Homework 6 solutions

The Table

Opcode	Xi	Yi	Zi	C0	Fi	Ci+1	N	C	Z	V
ADD	Ai	Bi	Ci	0	Si	Zi+1	F7	C8	nor (F0..F7)	C8xorX7
INC	Ai	0	Ci	1	Si	Zi+1	F7	C8	nor (F0..F7)	C8xorX7
DEC	Ai	1	Ci	0	Si	Zi+1	F7	C8	nor (F0..F7)	C8xorX7
SUB	Ai	~Bi	Ci	1	Si	Zi+1	F7	C8	nor (F0..F7)	C8xorX7
CMP	Ai	~Bi	Ci	1	Si	Zi+1	F7	C8	nor (F0..F7)	C8xorX7
PASS	Ai	0	0	x	Si	x	F7	C8/0	nor (F0..F7)	C8xorX7/0
NEG	~Ai	0	Ci	1	Si	Zi+1	F7	x	nor (F0..F7)	C8xorX7
XOR	Ai	Bi	0	x	Si	x	F7	0	nor (F0..F7)	0
XNOR	Ai	~Bi	0	x	Si	x	F7	0	nor (F0..F7)	0
NOT	~Ai	0	0	x	Si	x	F7	0	nor (F0..F7)	0
AND	Ai	Bi	0	x	Zi+1	x	F7	0	nor (F0..F7)	0
OR	Ai	Bi	1	x	Zi+1	x	F7	0	nor (F0..F7)	0
SHL	x	x	x	x	Ai-1	x	F7	A7	nor (F0..F7)	0
SHR	x	x	x	x	Ai+1	x	F7	0	nor (F0..F7)	0

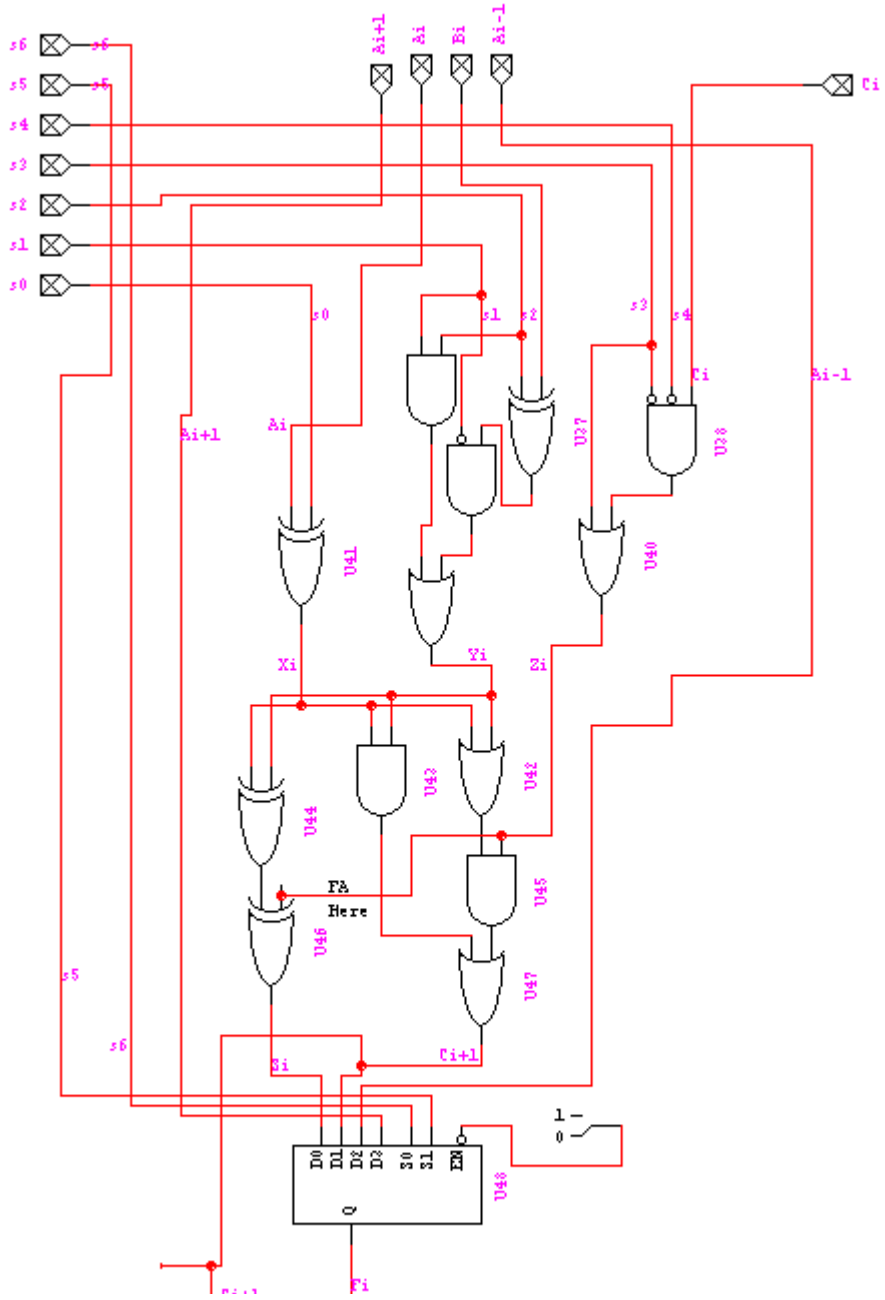
Design Notes

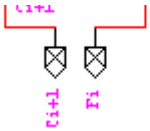
- Implement AND by setting Ci to 0 and selecting the result from Zi+1 (Carry out of the full adder). Zi+1 = XiYi+ZiYi+ZiXi. If we set Xi to 0, then Zi+1 = XiYi
- Implement AND by setting Xi to 1 and selecting the result from Zi+1. Zi+1 = XiYi+ZiYi+ZiXi. If we set Xi to 1, then Zi+1 = XiYi + (Xi+Yi) = Xi+Yi.
- Note that PASS looks more like a logical operation than like an arithmetic operation. If we think of it this way, the control lines get a little easier to optimize. C and V for PASS will always be zero either way, so no need to worry about control lines for those cases
- I decided to implement SHL and SHR by bypassing the FA completely, using a 4:1 multiplexor on the output. This way, most of the control lines will be don't cares for these two instructions.
- Whenever Zi is set to 1 or 0, Ci+1, Zi, and C0 are don't cares.

Control Line Definitions:

- s0 controls Xi: Xi = s0 xor Ai (conditionally invert Ai)
- s1s2 control Yi: Yi = s1's2'Bi + s1's2Bi' + s1s2'(0) + s1s2(1) (4:1 mux on s1,s2). This simplifies to s1'[s2 xor Bi] + s1s2
- s3s4 control Zi: Zi = s3's4'Ci + s3's4(0) + s3s4'(1) + s3s4(1) (4:1 mux on Ci, 0 1). This simplifies to s3's4'Ci + s3
- s5s6 control Fi: Fi = 4:1 mux on Si, Zi+1, Ai-1, Ai+1 selected by s5s6.
- s7: Disables V and C in the case of logic operations: V = s7(C8xorC7), C* = s7(C8)
- s8: Enables C in the case of SHL: C = C* + s8A7
- s9: Determines value of C0, C0 = s9

Gate Level Implementation of an ALU BitSlice: Total Gates = 13 + (4:1mux) + Inverter-for-Bi = 18gates





Remainder of System:

Condition Codes

- V: 2 gates (C8xorC7)s7
- C: 3 gates (C8s7)+(A7s8)
- Z: 1 gate (8-input NOR)
- N: 0 gates (F7)

Control Logic: By inspecting the table above. The following control lines are asserted for the following instructions

- s0 = [NEG] + [NOT]
- s1 = [INC] + [DEC] + [PASS] + [NEG] + [NOT]
- s2 = [DEC] + [SUB] + [CMP] + [XNOR]
- s3 = [OR]
- s4 = [ARITHMETIC]'
- s5 = [SH(L/R)]
- s6 = [AND] + [OR] + [SHR]
- s7 = [ARITHMETIC]
- s8 = [SHL]
- s9 = ([ADD] + [DEC])'

Optimizing the Control Logic: Determine encoding by placing the the instructions in a K-MAP while trying to keep the groups together according to the above. For example, NEG and NOT are close together to make s0 simple, ADD and DEC are adjacent to make s9 simple, s1 and s2 are grouped as good as can be without violating the separation between logic and arithmetic functions, etc. This is probably not an optimal placement, but its not bad.

P3P2P1P0	00	01	11	10
00	OR	PASS	INC	x
01	AND	NOT	NEG	x
11	SHR	XNOR	DEC	ADD
10	SHL	XOR	CMP	SUB

Letting ARITHMETIC = P3, we organize the k-map so that all arithmetic functions are in the P3=1 region. According to the K-MAP we get the following logic functions. All but s1 and s2 can be implement with one gate or less.

- s0 = P2P1'P0
- s1 = P2P1' + P3P2P0

- s2 = P2P1P0 + P3P2P0’
- s3 = P2’P1’P0’
- s4 = P3’
- s5 = P3P2P1
- s6 = P3'P2's8'
- s7 = P3
- s8 = P3'P2'P1P0'
- s9 = (P3P1P0)'

Decoder Gate Count = 12 + (4 inversions) = 16

Total System = (BitSlice*8) + (CC) + (Decoder) + (2 control line inversions) = 144 + 6 + 16 + 2 = 168 gates

The critical delay is as follows:

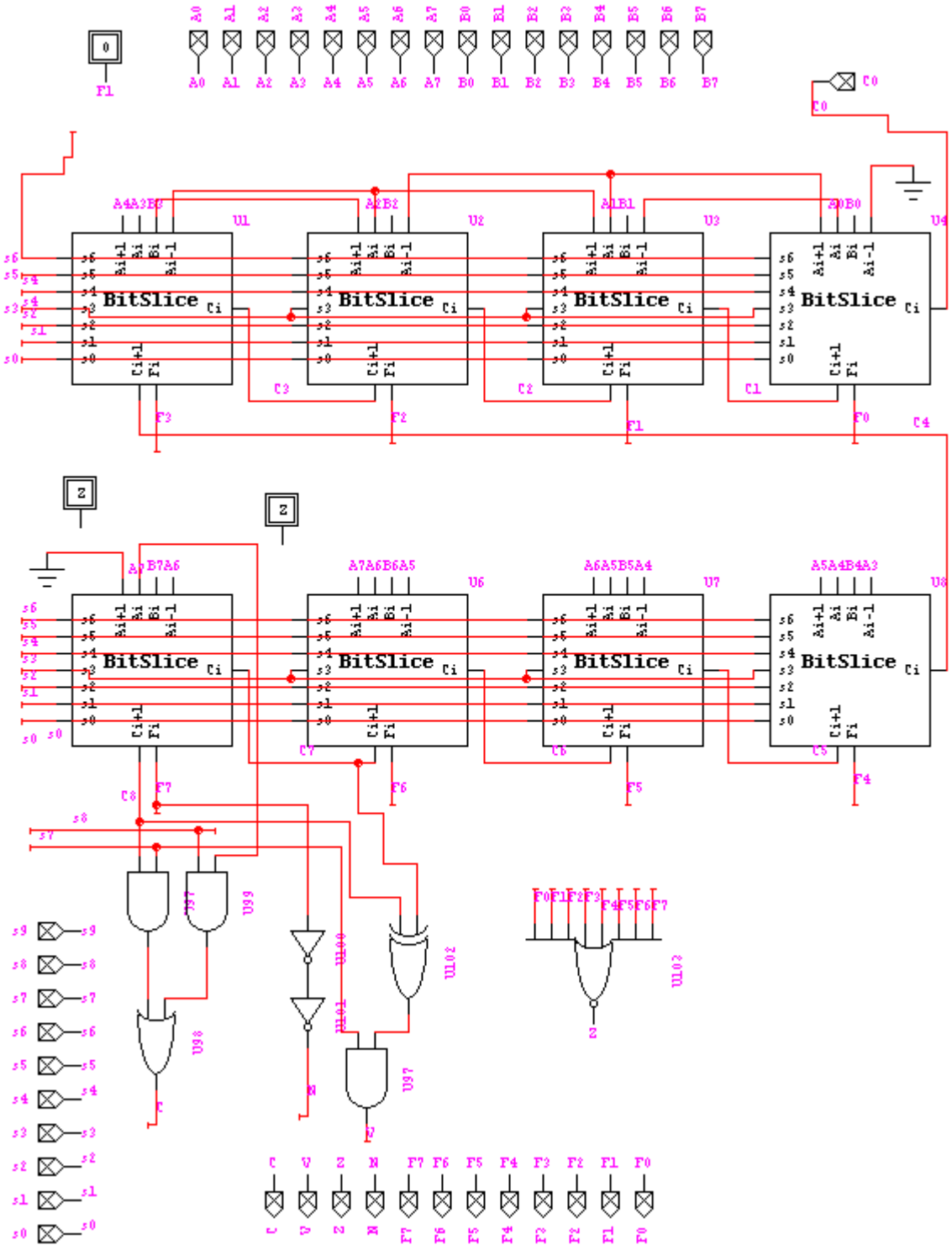
- For bit 0, the critical path is from P3 to s2 to Yi to Ci+1: 11 gates
- For bits 1-6, the critical path is from Ci to Ci+1: 6 gates
- For bit 7, the critical path is from C7 to (C or F7): 6 gates

The total delay = 11 + (6*6) + 6 = 53 gate delays

Here is the Verilog Model for the Controller:

```
module Decoder(P3, P2, P1, P0, s0, s1, s2, s3, s4, s5, s6, s7, s8, s9);
    input P3;
    input P2;
    input P1;
    input P0;
    output s0;
    output s1;
    output s2;
    output s3;
    output s4;
    output s5;
    output s6;
    output s7;
    output s8;
    output s9;
    assign s7 =  P3;
    assign s8 = ~P3 & ~P2 & P1 & ~P0;
    assign s3 = ~P2 & ~P1 & ~P0;
    assign s9 = ~(P3 & P1 & P0);
    assign s4 = ~P3;
    assign s5 = ~P3 & ~P2 & P1;
    assign s6 = ~P3 & ~P2 & ~s8;
    assign s0 =  P2 & ~P1 & P0;
    assign s1 =  (P2 & ~P1) | (P3 & P2 & P0);
    assign s2 =  (P2 & P1 & P0) | (P3 & P1 & ~P0);
endmodule
```

ALU Schematic



Test Vectors

Ready, Vectors Processed (17), No Errors

	0	1	2	3	4	5	6
0	$TIME	$I [P3..P0	$E Z	$E V	$E N	$E C	$E [F7..F0
1	$D 50	0 "OR"	0	0	1	0	F5
2	$D 50	1 "AND"	0	0	0	0	05
3	$D 50	2 "SHL"	0	0	0	1	4A
4	$D 50	3 "SHR"	0	0	0	0	52
5	$D 50	4 "PASS"	0	0	1	0	A5
6	$D 50	5 "NOT"	0	0	0	0	5A
7	$D 50	6 "XOR"	0	0	1	0	F0
8	$D 50	7 "XNOR"	0	0	0	0	0F
9	$D 50	A "SUB"	0	1	0	1	50
10	$D 50	B "ADD"	0	0	1	0	FA
11	$D 50	C "INC"	0	0	1	0	A6
12	$D 50	D "NEG"	0	0	0	0	5B
13	$D 50	E "CMP"	0	1	0	1	50
14	$D 50	F "DEC"	0	0	1	1	A4
15	$STOP						
16	1451						

Top Level Schematic