ネットワーク系演習計算機ハードウェア最終レポート

南田宗太郎 (28114124)

2018年7月25日

1 レポート課題7

1.1 7-1 の課題について

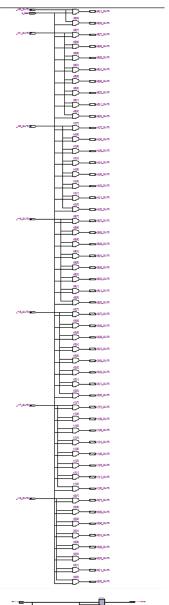
八個の8ビットレジスタを持つように拡張したソースコードを以下に記述する。

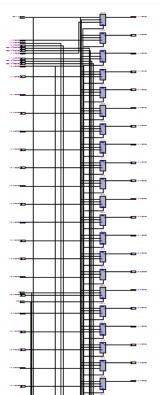
```
SUBDESIGN reg_d
   a_sel[2..0], b_sel[2..0], c_sel[2..0], c_load : INPUT;
   a_out[7..0], b_out[7..0] : OUTPUT;
   c_in[7..0],clk,reset : INPUT;
)
VARIABLE
r[7..0][7..0] : DFFE;
BEGIN
r[][].(clk,clrn) = (clk,!reset);
CASE a_sel[] IS
WHEN 0 => a_{out}[] = r[0][].q;
WHEN 1 => a_{out}[] = r[1][].q;
WHEN 2 => a_{out}[] = r[2][].q;
WHEN 3 => a_{out}[] = r[3][].q;
WHEN 4 \Rightarrow a_{out}[] = r[4][].q;
WHEN 5 => a_{out}[] = r[5][].q;
WHEN 6 => a_{out}[] = r[6][].q;
```

```
WHEN 7 \Rightarrow a_{out}[] = r[7][].q;
END CASE;
CASE b_sel[] IS
WHEN 0 => b_{out}[] = r[0][].q;
WHEN 1 => b_{out}[] = r[1][].q;
WHEN 2 => b_{out}[] = r[2][].q;
WHEN 3 => b_{out}[] = r[3][].q;
WHEN 4 \Rightarrow b_{out}[] = r[4][].q;
WHEN 5 => b_{out}[] = r[5][].q;
WHEN 6 => b_{out}[] = r[6][].q;
WHEN 7 \Rightarrow b_{out}[] = r[7][].q;
END CASE;
FOR i IN O TO 7 GENERATE
r[i][].d = c_in[];
END GENERATE;
CASE c_sel[] IS
WHEN 0 \Rightarrow r[0][].ena = c_load;
WHEN 1 \Rightarrow r[1][].ena = c_load;
WHEN 2 \Rightarrow r[2][].ena = c_load;
WHEN 3 \Rightarrow r[3][].ena = c_load;
WHEN 4 \Rightarrow r[4][].ena = c_load;
WHEN 5 \Rightarrow r[5][].ena = c_load;
WHEN 6 \Rightarrow r[6][].ena = c_load;
WHEN 7 \Rightarrow r[7][].ena = c_load;
END CASE;
END;
```

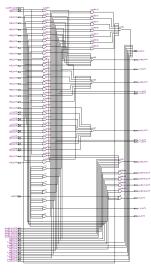
1.2 7-2 の課題について

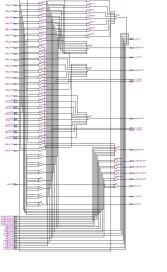
ブロック図を以下に添付する。

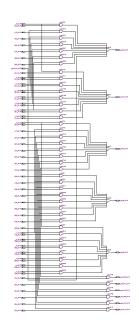


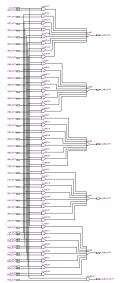












1.3 7-3 の課題について

以下にシュミレーションの結果の波形図を添付する。

波形図 1 では c_sel に 1,b_sel に 3,a_sel に 1,c_in に 12 を入力している。今回 c_sel で 1 を選んだの でレジスタ番号 1 のレジスタに 12 が入力され、a_sel に 1 が入力されているため a_out にはレジス タ番号 1 に格納されている 12 が出力され b_sel に 3 が入力されているためレジスタ番号 3 に格納 されている 0 が出力されることが期待される結果である。波形図 1 より期待される結果が正しく 出力されていることがわかる。

波形図 1 では c_sel に 1,b_sel に 3,a_sel に 1,c_in に 12 を入力している。波形図 1 とは違い a_sel,b_sel どちらとも 1 が入力されているため、a_out,b_out どちらも出力されるものは 12 が期待 される結果である。波形図 2 より期待される結果が正しく出力されていることがわかる。 波形図 3 は c load の入力を 0 にしたものである。 c load が 0 なので a out, b out どちらの出力も 0 が期待される結果である。波形図 3 より期待される結果が正しく出力されていることがわかる。

			-	rs 15	Įm 21.	fre N	ýra 43.	m 81.	Ç no 80	Pre 191	Inc. Militar
	Nem		203 n		39.	(re X	1				
690	Bas	e.	A [12	FI	39			[1]	0		
97	1 [#00	40								
01	1 1	×(0)	A0								
494	1 1	400	A.0								
91	1 [w(2)	AI								
@1	1 1	wC0	A.0								
(9°1		400	And					0			
ESP-11	ı F	×63	A.0								
@-11	1 1	e01	A.0								
EP 12	H bo	add.	ABO					00			
@1L	l F	wD3	A.0								
@ 15	1 1	×00	6.0								
69.12	1	e(0)	40								
© 11	1 1	w(D)	A.0								
@ 13	1 1	w(D)	A.9								
49-21	{ []	v(00	6.0								
€P22	Bby	el	AIX					00			
ESP-22	ł t	+01	A.0								
EEP-25	l L	# ES	Al								
ED 25	Boy		A [12					DEI			
EB-27	1 [Jedist	60								
EB-23	1 -	initi	A.0								
ED-13	ł t	.in[4]	A.0								
ESP-02	1 1	in\$2\$	Al								
EEP-33	1 +	in[1]	A.0								
EBP-15	1	.edit	61								
(SP-15	Bo	al	A DO								
EB-27	ł ŀ	×63	A.0								
ED-18	1 [:	×60	21								
Graph Grap	B book and a second a second and a second and a second and a second and a second an	. 1									
ED-41	1 100	er.	A D								
_	_	_	_								
	Nome	. 1	un.	ps 18.	0 au 20 20 20 21 21 21 21 21 21 21 21 21 21 21 21 21	7rs 10	f so 60:	ns 53.	80 56	Q ra 70.1	na 960 na
200	Name of the control o		A D2 A A A A A A A A A A A A A A A A A A		400						
971 972 973 973 973 973 971 971 971 971 971 971 971 971 971 971	1 1 1 P	M(7)	A B		.480			10			
(9 /2	I F	46(6)	AI								
691	l t	u(0)	A11								
91	1 1	MOD	A1								
69 5	1 -	L6(3)	A1								
697	l t	H00	A11								
Ø 3	BAR	el	ADD					DI			
gp-11	l 1-	400	A1								
EEP-12	t:	e00	Ail								
6911	Bloo	st.	A Dia		X				er e		
914	t-	46(7) 400)	A11								
Ø11	F:	M00	AI								
@17:	1 -	MIG	61								
6911	1	14(3)	211								
GP 21	1 -	DDw	A1								
@21		M00	A1					0			
ED-27	100	e(2)	ALL								
£24	1 -	e01	A1								
EP-25		e00	A 111					[0]			
EB-27	T E	x(2)	AI					100			
EB-21	l -	rD0	A1 F								
ESP-29	1	P-00	211								
gg=31	1 -	ir(D)	A1								
EEP-32	h	inD3	41								
EB-24	[E	200	Ail								
gg=35	0,6	ad	A1								
GP-28	82	el eD0	ALC					DI			
gp-20	1 1	4E0	AI								
gg-35	1 :	e(0)	A1								
(P-11	rese	-1	41						_		
	1		- 1								
	Name		10 m - P	19 10) re 29.	(m) (iii)re J	(no 4).	rs 58.	(m H	ýre 393	ino SE() no
	_	_	140			-		- 01			
691	l" K	utiti	AB								
@2	1 1	≠ 100	40								
C9 F	1 1	141 M	201								
491	1 F	utbl	A 0								
Ø1	1 1	#125 #101	40								
191	[C	410	Ail:								
100	80	el	ACC					DO			
EP 10	1 1	el[1]	20								
EP-12	1 4	400	AI								
(p 11	B 10	ut ualife	ASS					DO			
@ 14 @ 15	1 1	10 Per	25								
49 10	1 F	48	AG								
49 TT	1 1:	(4) (4)	A C								
Ø 19	l F	w125	80								
€9 30	ı F	4[1]	A 0								
49 21 10 72	B >-	ur(0)	ACC					DO			
D-22	1 F	elCEI	AU								
EP-34	1 1	400 90%	ASI								
66° 26	Ger	- (4)	A Etz					02			
II ≥ 27	Į ŀ	H01	40								
EP 79	1 [H(0)	40								
100 20	1 F	h60	At								
	1 1-	D04.	11								
(D-11		h(1)	Ac								
(2)-11 (2)-12 (3)-13		-00	At								
(2)-11 (2)-12 (2)-13 (2)-14	E	and a									
(2) 11 (2) 12 (2) 13 (2) 14 (2) 16 (2) 16		red el	Atok					E0			
139-11 139-12 139-10 139-16 139-16 139-17	900	od el al(2)	ATC					00			
(2) 11 (2) 12 (2) 13 (2) 14 (2) 15 (2) 16 (2) 17 (2) 17 (2) 18	() () () ()	ed el al(2) al(1)	A EC					10			
Description		ed el al(2) al(2) al(0)	MANUAL AND CONTRACTOR					10			

2 レポート課題8

2.1 8-1

alu_d を改造し,8 つの機能に拡張しました。, 具体的には既存の 4 つの機能に加えて大小比較、ビット毎の論理積、ビット毎の論理和, ビットシフトを追加しました。

```
SUBDESIGN alu_d
(
 clk, reset, ena : INPUT;
 a_in[7..0], b_in[7..0] : INPUT;
 ctrl[1..0] : INPUT;
 s_out[7..0] : OUTPUT;
 cflag, zflag : OUTPUT;
)
VARIABLE
cff, zff : DFFE;
BEGIN
cff.(clk, clrn, ena) = (clk, !reset, ena);
zff.(clk, clrn, ena) = (clk, !reset, ena);
cflag = cff.q;
zflag = zff.q;
zff.d = (s_out[] == 0);
CASE ctrl[] IS
WHEN 0 => (cff.d, s_out[]) = (B"0", a_in[]+1);
WHEN 1 => (cff.d, s_out[]) = (B"0", a_in[]-1);
WHEN 2 => (cff.d, s_out[]) = (B"0", a_in[])+(B"0", b_in[]);
WHEN 3 => (cff.d, s_out[]) = (B"0", a_in[])-(B"0", b_in[]);
WHEN 4 => IF a_in[] >= b_in[]
THEN s_out[] = a_in[];
ELSE s_out[] = b_in[];
END IF;
WHEN 5 => FOR i IN O TO 7 GENERATE
s_out[i] = a_in[i]&b_in[i];
```

```
END GENERATE;
WHEN 6 => FOR i IN 0 TO 7 GENERATE
s_out[i] = a_in[i]#b_in[i];
END GENERATE;
WHEN 7 => (s_out[]) = (a_in[6..0],B"0");
END CASE;
END;
```

2.2 8-2

以下にシュミレーションをした際の波形図を添付する。クロックの 2 周期を境に ctrl を 1 ずつ 増やしていくように入力を定めた。ctrl が 0 の時、 a_i n の値が 1 増えたものが s_i out に出力されている。これは期待していた正しい結果である。

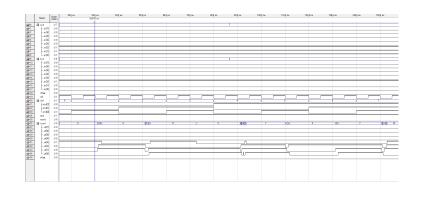
ctrlが1~3の時も期待していた結果が出力されている。

ctrl が 4 の時、期待される出力は a_in と b_in を大小比較して大きい方の値である a_in が出力されることである。波形図より期待通りの出力が得られているので正しいと言える。

ctrl が 5 の時、期待される出力は a.in と b.in の論理積が出力されることである。具体的には 00000111 と 00000100 の論理積なので 4 が出力されることが期待されている。波形図より期待通りの出力が得られているので正しいと言える。

ctrl が 6 の時、期待される出力は a.in と b.in の論理和が出力されることである。具体的には 00000111 と 00000100 の論理和なので 7 が出力されることが期待されている。波形図より期待通りの出力が得られているので正しいと言える。

ctrl が 7 の時、期待される出力は a_in の 1 ビットだけ右に論理シフトした値が出力されることである。具体的には 00000111 の右ビットシフトなので 14 が出力されることが期待されている。波形図より期待通りの出力が得られているので正しいと言える。



3 レポート課題 9

3.1 9-1

私は load,store,add 命令を利用して f(n) = f(n-1) + f(n-2) のフィボナッチ数列を計算するプログラムを組みました。プログラムの動きと正しいことを報告します。

fetcha でメモリから命令部分をとって来てそれを ira に格納するようにしました。fetchb でメモリからデータ部分をとって来てそれを irb に格納するようにしました。execa で ira に入っている命令を実行するようにしました。つまり今回はステージの更新と計算の実行は別のタイミングで行う必要があるのでステージの管理をしているフリップフロップはネガティブエッジトリガになるように、そのほかはポジティブエッジトリガになるように接続しました。波形を見るとそれが正しく動いていることがわかります。次にメモリの初期値について報告します。フィボナッチ数列の初項と2項目を120と128番地に格納しています。0番地と2番地の命令はload命令でレジスタに格納するようにしています。4番地の命令はadd命令で結果をレジスタに格納しています。6番地の命令はstore命令で先ほどの結果を格納したものをメモリの136番地に書き込むようにしています。8番地の命令はたほど格納して第2項目と計算した第3項目が格納されているレジスタを指定してadd命令を実行しています。10番地はその結果を144番地に書き込むようにしています。以降、同じ動きです。そして今回第6項までしか求められませんでした。原因は実行時間が足りなくこれ以上にメモリに命令を記述すると中途半端なところで実行が終わりハザードを起こす危険性があったのでこれ以上は求められませんでした。

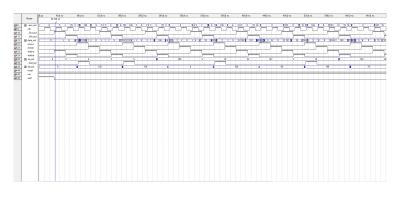


図1 波形図(途中まで)

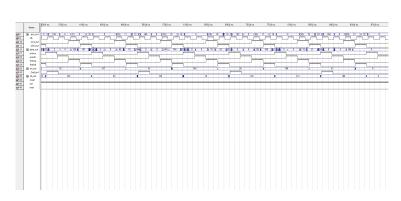


図2 波形図(前の続き)

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	8	120	9	128	146	4	18	136
8	147	40	19	144	148	76	20	152
16	149	112	21	160	0	0	0	0
24	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0
80	100	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0
120	1	0	0	0	0	0	0	0
128	1	0	0	0	0	0	0	0
136	2	0	0	0	0	0	0	0
144	3	0	0	0	0	0	0	0
152	5	0	0	0	0	0	0	0
160	8	0	0	0	0	0	0	0
168	0	0	0	0	0	0	0	0
176	0	0	0	0	0	0	0	0
184	0	0	0	0	0	0	0	0
192	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0
208	0	0	0	0	0	0	0	0
216	0	0	0	0	0	0	0	0
224	0	0	0	0	0	0	0	0
232	0	0	0	0	0	0	0	0
240	0	0	0	0	0	0	0	0
248	0	0	0	0	0	0	0	0

図3 メモリの結果(019番地が初期値、120番地より縦に実行結果)

3.2 9-2

先ほどのフィボナッチ数列をシフトを用いて拡張して f(n)=f(n-1)+2f(n-2) のフィボナッチ数列を計算するプログラムを組みました。先ほどとの変更点を報告します。先ほどのメモリに初期値と違い load してからそのまま add するのではなく、最初に load した方を shift 命令を使い二倍してから add するように初期値を書き換えました。以下に波形図とメモリの結果を添付します。

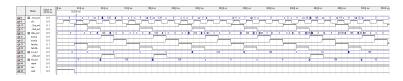


図4 波形図 (途中まで)

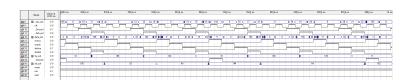


図5 波形図(前の続き)

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	8	120	9	128	184	0	146	4
8	18	136	185	32	144	40	16	144
16	186	64	145	64	17	152	0	0
24	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	0
40	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
56	0	0	0	0	0	0	0	0
64	0	0	0	0	0	0	0	0
72	0	0	0	0	0	0	0	0
80	100	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0
96	0	0	0	0	0	0	0	0
104	0	0	0	0	0	0	0	0
112	0	0	0	0	0	0	0	0
120	1	0	0	0	0	0	0	0
128	1	0	0	0	0	0	0	0
136	3	0	0	0	0	0	0	0
144	5	0	0	0	0	0	0	0
152	11	0	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0
168	0	0	0	0	0	0	0	0
176	0	0	0	0	0	0	0	0
184	0	0	0	0	0	0	0	0
192	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0
208	0	0	0	0	0	0	0	0
216	0	0	0	0	0	0	0	0
224	0	0	0	0	0	0	0	0
232	0	0	0	0	0	0	0	0
240	0	0	0	0	0	0	0	0
248	0	0	0	0	0	0	0	0

図 6 メモリの結果(0 21 番地が初期値、120 番地より縦に実行結果)