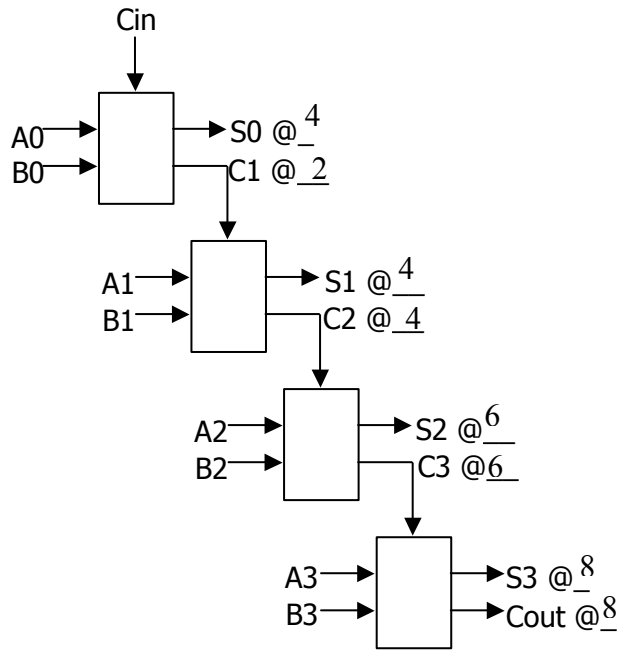
Name: Email: ID#

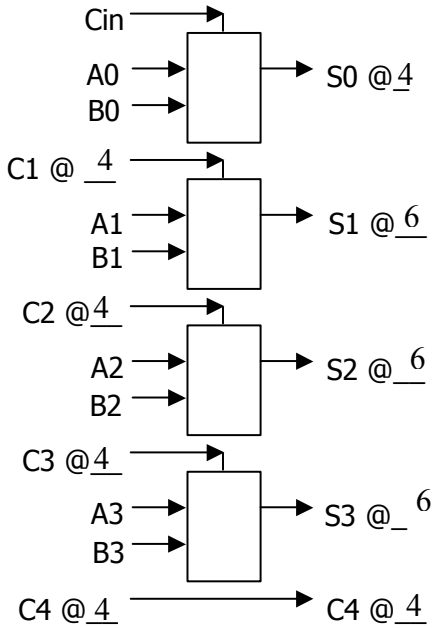
UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering and Computer Sciences
EECS150 Fall 2001

Prof. Subramanian

Midterm III

1) Recalculate the various propogation delays in a 4-bit carry lookahead adder and a ripple adder assuming that XOR gates have twice the delay of all other gates.

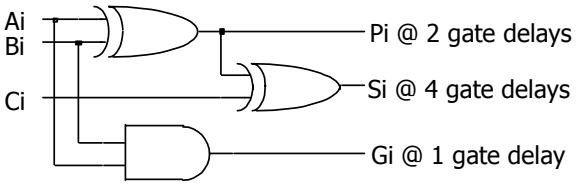




For the ripple adder, we have:

S0 = A xor B xor Cin = 4 gate delays
C1 = BCi + ACi + AB = 2 gate delays
Si = 2 past Ci-1
Ci = 2 past Ci-1

For the carry lookahead adder, we have:
For the basic 1-bit circuit, we have:



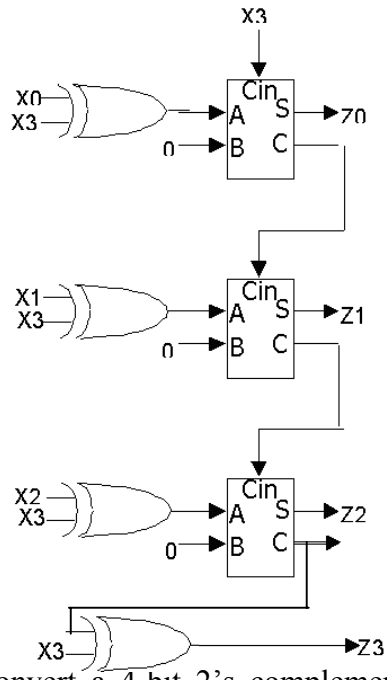
Therfore, C1-C4 @ 2 past Pi (since Ci is two level AND/OR logic) = 4 gate delays

S0 @ 4 gate delays
S1-S3 @ 2 past Ci = 6 gate delays

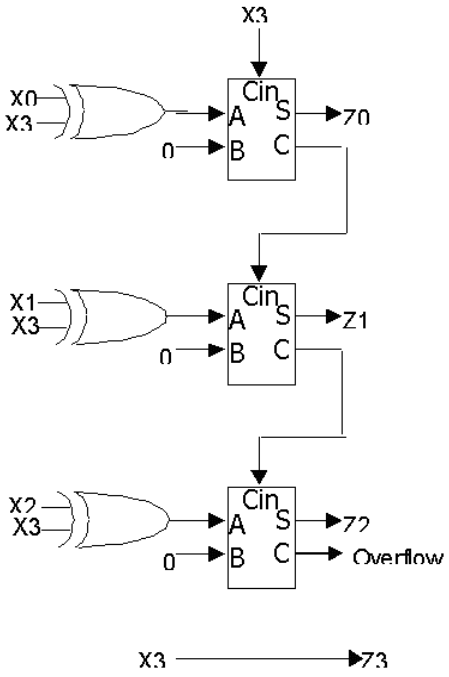
Name: Email: ID#

2) The easiest way to perform arithmetic operations using the sign-and-magnitude system is to convert to 2's complement for your calculations. You are given the following components:
   - A 4-bit universal shift register
   - As many 1-bit full adders as you need
   - As many 2-input XOR gates as you need
   - As many 2-input NAND gates as you need

   a) Design a circuit to convert a 4-bit sign-and-magnitude number to 2's complement number.

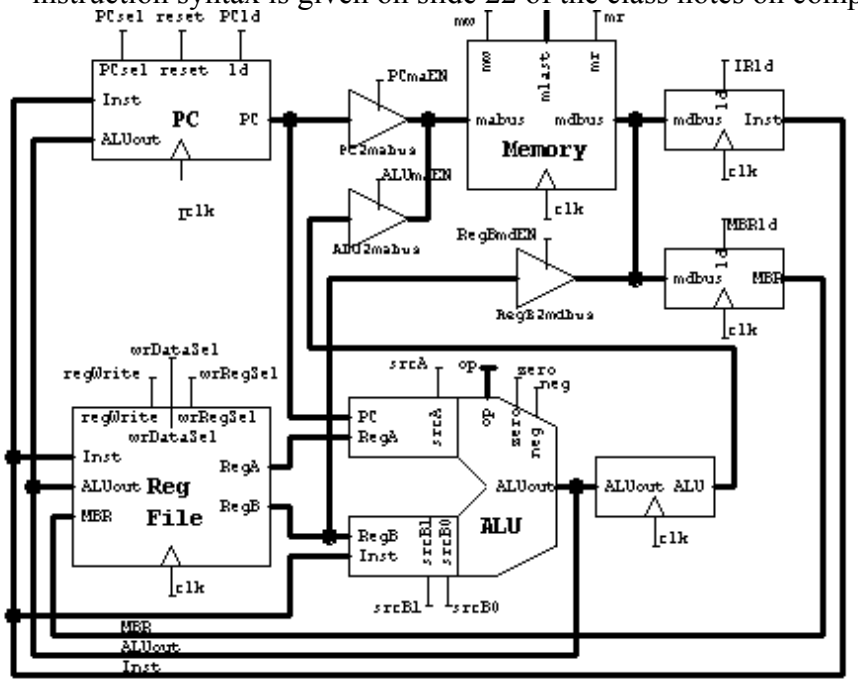


   b) Design a circuit to convert a 4-bit 2's complement number to a sign-and-magnitude number.



Name: Email: ID#

3) Trace the execution of the current instruction in the following microprocessor datapath. The instruction syntax is given on slide 22 of the class notes on computer organization.



PC: 38_H
REG[000]:0000_H
REG[001]:$00F0_H$
REG[010]:0040_H
REG[011]:0100_H
REG[100]:0000_H
REG[101]:1100_H
REG[110]:$FF00_H$
REG[111]:$FF00_H$
MEM[38_H]:2502_H
MEM[$F0_H$]:1264_H
MEM[$F1_H$]:0000_H
MEM[$F2_H$]:1000_H
MEM[$F3_H$]:$FFFF_H$
MEM[$F4_H$]:0000_H
… …
MEM[FF_H]:0000_H

a) List the sequence of transfer operations and control signals (see slide 28 for an example of what is expected from you).

Instruction fetch:

mabus ¬ PC;	– move PC to memory address bus (PCmaEN, ALUmaEN)
memory read;	– assert memory read signal (mr, RegBmdEN)
IR ¬ memory;	– load IR from memory data bus (IRld)
op ¬ add	– send PC into A input, 1 into B input (srcA, srcB0, scrB1, op)
PC ¬ ALUout	– load result of incrementing in ALU into PC (PCld, PCsel)

Instruction decode: (IR= 0010010100000010)

IR to controller (instuction is a LD instruction, rt is REG[010], rs is REG[001], offset = 0000010
values of A and B read from register file (rs, rt)

Instruction execution:

op ¬ add	– send regA into A input, Inst into B input, add (srcA, srcB0, scrB1, op)
ALU ¬ ALUout	– clock ALU output into ALU Buffer
mabus ¬ ALU	– move ALU to memory address bus (PCmaEN, ALUmaEN)
memory read;	– assert memory read signal (mr, RegBmdEN)
MBR ¬ memory;	– load MBR from memory data bus (MBRld)
rt ¬ MBR	– store MBR into destination register (regWrite, wrDataSel, wrRegSel)

b) After the instruction execution is complete, list the contents of the following, in hexadecimal:

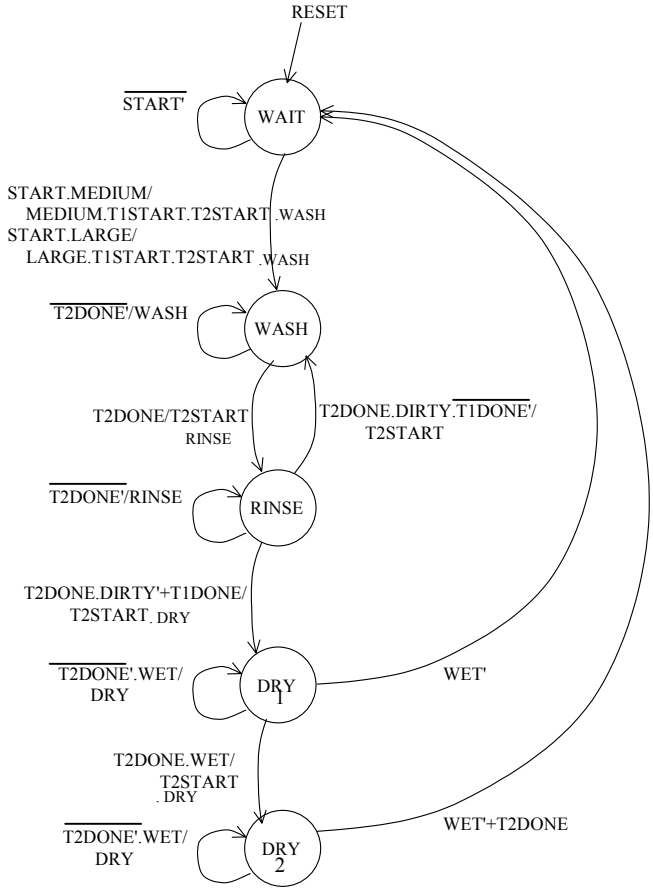
PC: 39_H	REG[010]: 1000_H	REG[101]: 1100_H
REG[000]: 0000_H	REG[011]: 0100_H	REG[110]: $FF00_H$
REG[001]: $00F0_H$	REG[100]: 0000_H	REG[111]: $FF00_H$

c) Suppose you wanted to display the results of the execution of your instruction on the display (which is connected to MEM[FF_H]. Write the contents of MEM[39_H].

The instruction is sw, REG[001], REG[010], 0000010, i.e., 0100010100001111
MEM[39_H] = $450F_H$

Name: Email: ID#

4) You are going to implement the following machine (which you may remember as the washing machine from midterm I) using a pure jump counter (Note that there are MEDIUM and LARGE inputs and outputs).



a) Determine an optimal encoding for the states.

WAIT: 000 RINSE: 010 DRY2: 100

WASH: 001 DRY1: 011

b) Determine the control functions:

CNT = WAIT•START + WASH•T2DONE + RINSE•(T2DONE•DIRTY’ + T1DONE) + DRY1•T2DONE•WET

LD = RINSE•T2DONE•DIRTY•T1DONE’

CLR = DRY1•WET’ + DRY2•(WET’ + T2DONE)

c) List the contents of the Jump ROM.

Addr.	Contents	Addr.	Contents	Addr.	Contents	Addr.	Contents
000	XXX	010	001	100	XXX	110	XXX
001	XXX	011	XXX	101	XXX	111	XXX

Name: Email: ID#

d) List the output functions:

LARGE = WAIT•START•LARGE

MEDIUM = WAIT•START•MEDIUM

T1START = WAIT•START

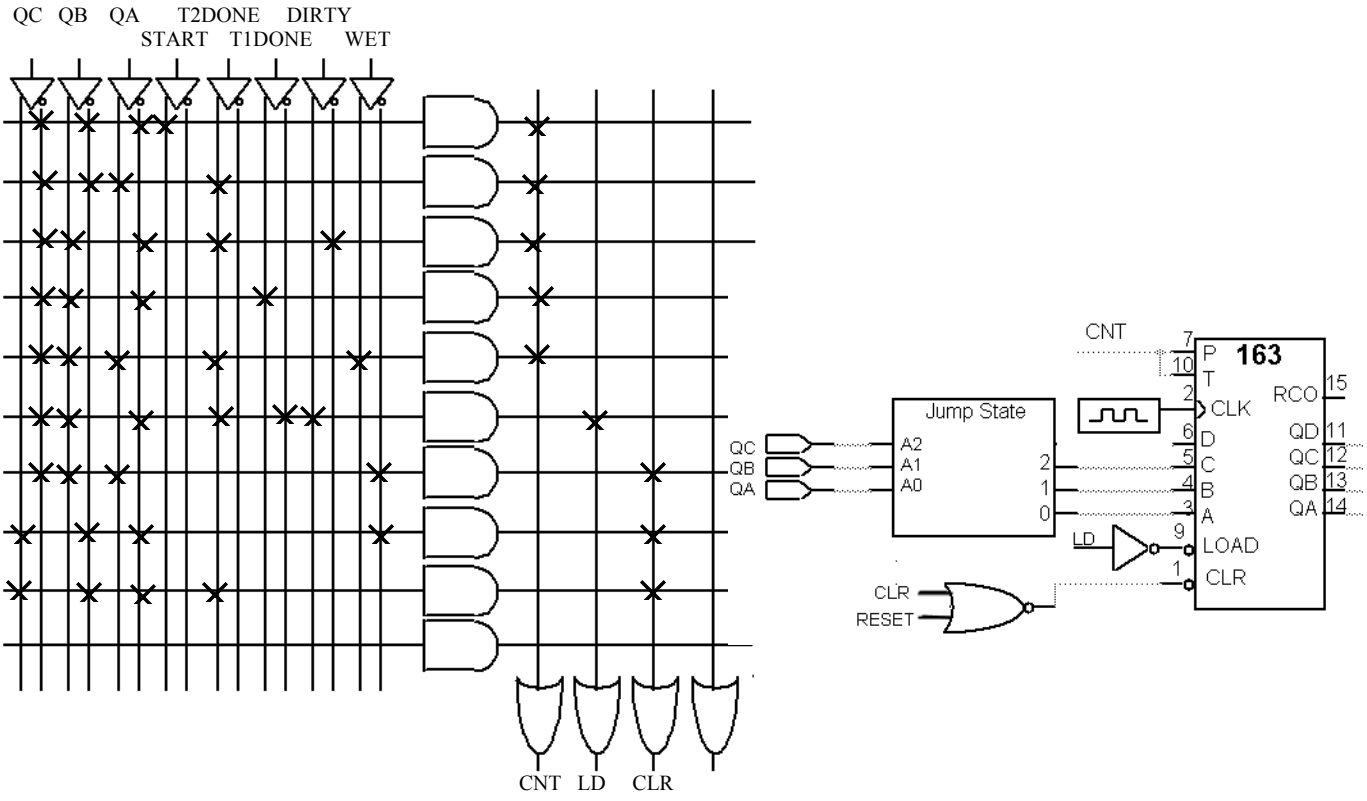
T2START = WAIT•START + WASH•T2DONE + RINSE•(T2DONE•DIRTY' + T1DONE) + RINSE•DIRTY•T2DONE+DRY1•T2DONE•WET

WASH = WAIT•START + WASH•T2DONE'

RINSE = WASH•T2DONE + RINSE•T2DONE'

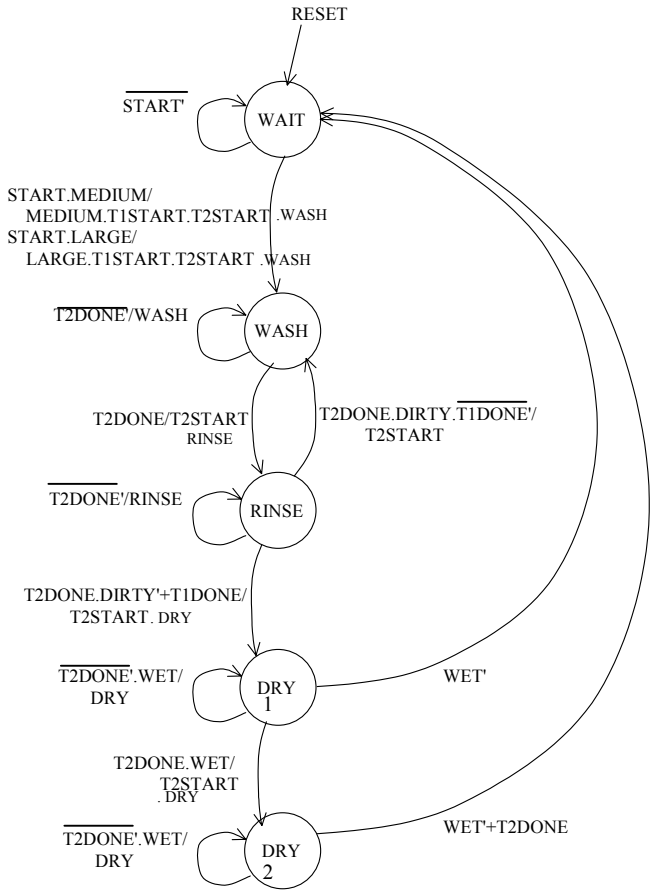
DRY = RINSE•(T2DONE•DIRTY'+T1DONE)+DRY1•WET+ DRY2•T2DONE'•WET

e) Draw a circuit diagram for the jump counter. You do not need to show the output function implementations. Use the PLA below to implement your control functions, and use a 74163 4-bit counter as the state storage element.



Name: Email: ID#

5) You are going to re-implement the washing machine controller from the previous problem using vertical microcode.



a) List all sets of mutually exclusive outputs

(MEDIUM, LARGE) and (WASH, RINSE, DRY)

b) Assuming you use 1-hot encoding for all non-mutually exclusive output groupings, what is the minimum number of bits you need to encode the outputs?

4 (T1START, T2START, MEDIUM/LARGE, WASH/RINSE/DRY)

c) Assume you need a 5-bit ROM address to encode the machine. Write out the format (as on slide 39 of the controller implementation notes) of the RT and BJ instructions for this machine assuming (b).

BJ = Type (1 bit): Condition Select (3 bits): Compare (1 bit): Address (5 bits)
Condition Selects: START=000, MEDIUM=001, LARGE=010, T2DONE=011, T1DONE=100, DIRTY=101, WET=110

RT = Type (1 bit): Output (4 bits): Unused (5 bits)
Output = T1START:T2START: MEDIUM/LARGE: WASH/RINSE/DRY