

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ Οργάνωση Υπολογιστών

Ομάδα: LAB31235453

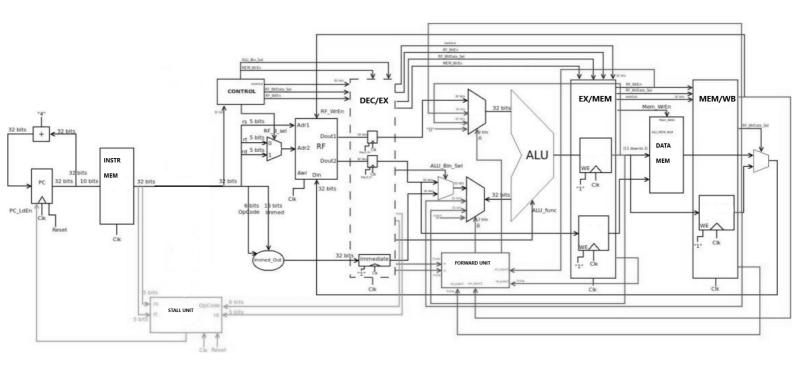
Μπαλαμπάνης Ηλίας 2014030127 Μποκαλίδης Αναστάσιος 2014030069

Αναφορά Εργαστηρίου 5

Περιγραφή Άσκησης

Σε αυτό το εργαστήριο μας ζητήθηκε να μετατρέψουμε τον επεξεργαστή μας σε έναν pipelined processor. Στην ουσία προσθέσαμε 3 επιπλέον καταχωρητές οι οποίοι υλοποιούν το pipeline. Τέλος σημαντικό να σημειωθεί πως πλεον το Top Module του Xilinx project είναι το Datapath καθώς το Control έγινε component του.

Παρακάτω παραθέτουμε το datapath σε block diagram, το οποίο βρίσκεται και στο .zip αρχείο.



Control Unit

Αλλάξαμε το Control Unit, έτσι ώστε να μην είναι μια FSM. Πλέον το Control υλοποιείται μέσα στο Datapath και τα σήματα, που είναι απαραίτητα για την υλοποίηση της εντολής, δίνονται όλα μαζί από αυτό με το γνωστό τρόπο του instruction fetch. Το RF_B_Sel δίνεται κατευθείαν στο Dec Stage και τα υπόλοιπα σήματα που βγαίνουν από το Control πάνε στον DEC/EX καταχωρητή. Από τον DEC/EX βγαίνουν τα σήματα ALU_Bin_Sel και ALU_func, τα οποία ελέγχουν την λειτουργία της ALU, και τα υπόλοιπα σήματα πάνε στον EX/MEM reg. Έπειτα από τον EX/MEM βγαίνει το σήμα MEM_WrEn, το οποίο πάει στην Data Memory, και τα υπόλοιπα σήματα πάνε στον MEM/WB. Τέλος, από το MEM/WB βγαίνουν το σήμα RF_WrData_Sel για την επιλογή του τι θα καταχωρήσει στην RF, το σήμα RF_WrEn και το rd, που πάει στο Dec Stage.

Αποσφαλμάτωση των Data Hazards

Forward

Δημιουργήσαμε το Forward Unit για να αντιμετωπίσουμε τα data hazards, που μπορεί να δημιουργηθούν κατά την εκτέλεση των εντολών. Το Forward υλοποιείται και σε one-cycle αλλά και σε two-cycle. Για έναν κύκλου ελέγχουμε από τον καταχωρητή ΜΕΜ/WB το rd με τα rs και rt από τον καταχωρτή DEC/EX. Αν ένα από τα δύο ή και τα δύο είναι ίσα με το rd και το RF_WrEn από τον ΜΕΜ/WB είναι ενεργό σε αυτή την περίπτωση δίνουμε τα κατάλληλα σήματα στους πολυπλέκτες στις εισόδους της ALU, δηλαδή κάνουμε one-cycle forward. Για να κάνουμε Forward για δύο κύκλους κάνουμε την ίδια ακριβώς διαδικασία με τον ένα κύκλο με τη διαφορά ότι αντί να παίρνουμε τα σήματα από τον ΜΕΜ/WB τα παίρνουμε από τον ΕΧ/ΜΕΜ. Οι εξόδοι του κουτιού αυτού είναι οι επιλογείς των πολυπλεκτών πριν την ALU.

Stall

Επίσης προσθέσαμε στο datapath μας το Stall Unit για την περαιτέρω αντιμετώπιση hazards. Σε αυτό το κομμάτι ελέγχουμε την τρέχουσα εντολή με την προηγούμενη. Αν η προηγούμενη είναι εντολή η lw και χρησιμοποιεί τον καταχωρητή, που φορτώνει, στην τρέχουσα (add) δημιουργούμε μία καθυστέρηση δύο κύκλων. Αυτό το κάνουμε για να μπορέσει να πάρει την πληροφορία από τη μνήμη και να τη χρησιμοποιήσει στην add. Για να σιγουρευτόυμε ότι δεν θα γίνει παράλληλα με το stall και το forward, γίνεται ο κατάλληλος έλεγχος στο forward unit. Η έξοδος αυτού του κουτιού είναι το PC_LdEn. Επίσης εδώ ελέγχεται και το εξωτερικό σήμα reset, μέσω της μικρής αυτής FSM.

Κυμματομορφές

Παρακάτω βλέπουμε τις εικόνες του simulation για να επιβεβαιώσουμε τη σωστή λειτουργία του pipeline processor.

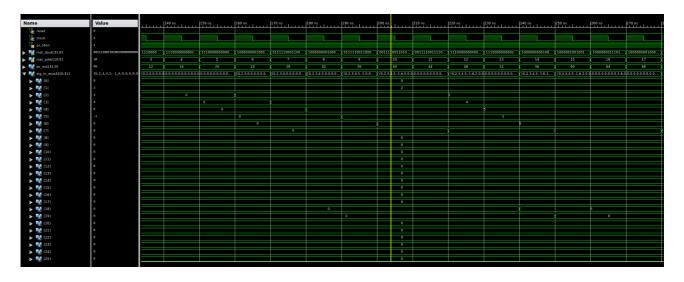
Στη συνέχεια, βλέπουμε το simulation στην αντιμετώπιση των data hazards

• Εκτέλεση χωρίς data hazards

Βλέπουμε δηλαδή πως στον καταχωρητή \$1, καταχωρείται το 2, στον \$2 το 3 κ.ο.κ , σύμφωνα με τις εντολές που δίνονται στο αρχείο ram.data. Παρατηρούμε το sw, οπου MEM[1026]= 5, ομοίως και για την lw παρακάτω.



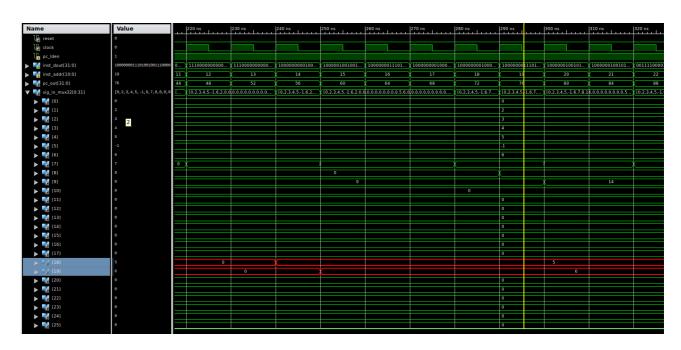
Οι καταχωρητές μετά την εκτέλεση των εντολών



Ta SW

Name	Value	170 ns	180 ns	190 ns	200 ns	210 ns
le reset	0					
lock	1					
l☐ pc_lden	1					
▶ 📑 inst_dout[31:0]	11100000000000010000000000	0111110001100	1000000001000	0111110011000	0011110011010	0011110011110
inst_addr[10:0]	12	7	8	9	10	11
pc_out[31:0]	48	28	32	36	40	44
sig_in_mux32[0:31]	[0,2,3,4,5,-1,6,2,0,0,0,0	[0,2,3,4,0,0,0,0,0,0	[0,2,3,4,5,0,0,0,0	[0,2,3,4,5,-1,0,0,	[0,2,3,4,5,-1,6,0,0,0	,0,0,0,0,0,0,0,0,0,0
▶ 📑 alu[31:0]	16	6	8	2	16	8
▶ ■ rf_b[31:0]	0	3	5	-1	6	k
▶ 📑 rd[4:0]	19	6	4	7	6	18
▶ ■ ram[2047:0]	[00000000000000000000000000000000000000	000000000000000000000000000000000000000	,0000000000000000	00000000000000000	0,0000000000000000	00000000000000000
▶ ■ data_addr[10:0]	1026	1023	1025	1026	1024	1028
▶ 📑 data_din[31:0]	0	0	3	5	-1	6
🗓 data_we	0					
▶ 📑 rd[4:0]	1	4	7	6	18	19
▶ ■ rf_a[31:0]	0	4	3	*	5	2
▶ ■ rf_b[31:0]	2	5	4294967295	6	k	0
mem_data[31:0]	0			0		
mem_data_out[31:0]	0				0	
▶ 📑 immed[31:0]	2	¥ 4		10	2	14
dec_ex_opcode[5:0]	15	32	31	32	31	*
dec_ex_rd[4:0]	19	6	4	7	6	18
imem_out_rs[4:0]	0	3	2	*	5	7
imem_out_rt[4:0]	0	0	5	*		0
🏗 pc_lden	1					
🗓 imem_en	1					

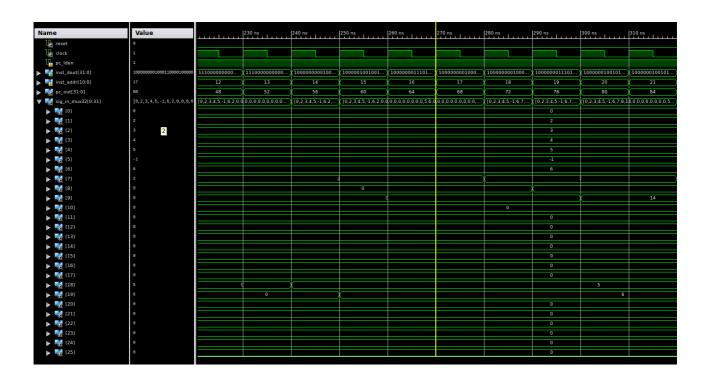
Ta LW



Forward

Βλέπουμε πως στον καταχωρητή \$1 καταχωρείται το 2, στον \$2 το 3, έπειτα γίνονται σωστά οι εντολές μέχρι να γίνει η καταχώρηση στον \$9 της πρόσθεσης του \$7 με τον \$7, δηλαδή 14. Στις επόμενες εκτελέσεις της add χρησιμοποιούνται καταχωρητές από δύο κύκλους πριν (forwarding within two cycles). Ενώ στις επόμενες εκτελέσεις της add μέχρι την καταχώρηση στον \$10 το 48 χρησιμοποιούνται καταχωρητές από ένα κύκλο πριν (forwarding within one cycle).

Two cycle regs



Two cycle run-time

Name	Value		230 ns	240 ns	1250 ns	260 ns	270 ns	1280 ns	1290 ns	1300 ns
V _o reset	0									
U clock	1									
1 pc_lden	1									
inst_dout[31:0]	1000000001000110000100000	111000000000	11100000000000	1000000000100	1000001001001	1000000011101	1000000001000	1000000001000	1000000011101	100000010010
inst_addr(10:0)	17	12	13	14	15	16	17	18	19	20
pc_out[31:0]	68	48	52	56	60	64	68	72	76	80
• 🌃 sig_in_mux32[0:31]	[0,2,3,4,5,-1,6,2,0,0,0,0	[0,2,3,4,5,-1,6,2,0,0	.0,0,0,0,0,0,0,0,0,0	[0,2,3,4,5,-1,6,2,	[0,2,3,4,5,-1,6,2,0,0	.0,0,0,0,0,0,0,0,5,6,0	.0,0,0,0,0,0,0,0,0,	[0,2,3,4,5,-1,6,7,	[0,2,3,4,5,-1,6,7,	[0,2,3,4,5,-1,6,
alu[31:0]	14	16	2	3	7	8	14	6	9	18
rf_b[31:0]	2	0	2	3	5	3	2	3	6	7
- 🕌 rd[4:0]	9	19	1	2	7	8	9	6	7	X
• 🎼 ram[2047:0]	[00000000000000000000000000000000000000	[000000000000000]	000000000000000000000000000000000000000	0000000000000000	00000000000000000	000000000000000000000000000000000000000	0,0000000000000000000000000000000000000	00000000000000000	0,000000000000000	0000000000000
• 📑 data_addr[10:0]	1026	1026	1028	10	24	1025	1026	1027	1025	1026
- 📑 data_din[31:0]	3	-	9	2	3	5	3	2	3	6
🖫 data_we	0									
rd[4:0]	6	1	2	7	8	9	6	7	k	9
→ 📑 rf_a[31:0]	3	-	d .	2	5	2			7	X
· 📑 rf_b[31:0]	3	2	3	5	3	2	3	6	7	6
 mem_data[31:0] 	0	0	5	6	X	0		5	K	0
 mem_data_out[31:0] 	0	-	d .	5	6	K	0		5	X
• 📑 immed[31:0]	3	2	X				3			
dec_ex_opcode[5:0]	32	15		6	X			3	2	
• 📑 dec_ex_rd[4:0]	9	19	1	2	7	8	9	6	7	X
imem_out_rs[4:0]	2	-		1	18	7			7	X
imem_out_rt[4:0]	2	-		18	2	7	2	6	7	6
l pc_lden	1									
la imem_en	1									

One cycle regs



One cycle run-time

Name	Value	270 ns	280 ns	290 ns		300 ns	310 ns	320 ns
∏ _o reset	0							
To clock	1							
୍ଲି pc_iden	1							
▶ ■ inst_dout[31:0]	1000000011101001001110000	1000000001000	1000000001000	100000	0011101	1000000100101	1000000100101	0011110000110
▶ ■ inst_addr[10:0]	19	17	18		19	20	21	22
pc_out[31:0]	76	68	72		76	80	84	88
sig_in_mux32[0:31]	[0,2,3,4,5,-1,6,7,8,0,0,0	[0,2,3,4,5,-1,6,2,0	[0,2,3,4,5,-1,6,7,	[0,2,3,4	,5,-1,6,7,	[0,2,3,4,5,-1,6,7,8,1	4,0,0,0,0,0,0,0,0,5	[0,2,3,4,5,-1,6,9,
▶ 📑 alu[31:0]	9	14	6	$\!$	9	18	24	48
▶ ■ rf_b[31:0]	6	2	3	X	6	7	6	14
▶ □ rd[4:0]	7	9	6	X	7			10
Tam[2047:0]	[00000000000000000000000000000000000000	[000000000000000000]	0,00000000000000000	0000000	0000000000	000000000000000000000000000000000000000	00000000000000000	000000000000000000
data_addr[10:0]	1025	1026	1027	X	1025	1026	1028	1030
data_din[31:0]	3	3	2	x	3	6	7	6
🖫 data_we	0							
▶ 📑 rd[4:0]	9	6	7	$\!$	9		10	18
▶ ■ rf_a[31:0]	7)X	3	$\!$	7	1	4	2
▶ ■ rf_b[31:0]	7	3	6	X	7	6	14	5
mem_data[31:0]	0	0	5	$\!=$	(5	6
mem_data_out[31:0]	5	0		X	5	k	i e	5
immed[31:0]	3				3			*
dec_ex_opcode[5:0]	32				32			
dec_ex_rd[4:0]	7	9	6	X	7			10
imem_out_rs[4:0]	7	X :	2	X	7		9	1
imem_out_rt[4:0]	7	2	6	X	7	6	9	0
l pc_iden	1							
lo imem_en	1							

• Stall

Στο Simulation αυτό βλέπουμε διαδοχικά την εκτέλεση των εντολών lw και add με stalling. Στον \$18(MEM[1026]) καταχωρείται το 5, στον \$6 το 10, στον \$19 (MEM[1028]) το 6, στον \$6 το 8, στον \$20(MEM[1026]) το 5 και στον \$6 το 7. Στο stall γίνεται καθυστέρηση εντολής για να μπορεί να εκτελεστεί σωστά η επόμενη.

Name	Value	320 ns	330 ns	340 ns	350 ns	360 ns	370 ns	380 ns	390 ns	400 ns	410 ns	420 ns	430 ns	440 ns	450 ns	460 ns
∐a reset	0			1												
10 clock																
1 pc_lden																
inst_dout[31:0]	0011110010010011000000000	0011110000110	1000001001000	V	00111100100100110000	0000000001011	1000000000100	0011110	000110100000000000	000000110	1000001010000		0000000000	000000000000000000000000000000000000000	00000	_
inst_addr[10:0]	23	22	₹	23		24	Ţ	25		26	↓	27		28	29	30
pc_out[31:0]	92	88	Ţ ———	95		96	Ţ	100		104	↓	108		112	116	120
sig_in_mux32[0:31]	[0,2,3,4,5,-1,6,9,8,24,0,	[0,2,3,4,5,-1,6,9,	[0,2,3,4,5,-1,6,9,	[0,2,3,4,5	-1,6,9, (0,2,3,4,5,-1,6	9,8,24,48,0,0,0,0,0,0,0,0,	. [0,2,3,4,5,-	1,10,9,8,24,48,0,0,0,	0,0,0,0,5,6,0,0,0,0,0,0	0,0,0,0,0,0)	[0,2,3,4,5,-1,8,9,	[0,2,3,4,5,-1,8,9,8,	24,48,0,0,0,0,0,0,0,0,5,	5,5,0,0,0,0,0,0,0,0,0,	[0,2,3,4,5,-1,7,9,8	24,48,0,0,0
alu(31:0)	10	48	8	10	—;——	16	1			8	1	7	\	0		
of_b(31:0)	5	14	} ———	\$	—;——			1	 	0		2	₹	0		
rd[4:0]	6	10	18	¥ 6	─ ∤	19		6	}	20		6	¥	0		
ram[2047:0]	[00000000000000000000000000000000000000	[00000000000000000000000000000000000000	000000000000000000	0000000000	000000000000000000000000000000000000000	00,000000000000000000000000000000000000	0,0000000000000000000000000000000000000	000000000000000000000000000000000000000	000,00000000000,000	00000000000000000	0000000000000,0000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000000,00000	000000000
data_addr[10:0]	1026	1030	1036	V	1026		1028		 	,	1026		1025		1024	
data_din[31:0]	5	₹	14	$\overline{}$,				1	}			2		0	
l data_we		1		1						1			1			
rd[4:0]	19	18	6	X	19		¥ 6	\	20		6			0		
rf_a(31:0)	5	2	₹	=	,		}	1	ļ		5			0		
rf_b(31:0)	6	3	}	X	6		¥ 6	\	0		2	}		0		
mem_data[31:0]			₹		_	-	}	6		}		1			0	
mem_data_out[31:0]		3	6	X	-	_	1	\	6	1	\		,			
immed[31:0]	11	1	1	$\overline{}$		11		}				1		0		
dec_ex_opcode(5:0)	32	32	15	32	_	15		32	1	15		32	¥	0		
dec_ex_rd[4:0]	6	10	18	¥ 6	_	19		6	}	20			Ţ	0		
imem_out_rs[4:0]	4	1	18	$\overline{}$	4		¥				20	1	1	0		
imem_out_rt[4:0]	0	· ·	18	X	0		19	* 	0		1	<u> </u>		0		
l pc_iden	0															
l imem_en	0															

Οι καταχωρητές μετά το Stall

Name	Value	470 ns	480 ns	490 ns	500 ns	510 ns	520 ns		530 ns	540 ns	550 ns	560 ns
Te reset	0											
1 clock	1											
l₁ pc_lden	1											
▶ ■ inst_dout[31:0]	000000000000000000000000000000000000000								0000000000	000000000000000000000000000000000000000	0000	
▶ ■ inst_addr[10:0]	36	31	32	33	34	35	\times	6	37	38	39	40
▶ ■ pc_out[31:0]	144	124	128	132	136	140	X 1	14	148	152	156	160
▼ 🔣 sig_in_mux32[0:31]	[0,2,3,4,5,-1,7,9,8,24,48							[0,2,	3,4,5,-1,7,9,8,24,48	,0,0,0,0,0,0,5,6,5,0,0	0,0,0,0,0,0,0,0,0	
▶ ■ [0]	0									0		
[1]	2									2		
▶ ■ [2]	3									3		
■ [3]	4									4		
[4]	5									5		
[5]	-1									-1		
▶ ■ [6]	7									7		
[7]	9									9		
[8]	8									8		
▶ ■ [9]	24									24		
▶ ■ [10]	48									48		
▶ ■ [11]	0									0		
▶ ■ [12]	0									0		
▶ ■ [13]	0									0		
▶ ■ [14]	0									0		
▶ ■ [15]	0									0		
▶ ■ [16]	0									0		
▶ ■ [17]	0									0		
► 18 [18]	5									5		
▶ ■ [19]	6									6		
▶ ■ [20]	5									5		
▶ ■ [21]	0									0		
[22]	0									0		
▶ 1 [23]	0									0		
▶ 1 [24]	0									0		
[25]	0									0		