぞれのブロックの説明を行う。

register は D\_BUS の下位 4bit を受け取って格納しておくことにより、ジャンプ命令などで結合するアドレスを保存しておく役割があるブロックである。

pc\_ctrl は D\_BUS の上位 4bit を受け取って、PCU の状態によってプログラムカウンタのロード信号を生成するブロックである。

counter はプログラムカウンタであり、pc\_ctrl からのロード信号が 0 のときは  $pc \leftarrow pc+1$  の処理を行いアドレスを更新する。pc\_ctrl からのロード信号が 1 のときは、D\_BUS からの 8bit と register からの 4bit を結合してジャンプ先のアドレスを生成し、アドレスを更新する役割があるブロックである。

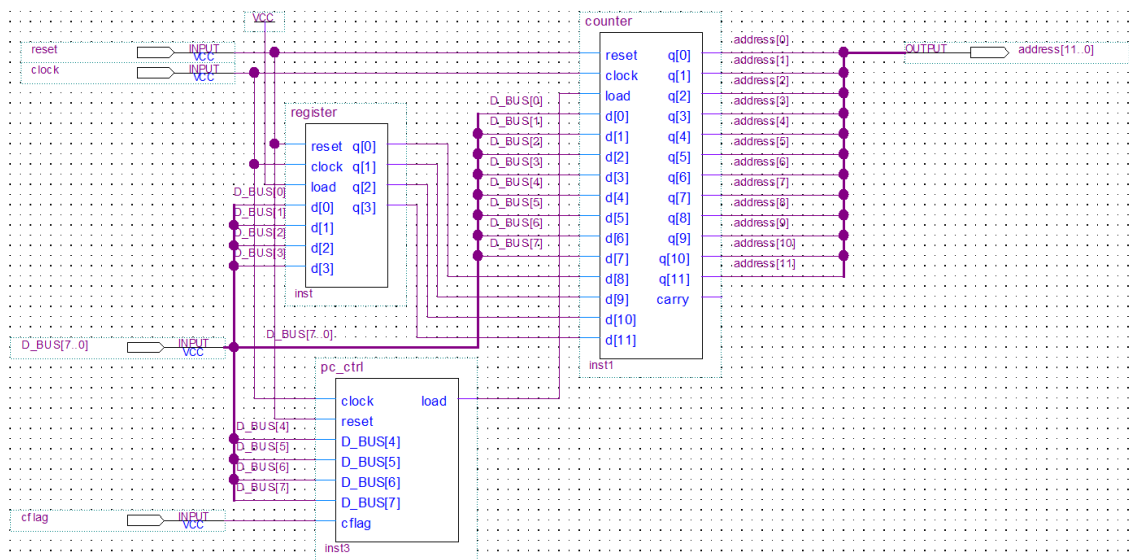


図 2 PCU のデータバス部の回路図

図 2 に PCU のデータバス部の回路図を示した。

## 2. 制御部

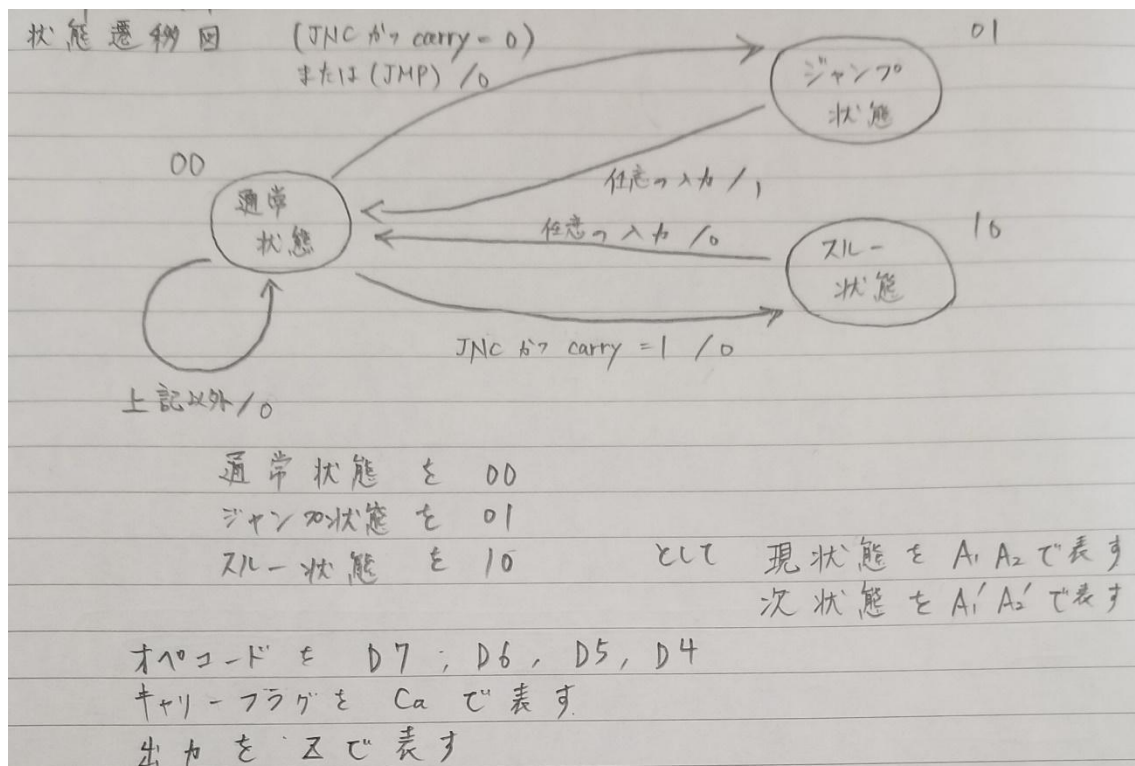


図 3 PCU の状態遷移図

図 3 に PCU の状態遷移図を示した。

表 2 PCU の状態遷移表

D7 D6 D5 D4 C <sub>a</sub>	A <sub>1</sub> A <sub>2</sub>	A <sub>1</sub> ' A <sub>2</sub> '	Z
1 1 1 0 0	0 0	0 1	0
1 1 1 0 0	0 1	0 0	1
1 1 1 0 0	1 0	0 0	0
1 1 1 0 1	0 0	1 0	0
1 1 1 0 1	0 1	0 0	1
1 1 1 0 1	1 0	0 0	0
1 1 1 1 X	0 0	0 1	0
1 1 1 1 X	0 1	0 0	1
1 1 1 1 X	1 0	0 0	0
それ以外	0 0	0 0	0
"	0 1	0 0	1
"	1 0	0 0	0

表 2 に PCU の状態遷移表を示した。表 2 より、 $Z = A_2$  とすればよいことが分かる。

表 3  $A_1'$  のカルノー図

$A_1' \backslash A_1 A_2$		Carry = 0																Carry = 1															
		D7	D6	D5	D4													D7	D6	D5	D4												
00	00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
01	01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
11	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
10	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			

表 3 に  $A_1'$  のカルノー図を示した。

これより  $A_1'$  の論理関数は

$$A_1' = \overline{A_1} \overline{A_2} D_7 D_6 D_5 \overline{D_4} C_a$$

表 4  $A_2'$  のカルノー図

$A_2'$	$A_1 A_2$	carry = 0								carry = 1							
		D7	D6	D5	D4	D3	D2	D1	D0	D7	D6	D5	D4	D3	D2	D1	D0
00	00	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
01	01	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11	11	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	10	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

表 4 に  $A_2'$  のカルノー図を示した。

これより  $A_2'$  の論理関数は

$$A_2' = \overline{A_1} \overline{A_2} D_7 D_6 D_5 \overline{D_4} \overline{C_a} + \overline{A_1} \overline{A_2} D_7 D_6 D_5 D_4$$

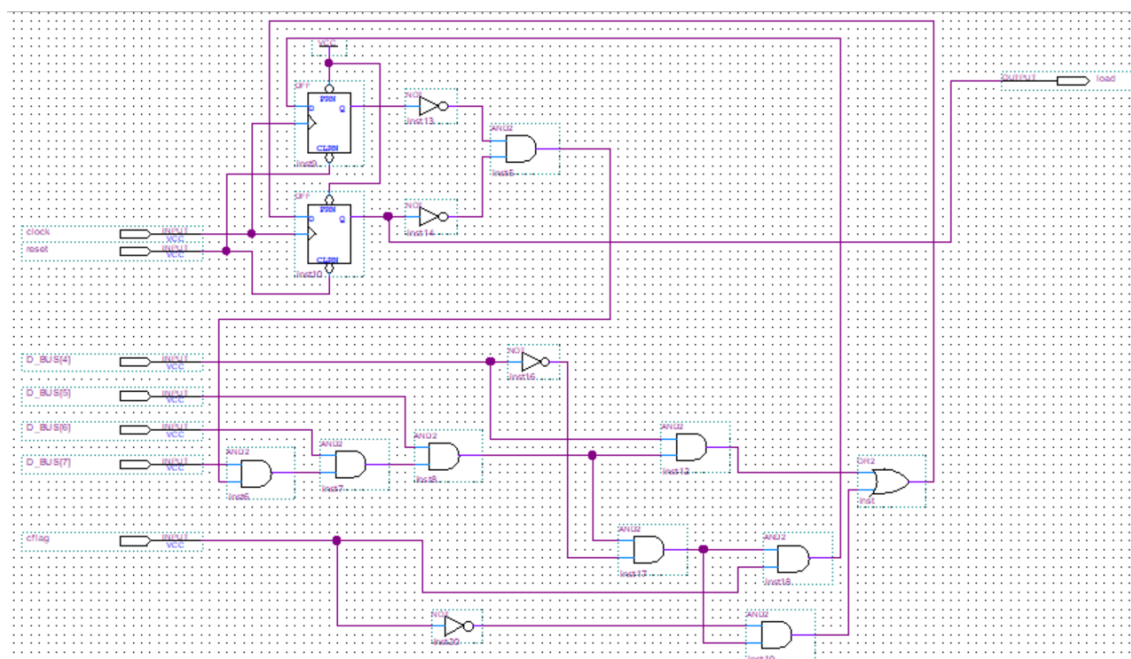


図 4 pc\_ctrl の回路図

設計した論理関数をもとに作成した、pc\_ctrl の回路図を図 4 に示した。

## ALU

### 1. データパス部

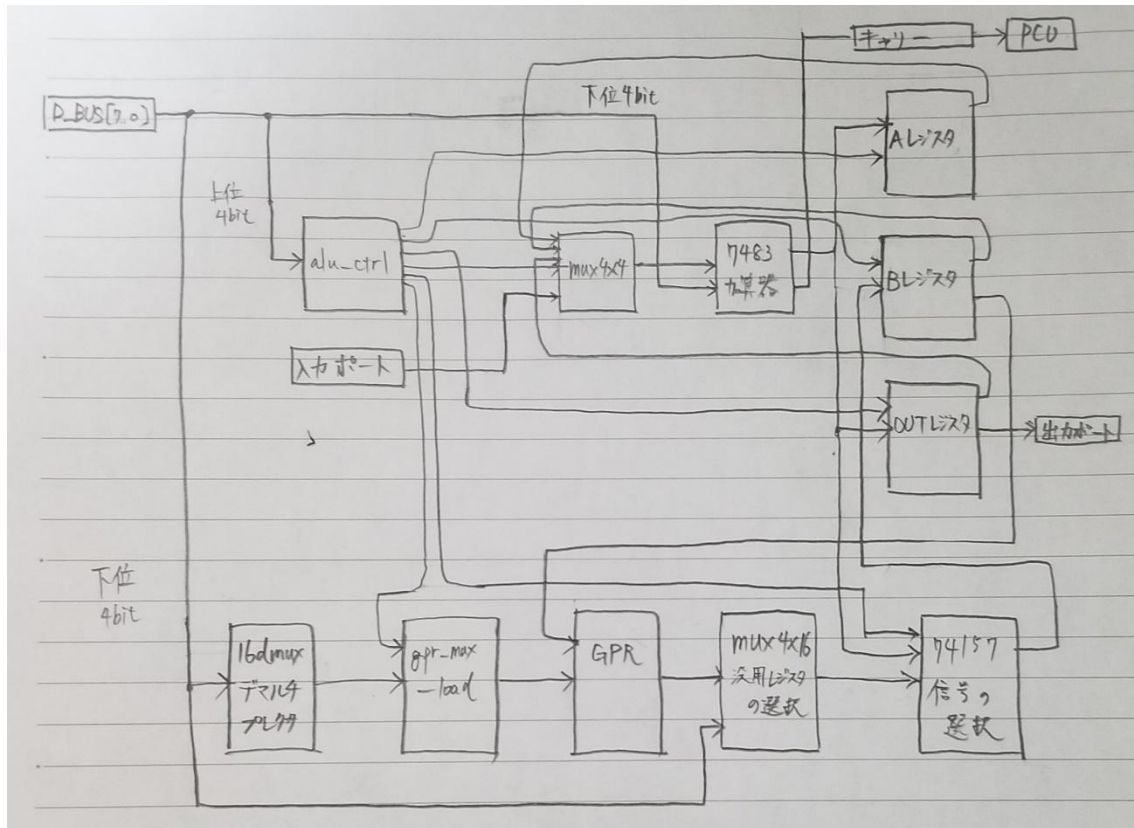


図 5 ALU データパス部のブロック図

図 5 に ALU データパス部のブロック図を示した。ブロック図の下半分に GPR の追加部分を記述している。

それぞれのブロックの説明を行う。

alu\_ctrl は D\_BUS の上位 4bit を受け取り、レジスタのロード信号や演算するレジスタを選ぶマルチプレクサの信号を生成するブロックである。

mux4x4 は alu\_ctrl で生成した信号により A レジスタ, B レジスタ, OUT レジスタ, 入力ポートの中から 1 つ選び、後続の加算器に流すブロックである。

7483 は mux4x4 の出力と即値を加算して後続に流すブロックである。

Aレジスタ,Bレジスタ,OUTレジスタは alu\_ctrl からのロード信号が1になったときに演算結果を格納するブロックである。

16dmux は D\_BUS の下位 4bit を受け取り、それを 16 進数の信号に変換するブロックである。

gpr\_mux\_load は MOV GPR,B (オペコード 1011)の命令の時に 1 になる信号と 16dmux の信号で論理積を取り、MOV GPR,B が実行されるときのみ 16dmux の出力信号を GPR に流すブロックである。

GPR は B レジスタの値を `gpr_mux_load` で選んだ 4bit レジスタに格納するブロックである。



mux4x16 は即値で対応する汎用レジスタを選ぶブロックである。

74157 は MOV B,GPR (オペコード 1010)の命令の時に mux4x16 を後続に流し、それ以外の時に加算器 7483 の信号を流すブロックである。

設計したブロック図に基づいて、実装した回路が図 2 である。

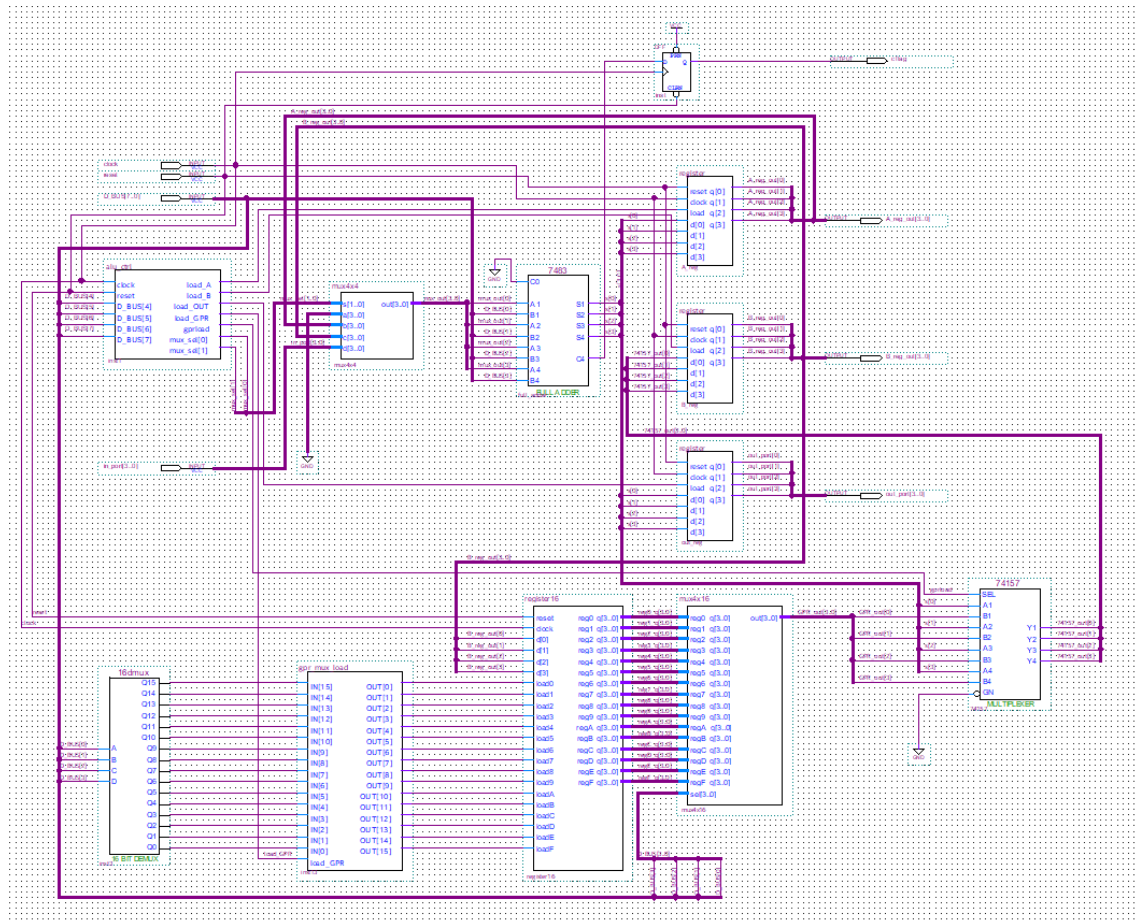


図 6 ALU のデータパス部の回路図

図 6 に実装した ALU のデータパス部の回路図を示した。

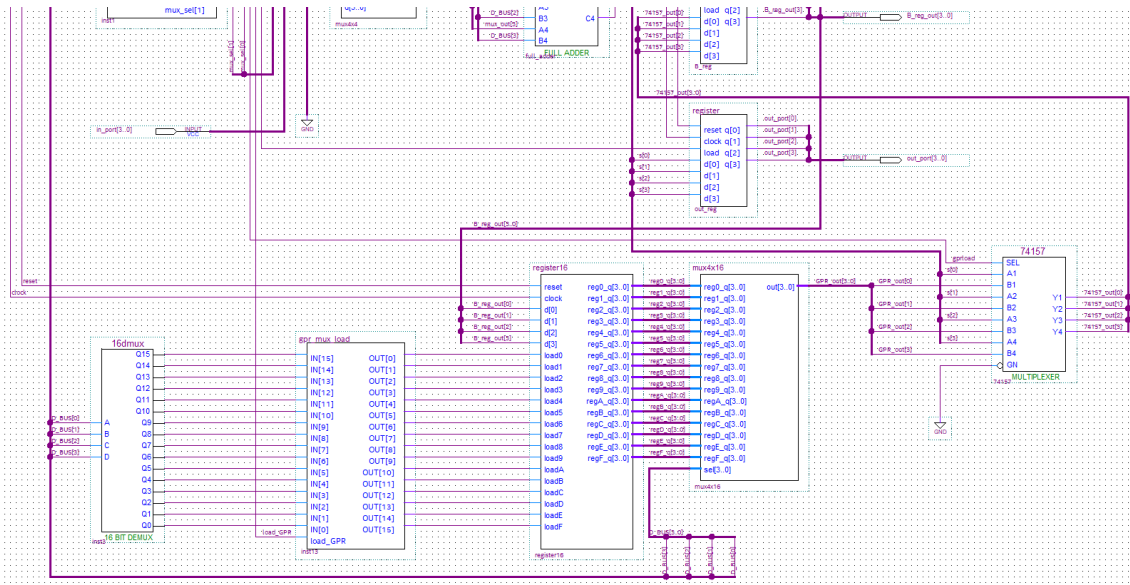


図 7 ALU のデータパス部の回路図の下半分を拡大したもの

図 7 に ALU のデータパス部の回路図の下半分を拡大したものを示した。GPR を追加した際に作成した部分である。

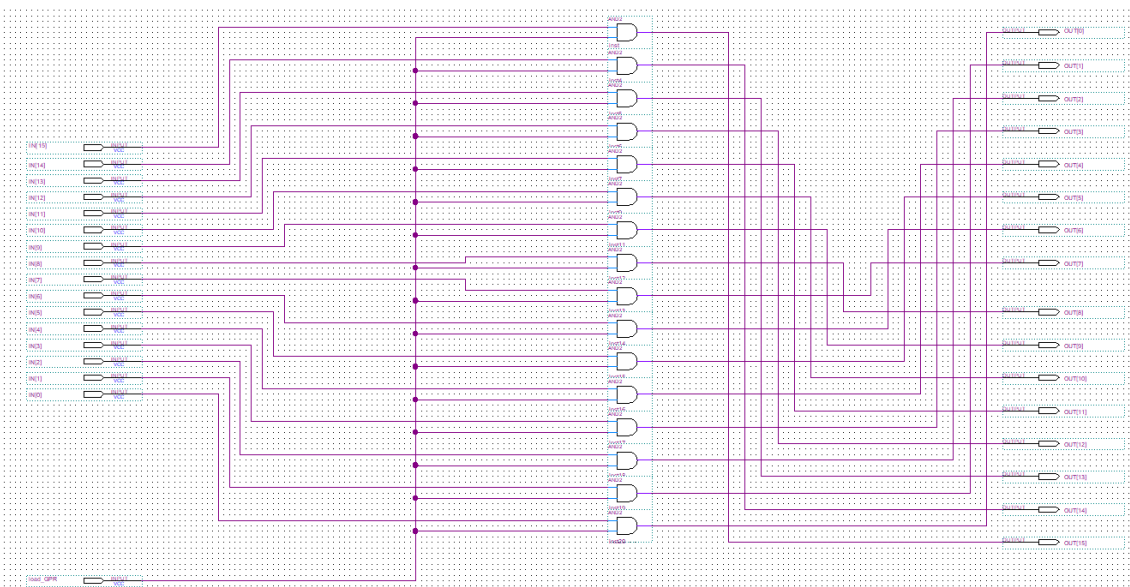


図 8 gpr\_mux\_load の回路図

図 8 に gpr\_mux\_load の回路図を示した。MOV GPR,B が実行されるときのみ 16dmux の出力信号を GPR に流している。

## 2. 制御部

alu\_ctrl の論理関数を示す。



まず、ALU の状態遷移図は次のようになる。

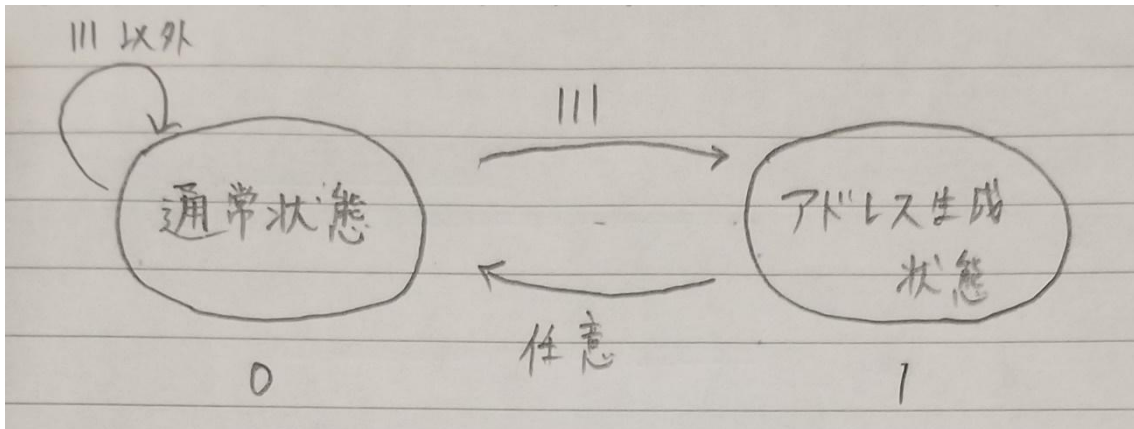


図 9 ALU の状態遷移図

図 9 に ALU の状態遷移図を示した。通常状態を 0, アドレス生成状態を 1 とする。通常状態にいるときにオペコードの上位 3bit が 111 のときにアドレス生成状態に遷移する。アドレス生成状態からは次のクロックで必ず通常状態に遷移する。

状態変数を  $S_t$  (通常状態 : 0, アドレス生成状態 : 1) とする。

$S_t'$  のカルノー図は

表 5 状態変数  $S_t'$  のカルノー図

$S_t'$		
$S_t$ \ $D_7 \sim D_5$	111	それ以外
0	1	0
1	0	0

表 5 より、 $S_t'$  の論理関数は

$$S_t' = \bar{S}_t D_7 D_6 D_5$$

表 6 それぞれの命令に対応して読み書きするレジスタ

opcode	書き込むレジスタ	読み込むレジスタ
0000	A	0
0001	A	A
0010	A	B
0011	A	In port
0100	B	0
0101	B	B
0110	B	A
0111	B	In port
1000	OUT	0
1001	OUT	B
1010	B	GPR[Imm]
1011	GPR[Imm]	B

表 6 にそれぞれの命令に対応して読み書きするレジスタを示した。この表よりそれぞれの load 信号と mux 信号を設計する。

load\_A のカルノー図は

表 7 load\_A のカルノー図

load-A		$D_7 \sim D_4$					
$S_t$		000	001	010	011	100	101
	0	1	1	1	1	0	0
1	0	0	0	0	0	0	0

表 7 より load\_A の論理関数は

$$load\_A = \overline{S_t} \overline{D_7} \overline{D_6}$$

load\_B のカルノー図は

表 8 load\_B のカルノー図

load\_B

$S_t \backslash D_7 \sim D_4$	000	010	011	111	100	その他
0	1	1	1	1	1	0
1	0	0	0	0	0	0

表 8 より load\_B の論理関数は

$$load\_B = \overline{S_t} \overline{D_7} D_6 + \overline{S_t} D_7 \overline{D_6} D_5 \overline{D_4}$$

load\_out のカルノー図は

表 9 load\_out のカルノー図

load\_out

$S_t \backslash D_7 \sim D_4$	000	010	その他
0	1	1	0
1	0	0	0

表 9 より load\_out の論理関数は

$$load\_out = \overline{S_t} D_7 \overline{D_6} \overline{D_5}$$

load\_GPR は即値で指定した GPR[Imm] に B レジスタの値を書き込む命令のときに 1 になる信号である。

load\_GPR に対応するのが MOV GPR,B (オペコード 1011)だから、カルノー図は

表 10 load\_GPR のカルノー図

load\_GPR

$S_t \backslash D_7 \sim D_4$	1011	それ以外
0	1	0
1	0	0

表 10 より load\_GPR の論理関数は

$$\text{load\_GPR} = \bar{S}_t D_7 \bar{D}_6 D_5 D_4$$

gprload は即値で指定した GPR[Imm]の値を B レジスタに書き込む命令のときに 1 になる信号である。

gprload に対応するのが MOV B,GPR (オペコード 1010)だから、カルノー図は

表 11 gprload のカルノー図

gprload

$S_t \backslash D_7 \sim D_4$	1010	それ以外
0	1	0
1	0	0

表 11 より gprload の論理関数は

$$\text{gprload} = \bar{S}_t D_7 \bar{D}_6 D_5 \bar{D}_4$$

S1, S0 は読み込むレジスタを選択するための 2bit のマルチプレクサの信号の上位ビットと下位ビットである。

S1 のカルノー図は

表 12 S1 のカルノー図

S1																	
St	D7~D4																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		0	0	1	1	0	1	1	0	0	0	0	0	0	1	1	0
1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 12 より S1 の論理関数は

$$\begin{aligned}
 S_1 &= \overline{S_t} \overline{D_7} \overline{D_6} D_5 + \overline{S_t} \overline{D_7} D_6 D_4 + \overline{S_t} D_7 \overline{D_6} D_4 \\
 &= load\_A \cdot D_5 + \overline{S_t} \overline{D_7} D_6 D_4 + \overline{S_t} D_7 \overline{D_6} D_4
 \end{aligned}$$

S0 のカルノー図は

表 13 S0 のカルノー図

S0																	
St	D7~D4																
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0		0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 13 より S0 の論理関数は

$$\begin{aligned}
 S_0 &= \overline{S_t} \overline{D_7} \overline{D_6} D_4 + \overline{S_t} \overline{D_7} D_6 D_5 \\
 &= load\_A \cdot D_4 + \overline{S_t} \overline{D_7} D_6 D_5
 \end{aligned}$$

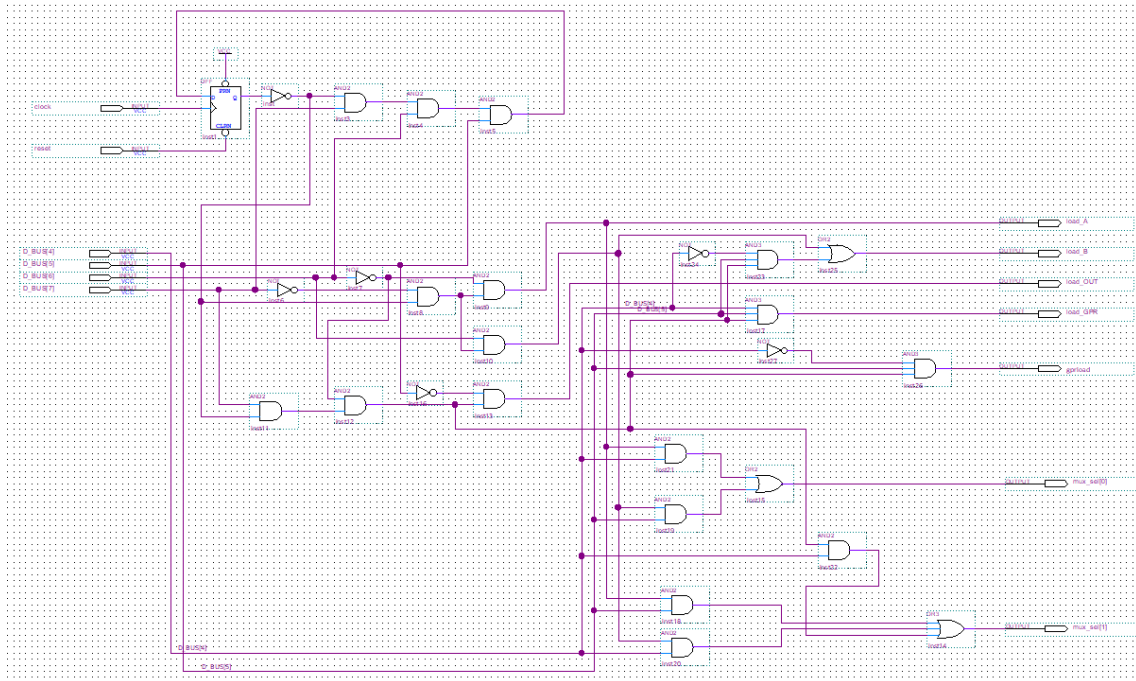


図 10 ALU の制御部 `alu_ctrl` の回路図

`alu_ctrl` の論理回路を図 10 に示した。表 5-表 13 で設計した論理関数をもとに、ALU の制御回路を作成した。

#### ・クリティカルパス遅延とロジックエレメント数の報告

##### 1. クリティカルパス(クロック周期を決めるパス)遅延

Acutual fmax(period)の値で評価

Slow 1200mV 100C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	89.58 MHz	89.58 MHz	clock	
2	134.99 MHz	134.99 MHz	clock_20mhz	

図 11 最大動作周波数

図 11 に最大動作周波数を示した。MCU4 の最大動作周波数は 89.58MHz であった。



Slow 1200mV 100C Model Setup: 'clock'								
<<Filter>>								
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	38.837	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst4	clock	clock	50.000	-0.060	11.101
2	38.909	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst7	clock	clock	50.000	-0.060	11.029
3	39.062	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst4	clock	clock	50.000	-0.060	10.876
4	39.134	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst7	clock	clock	50.000	-0.060	10.804
5	39.190	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst14 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst4	clock	clock	50.000	-0.059	10.749
6	39.262	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst14 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst7	clock	clock	50.000	-0.059	10.677
7	39.275	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst4	clock	clock	50.000	-0.056	10.667
8	39.339	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst6 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst4	clock	clock	50.000	-0.060	10.599
9	39.347	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst7	clock	clock	50.000	-0.056	10.595
10	39.441	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst7 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst5	clock	clock	50.000	-0.059	10.498
11	39.460	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst6 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst7	clock	clock	50.000	-0.060	10.478
12	39.479	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst5	clock	clock	50.000	-0.056	10.463
13	39.479	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst4	clock	clock	50.000	-0.056	10.463
14	39.479	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst6	clock	clock	50.000	-0.056	10.463
15	39.479	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst7	clock	clock	50.000	-0.056	10.463
16	39.498	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst5	clock	clock	50.000	-0.056	10.444
17	39.545	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst6	clock	clock	50.000	-0.060	10.393
18	39.553	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst5	clock	clock	50.000	-0.060	10.385
19	39.591	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst6	clock	clock	50.000	-0.056	10.351
20	39.634	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst6	clock	clock	50.000	-0.060	10.304
21	39.689	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst2 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst5	clock	clock	50.000	-0.068	10.241
22	39.704	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst5	clock	clock	50.000	-0.056	10.238
23	39.704	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst4	clock	clock	50.000	-0.056	10.238
24	39.704	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst6	clock	clock	50.000	-0.056	10.238
25	39.704	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg5 inst7	clock	clock	50.000	-0.056	10.238
26	39.728	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst10 inst2	CPU:CPU_1 ALU:inst_1 register:B_reg inst5	clock	clock	50.000	-0.062	10.208
27	39.735	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register:out_reg inst6	clock	clock	50.000	-0.067	10.196
28	39.736	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg7 inst5	clock	clock	50.000	-0.075	10.187
29	39.736	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg7 inst4	clock	clock	50.000	-0.075	10.187
30	39.736	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg7 inst6	clock	clock	50.000	-0.075	10.187
31	39.736	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg7 inst7	clock	clock	50.000	-0.075	10.187
32	39.764	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register:out_reg inst7	clock	clock	50.000	-0.067	10.167
33	39.775	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:gregE inst5	clock	clock	50.000	-0.076	10.147
34	39.775	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:gregE inst4	clock	clock	50.000	-0.076	10.147
35	39.775	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:gregE inst6	clock	clock	50.000	-0.076	10.147
36	39.775	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:gregE inst7	clock	clock	50.000	-0.076	10.147
37	39.778	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg4 inst5	clock	clock	50.000	-0.076	10.144
38	39.778	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 ALU:inst_1 register16:register16 register:greg4 inst4	clock	clock	50.000	-0.076	10.144

図 12 クリティカルパスの詳細

図 12 にクリティカルパスの詳細を示した。この図よりクリティカルパスが PCU 内のプログラムカウンタから出て、ROM を経由して ALU 内の registerB に到達するパスであることが読み取れる。

## 2. 回路規模

ロジックエレメント数で評価


Flow Summary	
 <<Filter>>	
Flow Status	Successful - Fri Dec 8 15:40:57 2023
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	board
Top-level Entity Name	board
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	744 / 28,848 ( 3 % )
Total registers	294
Total pins	113 / 329 ( 34 % )
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 13 Flow Summary

図 13 に MCU4 の Flow Summary を示した。これより、実装した論理素子数が 744 個で実装できる最大数の 28848 個に比べると 3%程度であったことが分かる。

プログラムの高速化として MCU4 の 2 段パイプライン化(実験 6.10)を行ったので、それについて報告する。使用したフォルダは MCU7c である。

パイプライン化の方針を示す。命令フェッチ(IF)と実行(EX)に段階を分けた 2 段パイプライン化を行った。具体的には ROM から命令を取り出して、いったん命令レジスタに命令を格納する。これが命令フェッチである。命令レジスタから取り出した命令を PCU と ALU に伝えて、PC の更新や演算を実行しレジスタに結果を格納する。これが実行である。その際に注意すべき点として、JMP と JNC でアドレスがジャンプするときにプログラムカウンタの挙動をパイプライン化前と変更する必要がある。

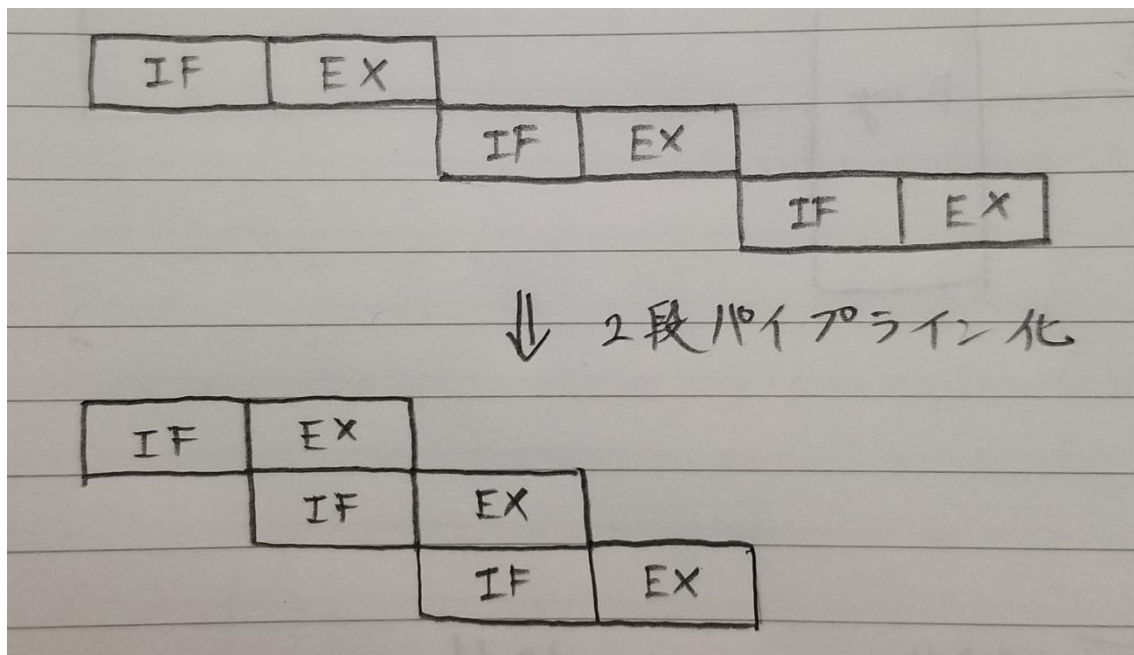


図 14 2 段パイプライン化後の命令実行の様子

図 14 にパイプライン化により、命令実行が効率化できている様子を示した。IF と EX を並列に動作するようにすることで、3 命令実行したいときに従来の  $\frac{2}{3}$  の時間で完了している。この極限を考えると、最初の IF と最後の EX を除いて命令が半分の時間で実行できるので、全体的見ると、理想的には約 2 倍の高速化が行えることになる。ただし、これは IF と EX の実行時間を等しいと仮定した場合の簡略図であり、実際には長時間かかる方の命令によって律速される点に注意する必要がある。

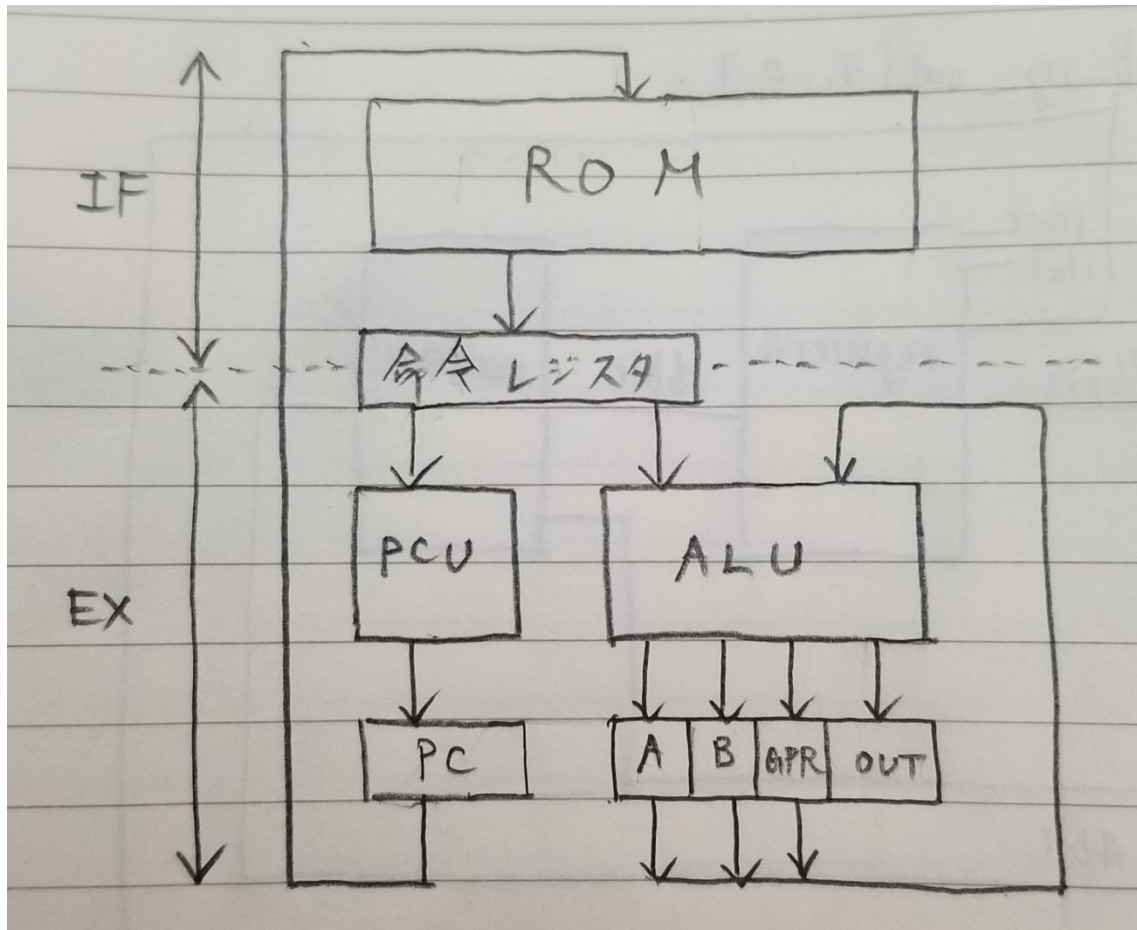


図 15 2 段パイプライン化後のブロック図

図 15 にパイプライン化後のブロック図を示した。ROM と PCU, ALU の間に命令レジスタを入れることにより、回路の機能を切り分けることで並列実行を可能にしている。

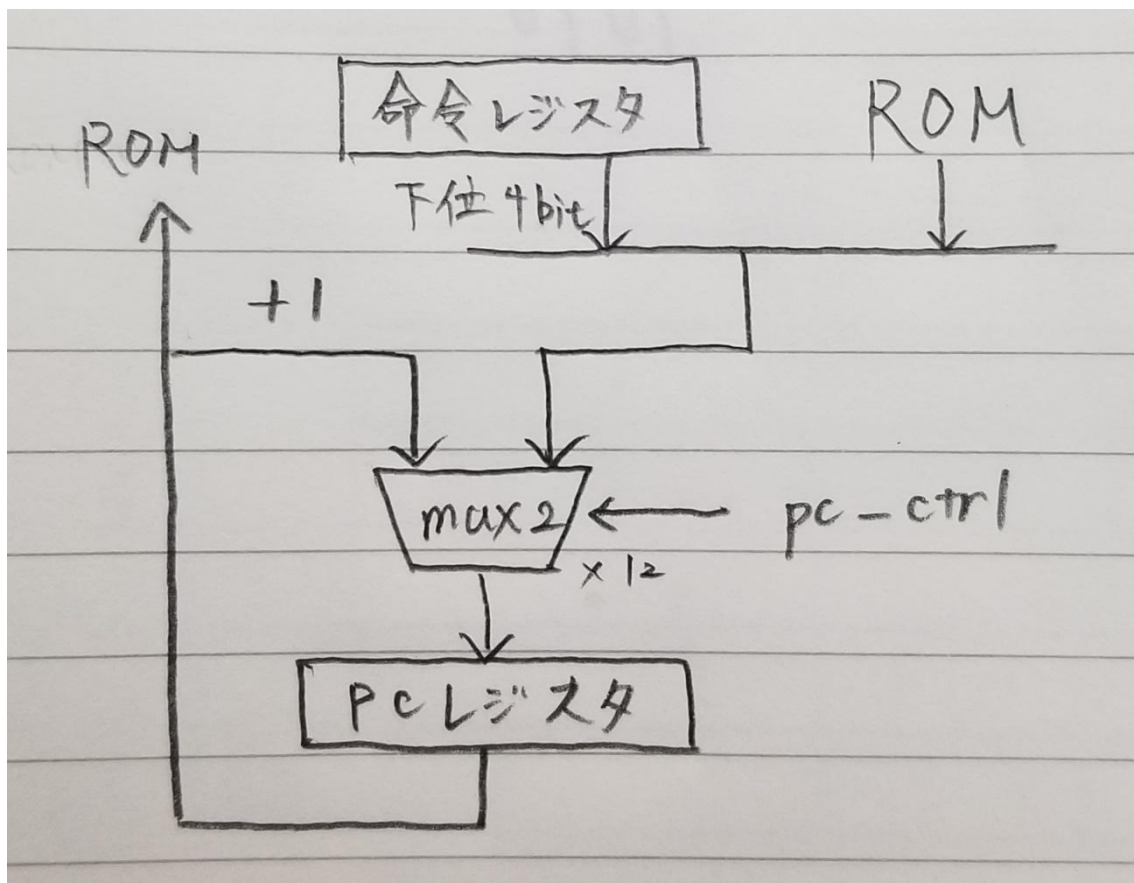


図 16 2 段パイプライン化後の PCU のブロック図

図 16 にパイプライン化後の PCU のブロック図を示した。通常状態のときは `pc_ctrl` の信号が 0 になるので、`mux2` が PC を 1 ずつ増加させる信号を選択し、PC レジスタを更新している。このブロック図において、アドレスをジャンプさせる命令が入ったときを考える。命令レジスタの上位 4bit および ALU から来るキャリーフラグによって、`pc_ctrl` がジャンプ状態に移るタイミングに注意する必要がある。このとき、命令レジスタに格納していた前の命令の下位 4bit と ROM から読みだしてきた 8bit を結合して 12bit のアドレスを得て、そのアドレスに即時ジャンプするように設計し直す必要がある。それにより、ALU とのタイミングをずらすことなくアドレスをジャンプさせることができることが分かった。

まとめると、パイプライン化の設計には命令レジスタを入れるだけでは不十分であり、PCU の再設計が必要であった。

実際に設計した回路図を示す。

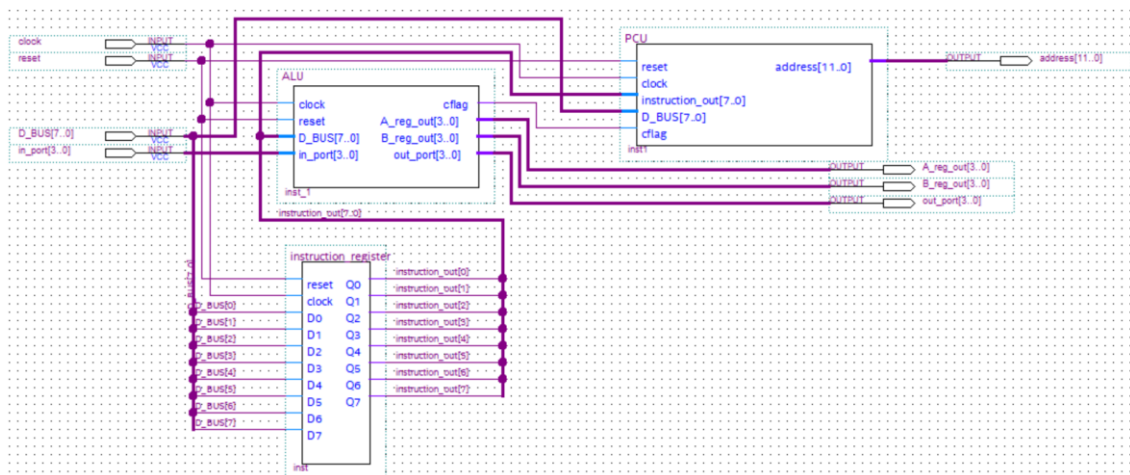


図 17 2 段パイプライン化後の CPU の回路図

図 17 に CPU の回路図を示した。命令レジスタを追加して、ALU には命令レジスタから、PCU には命令レジスタおよび ROM からの入力がある。

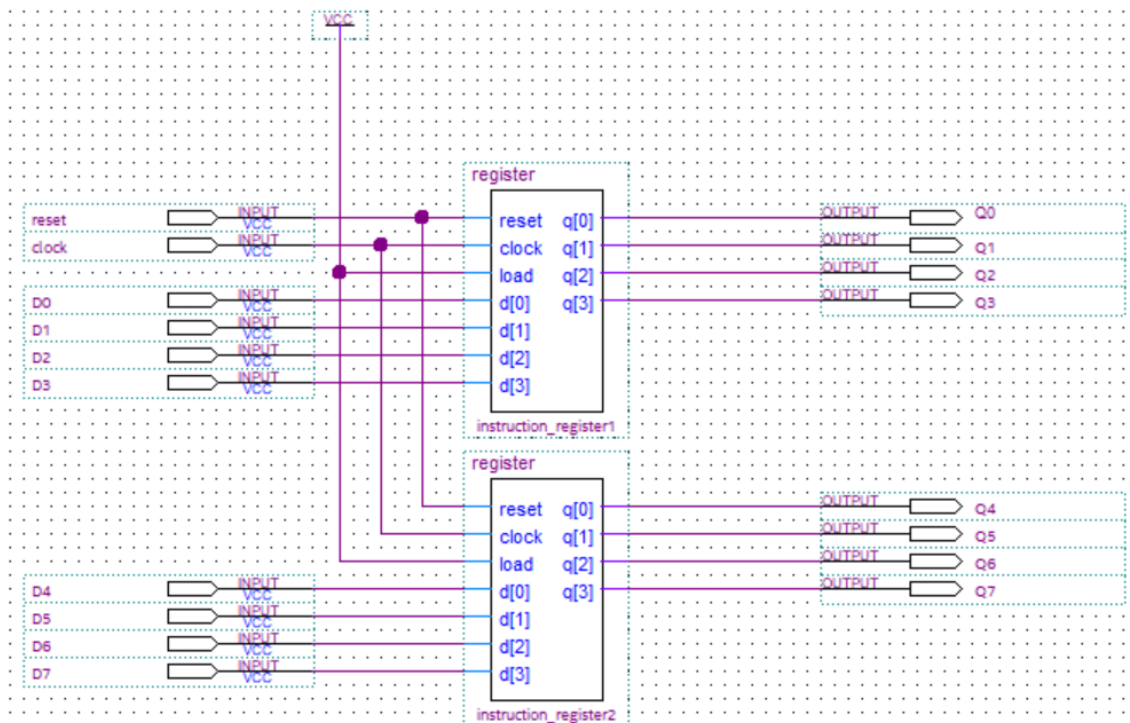


図 18 2 段パイプライン化後の命令レジスタの回路図

図 18 に命令レジスタの回路図を示した。内部は 4bit レジスタを 2 つ並べたものである。



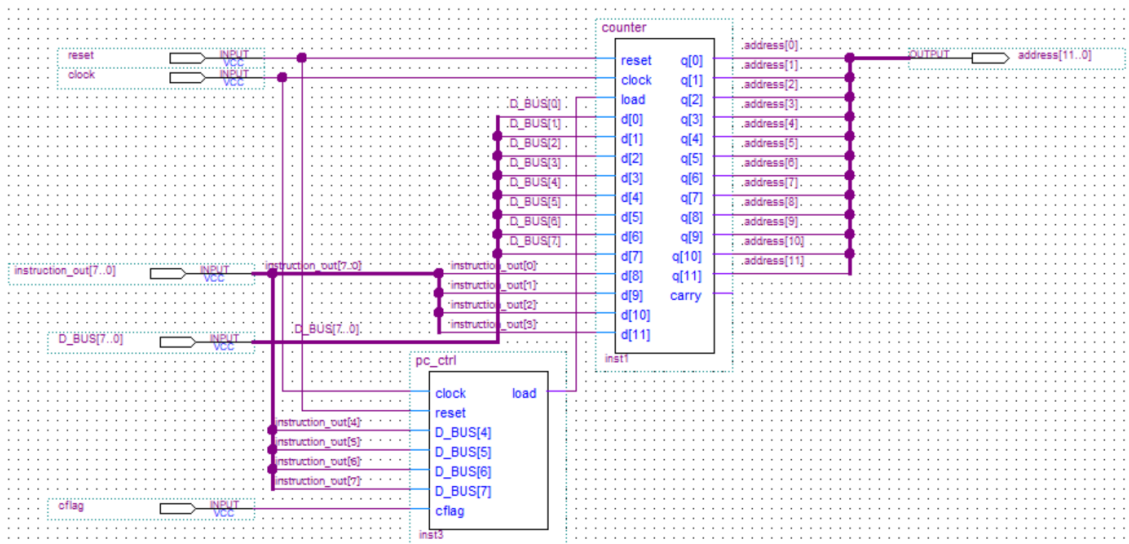


図 19 2 段パイプライン化後の PCU の回路図

図 19 には PCU の回路図を示した。pc\_ctrl には命令レジスタからの入力、プログラムカウンタには命令レジスタおよび ROM からの入力がある。

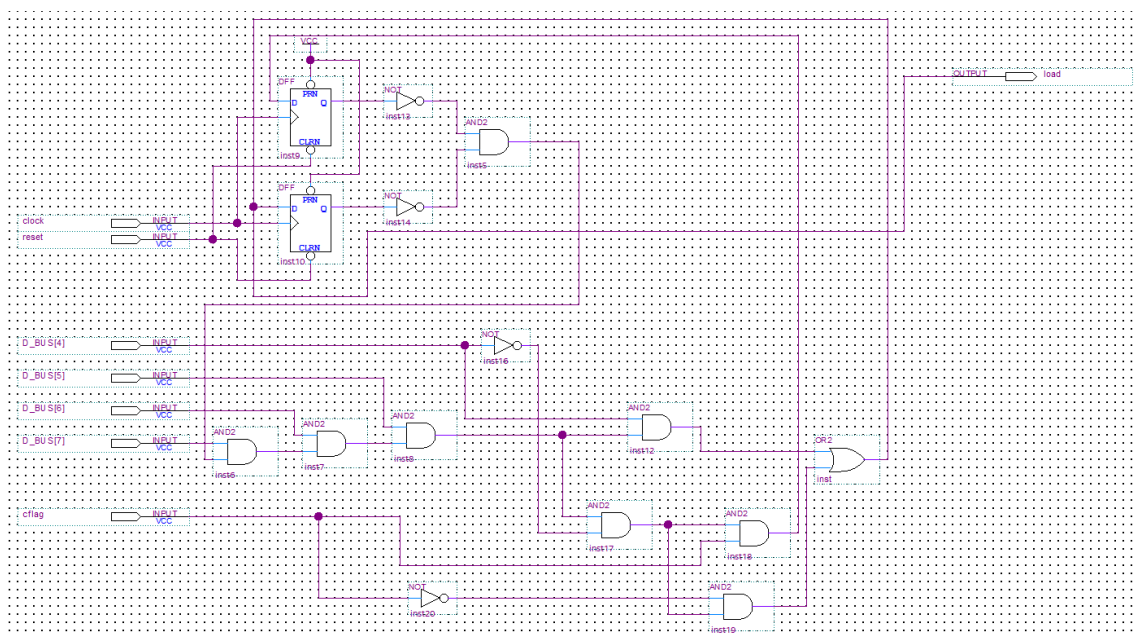


図 20 2 段パイプライン化後の pc\_ctrl の回路図

図 20 には pc\_ctrl の回路図を示した。変更点は、DFF を通過した後の信号を load 信号としていたものを、1 クロック前の DFF 通過する前のところから取るようにした。

次に、MCU4 のパイプライン化後のシミュレーション結果を示す。

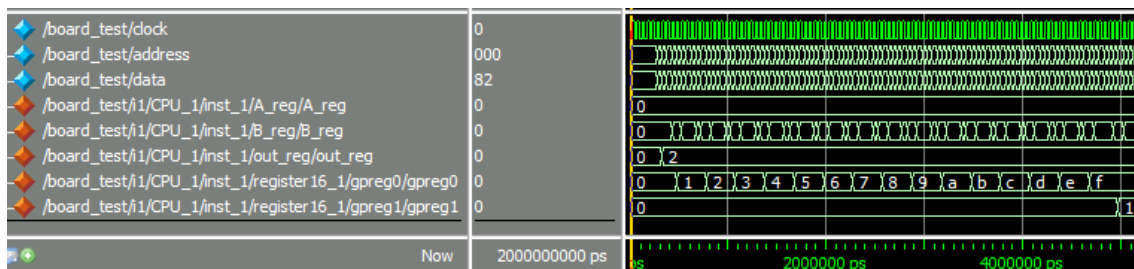


図 21 シミュレーション結果 1

図 21 で GPR0 が加算されていき、E→F に変わるとき GPR1 が加算されるというパイプライン化前のシミュレーション結果と一致した。

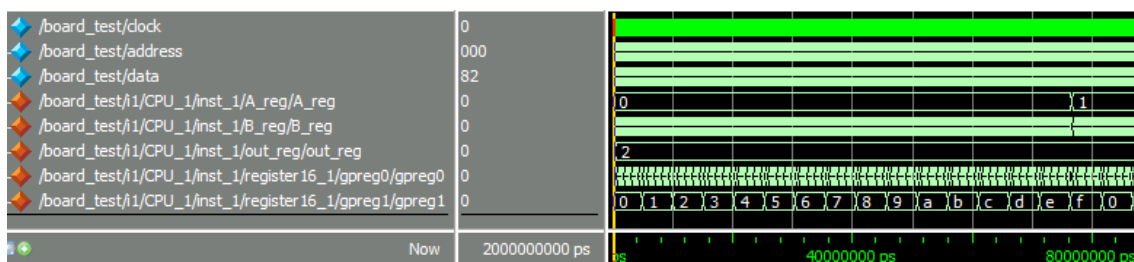


図 22 シミュレーション結果 2

図 22 で GPR1 が E→F に変わるとき regA が加算されるというパイプライン化前のシミュレーション結果と一致した。

図 21, 22 および 2 段パイプライン化した MCU6 での検証(使用したフォルダは MCU7), 実機ボードでのシミュレーション等の結果より、2 段パイプライン化が正しく実装できていることが確認できた。

最後に、パイプライン化後のクリティカルパス遅延とロジックエレメント数を示す。

#### 1. クリティカルパス遅延

Slow 1200mV 100C Model Fmax Summary				
<<Filter>>				
	Fmax	Restricted Fmax	Clock Name	Note
1	136.67 MHz	136.67 MHz	clock	
2	138.29 MHz	138.29 MHz	clock_20mhz	

図 23 最大動作周波数

図 23 に最大動作周波数を示した。これより、136.67MHz となり、パイプライン化前の

89.58MHz に比べると 47.09MHz だけ高速化できたことが分かる。これは図 14 で示したような理想的には 2 段パイプライン化の動作速度が 2 倍になる、というほどには高速化しなかった。それでも約 1.53 倍であったので、十分高速化できたといえる。

Slow 1200mV 100C Model Setup: 'clock'								
<<Filter>>								
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	42.683	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst6 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst5	clock	clock	50.000	-0.073	7.242
2	42.819	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst5	clock	clock	50.000	-0.072	7.107
3	42.865	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst6 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	clock	clock	50.000	-0.073	7.060
4	42.961	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst5	clock	clock	50.000	-0.074	6.963
5	43.001	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst9 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	clock	clock	50.000	-0.072	6.925
6	43.040	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 register.out_reg inst7	clock	clock	50.000	-0.077	6.881
7	43.068	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst5	clock	clock	50.000	-0.073	6.857
8	43.136	CPU:CPU_1 instruction_regist...instruction_register1 inst5	CPU:CPU_1 ALU:inst_1 register.B_reg inst7	clock	clock	50.000	-0.073	6.789
9	43.143	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	clock	clock	50.000	-0.074	6.781
10	43.250	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	clock	clock	50.000	-0.073	6.675
11	43.314	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 register.B_reg inst7	clock	clock	50.000	-0.074	6.610
12	43.354	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 register.B_reg inst6	clock	clock	50.000	-0.074	6.570
13	43.470	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst10 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst5 inst2	clock	clock	50.000	-0.074	6.454
14	43.482	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 register.out_reg inst6	clock	clock	50.000	-0.077	6.439
15	43.560	CPU:CPU_1 instruction_regist...instruction_register1 inst5	CPU:CPU_1 ALU:inst_1 register.B_reg inst4	clock	clock	50.000	-0.073	6.365
16	43.594	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst7 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst5 inst2	clock	clock	50.000	-0.073	6.331
17	43.596	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst10 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst4 inst2	clock	clock	50.000	-0.074	6.328
18	43.668	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst7 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst4 inst2	clock	clock	50.000	-0.073	6.257
19	43.743	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst14 inst2	clock	clock	50.000	-0.077	6.178
20	43.747	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst11 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst4 inst2	clock	clock	50.000	-0.074	6.177
21	43.783	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst2 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst14 inst2	clock	clock	50.000	-0.077	6.138
22	43.819	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst5 inst2	clock	clock	50.000	-0.074	6.105
23	43.828	CPU:CPU_1 instruction_regist...instruction_register1 inst5	CPU:CPU_1 ALU:inst_1 register.B_reg inst6	clock	clock	50.000	-0.073	6.097
24	43.871	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst8 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst5 inst2	clock	clock	50.000	-0.073	6.054
25	43.968	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst14 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst5	clock	clock	50.000	-0.070	5.960
26	44.017	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 register.A_reg inst7	clock	clock	50.000	-0.073	5.908
27	44.021	CPU:CPU_1 instruction_regist...instruction_register1 inst6	CPU:CPU_1 ALU:inst_1 register.B_reg inst5	clock	clock	50.000	-0.073	5.904
28	44.025	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 inst	clock	clock	50.000	-0.073	5.900
29	44.054	CPU:CPU_1 instruction_regist...instruction_register1 inst7	CPU:CPU_1 ALU:inst_1 register.B_reg inst5	clock	clock	50.000	-0.073	5.871
30	44.111	CPU:CPU_1 instruction_regist...instruction_register1 inst7	CPU:CPU_1 ALU:inst_1 register.B_reg inst4	clock	clock	50.000	-0.073	5.814
31	44.136	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst10 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst6	clock	clock	50.000	-0.074	5.788
32	44.140	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst7 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst5	clock	clock	50.000	-0.073	5.785
33	44.150	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst14 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	clock	clock	50.000	-0.070	5.778
34	44.165	CPU:CPU_1 instruction_regist...instruction_register2 inst6	CPU:CPU_1 ALU:inst_1 register.out_reg inst7	clock	clock	50.000	-0.077	5.756
35	44.198	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst3 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	clock	clock	50.000	-0.074	5.726
36	44.206	CPU:CPU_1 instruction_regist...instruction_register2 inst4	CPU:CPU_1 ALU:inst_1 register.out_reg inst5	clock	clock	50.000	-0.077	5.715
37	44.227	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst7 inst2	CPU:CPU_1 instruction_register.instruction_register1 inst6	clock	clock	50.000	-0.073	5.698
38	44.238	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst2 inst2	CPU:CPU_1 PCU:inst1 counter:inst1 count1:inst13 inst2	clock	clock	50.000	-0.074	5.686

図 24 クリティカルパスの詳細

図 24 にクリティカルパスの詳細を示した。これより、クリティカルパスが PCU 内のプログラムカウンタから、ROM を経由して命令レジスタに入るパスとなり、図 12 のパイプライン化前のクリティカルパスと比較すると変化していることが分かる。そのときの Slack は大きくなっているため、命令レジスタを入れてパイプライン化したことによるクリティカルパスの改善ができていることが読み取れる。

## 2. ロジックエレメントの総数


Flow Summary	
 <<Filter>>	
Flow Status	Successful - Fri Dec 8 15:53:47 2023
Quartus Prime Version	22.1std.2 Build 922 07/20/2023 SC Lite Edition
Revision Name	board
Top-level Entity Name	board
Family	Cyclone IV E
Device	EP4CE30F23I7
Timing Models	Final
Total logic elements	746 / 28,848 ( 3 % )
Total registers	298
Total pins	113 / 329 ( 34 % )
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 25 Flow Summary

図 25 にパイプライン後の Flow Summary を示した。これより、実装した論理素子数が 746 個で実装できる最大数の 28848 個に比べると 3%程度であった。図 13 よりパイプライン化前は論理素子数が 744 個であったため、ほとんど実装した論理素子数が増加していないことが分かる。しかしながら、パイプライン化により大幅に性能向上ができることが分かったので、回路規模をほとんど変えることなく性能向上できるパイプライン化のメリットの大きさを実感できた。