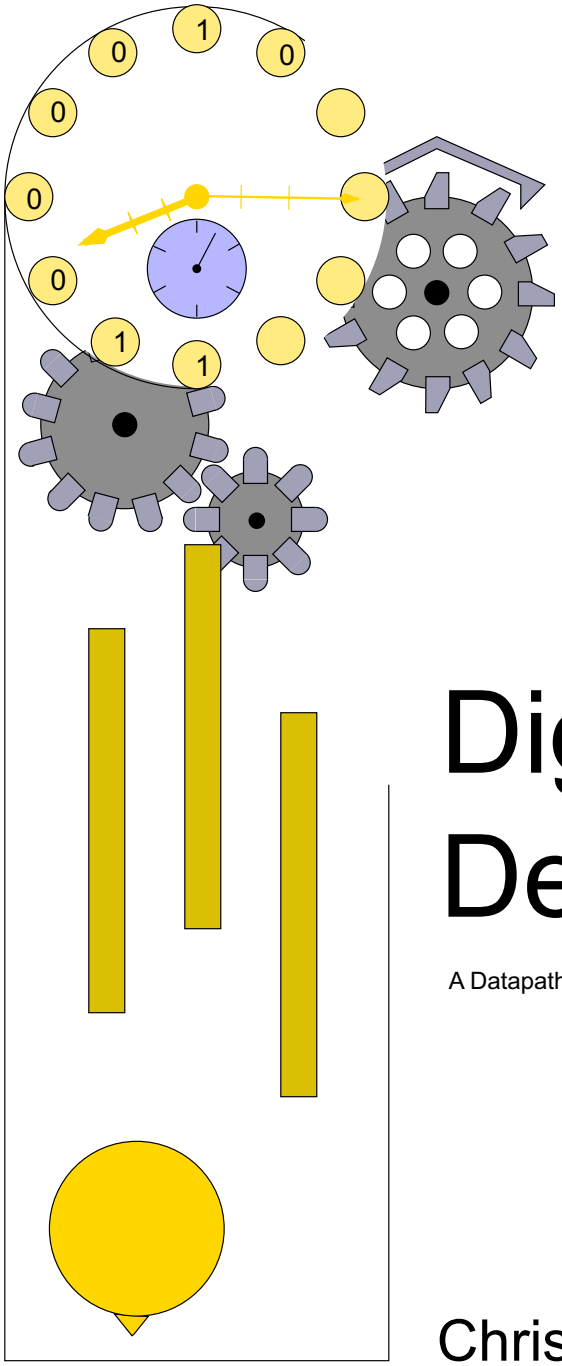


Digital Design

A Datapath and Control Approach

Chris Coulston

October 3, 2024

Contents

0.1 Sequential Circuits

Helpful Stuff

When the bit is stored.

1. Latches - a latch continuously samples its inputs. Latches do not have a clock input.

2. Clocked latches - a clocked latch samples its inputs when the clock input equals 1. When the clock input equals 0 the clocked latch does not change the currently stored bit.

3. Flip flops - a flip flop samples its input when the its clock input rises. The clock is said to rise when it goes from a logic 0 to a logic 1. When the clock input is not rising the flip flop does not change the currently stored bit.

How the input(s) is transformed into a stored bit.

D	Q+
0	0
1	1

T	Q+
0	Q
1	Q'

S	R	Q+
0	0	Q
0	1	0
1	0	1
1	1	x

J	K	Q+
0	0	Q
0	1	0
1	0	1
1	1	Q'

The *When* and *How* dimensions are arranged in the following table.

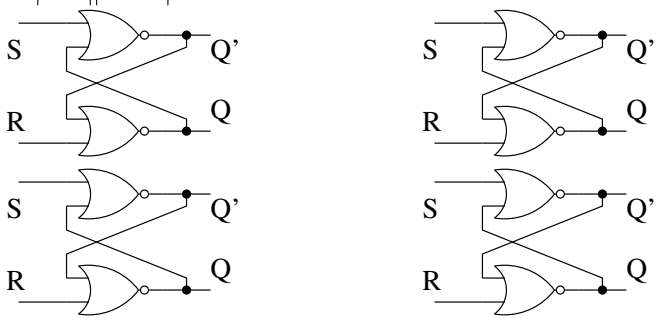
	Latch	Clocked Latch	Flip Flop
D			
T			
SR			
JK			

Problems

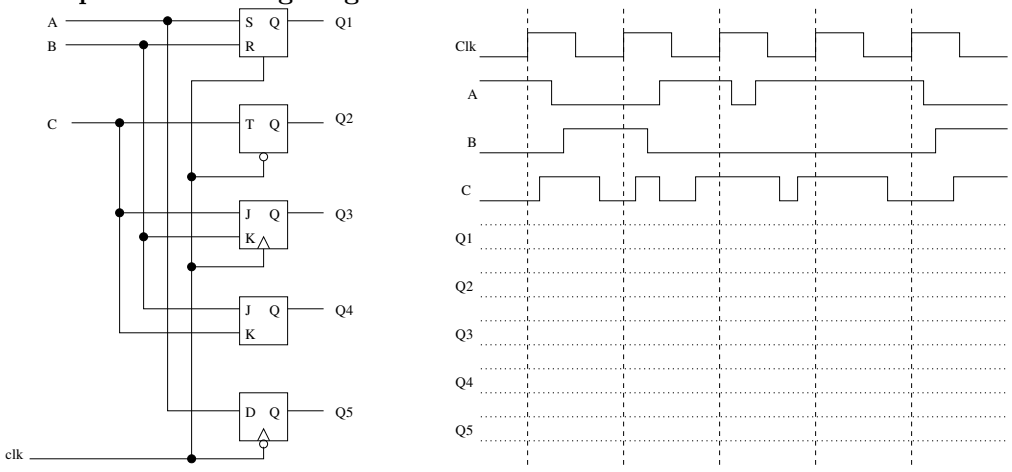
Here are some problems to work on.

Complete the state table with the help of the following figures.

S	R	Q^+	Q'^+
0	0		
0	1		
1	0		
1	1		



Complete the timing diagram.



0.2 Sequential Building Blocks

Helpful Stuff

Here are the devices introduced in this chapter.

Nomenclature:	N-bit register
Data Input:	N-bits vector $D = d_{N-1} \dots d_1 d_0$.
Data Output:	N-bit vector $Q = q_{N-1} \dots q_1 q_0$
Control:	1-bit c
Status:	none
Others:	1-bit edge-sensitive clock. 1-bit asynchronous active low reset.
Behavior:	

reset	clk	C	D	Q^+	comment
0	x	x	x	0	reset
1	0,1,falling	x	x	Q	hold
1	rising	0	x	Q	hold
1	rising	1	D	D	load

Nomenclature:	N-bit shift register with parallel load
Data Input:	N-bits vector $D = d_{N-1} \dots d_1 d_0$.
Data Output:	N-bit vector $Q = q_{N-1} \dots q_1 q_0$
Control:	2-bits $c = c_1 c_0$
Status:	none
Others:	1-bit edge-sensitive clock. 1-bit asynchronous active low reset.
Behavior:	

reset	clk	C	D	Q^+	comment
0	x	xx	x	0	reset
1	0,1,falling	xx	x	Q	hold
1	rising	00	x	Q	hold
1	rising	01	x	$Q >> 1$	shift right
1	rising	10	x	$Q << 1$	shift left
1	rising	11	x	D	load

Nomenclature:	N-bit counter with parallel load
Data Input:	N-bits vector $D = d_{N-1} \dots d_1 d_0$.
Data Output:	N-bit vector $Q = q_{N-1} \dots q_1 q_0$
Control:	2-bits $c = c_1 c_0$
Status:	none
Others:	1-bit edge-sensitive clock. 1-bit asynchronous active low reset.
Behavior:	

reset	clk	C	D	Q^+	comment
0	x	xx	x	0	reset
1	0,1,falling	xx	x	Q	hold
1	rising	00	x	Q	hold
1	rising	01	x	D	count up
1	rising	10	D	D	count up
1	rising	11	x	D	load

Nomenclature:	three state buffer
Data Input:	1-bit X.
Data Output:	1-bit Y
Control:	1-bit c
Status:	none
Others:	none
Behavior:	Output equals input when $C = 1$ otherwise output disconnected from input.

Nomenclature:	NxM RAM (random access memory)
Data Input:	M-bit vector $D = d_{M-1} \dots d_1 d_0$ $log_2(N)$-bit address $A = a_{log_2(N)-1} \dots a_1 a_0$
Data Output:	M-bit vector $D = d_{M-1} \dots d_1 d_0$
Control:	1-bit CS (chip select), RE (Read enable), WE (write enable)
Status:	none
Others:	none

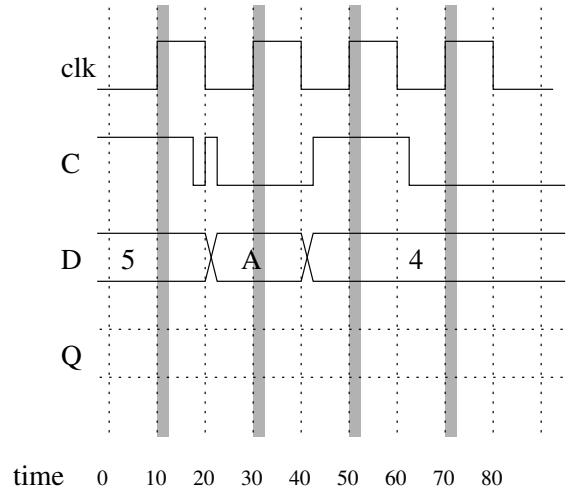
Behavior:

A	CS	RE	WE	D	Note
x	0	x	x	Z	RAM deactivated
x	1	0	0	Z	RAM deactivated
A	1	0	1	D	RAM[A] = D (write)
A	1	1	0	RAM[A]	D = RAM[A] (read)

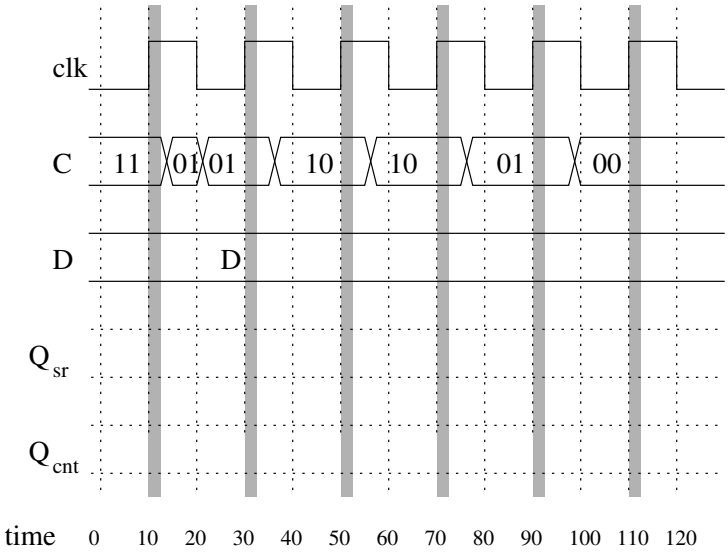
	Left	Right
Arithmetic	$x_2x_1x_00$	$x_3x_3x_2x_1$
Circular	$x_2x_1x_0x_3$	$x_0x_3x_2x_1$
Logical	$x_2x_1x_00$	$0x_3x_2x_1$

Problems

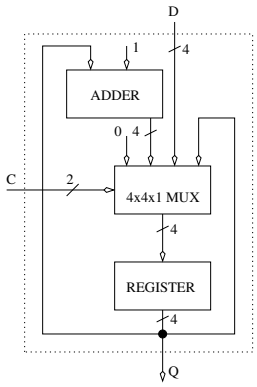
Plot Complete the timing diagram for the register. You may assume that Q is initially 0.



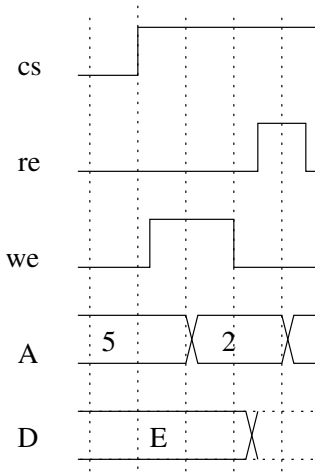
Plot Let Q_{sr} is the output of a logical shift register (assume that 0 is shifted into the vacated bit position. Let Q_{cnt} is the output of a counter.


clk
C
11
01
01
10
10
01
00
D
D
Q sr
Q cnt
time
0
10
20
30
40
50
60
70
80
90
100
110
120

Label the mux inputs. Make the resulting circuit operate according to the truth table shows at left.

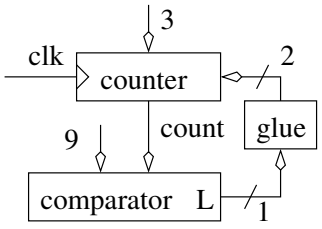


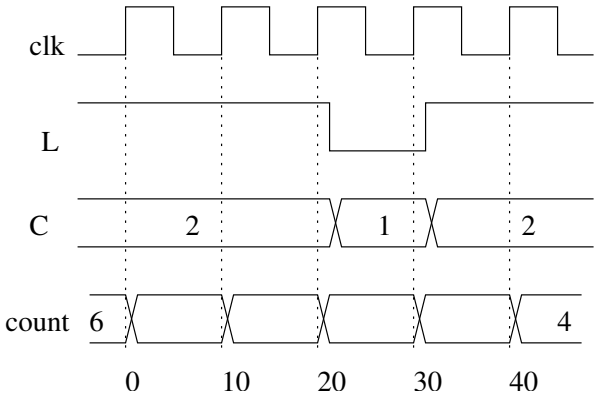
reset	clk	C	D	Q^+	comment
0	x	xx	x	0	reset
1	0,1,falling	xx	x	Q	hold
1	rising	00	x	Q	hold
1	rising	01	x	0	clear
1	rising	10	D	$Q+1$	count up
1	rising	11	x	D	load



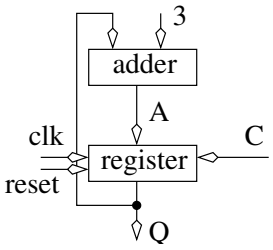
Complete the timing diagram. Note any changes in the RAMs content.

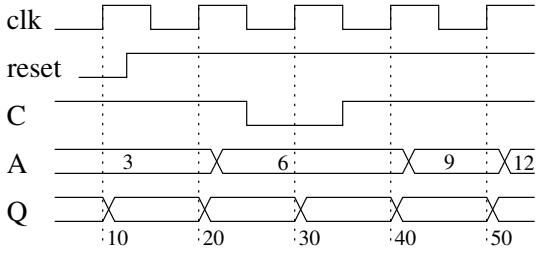
Complete the timing diagram. Use the counter control table from page 128. Assume that c1=L and c0=L'.



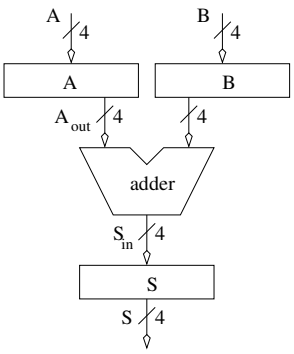


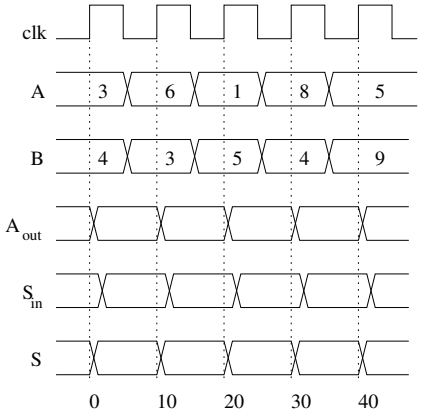
Complete the timing diagram. Put "u" in spaces where the output is undefined.



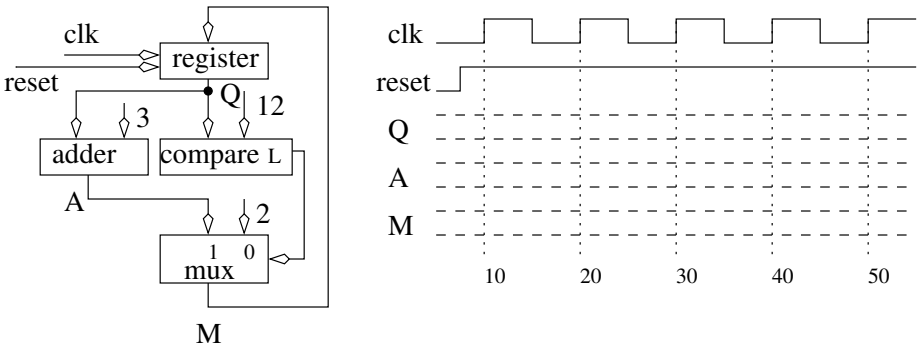


Complete the timing diagram. Put "u' in spaces where the output is undefined.

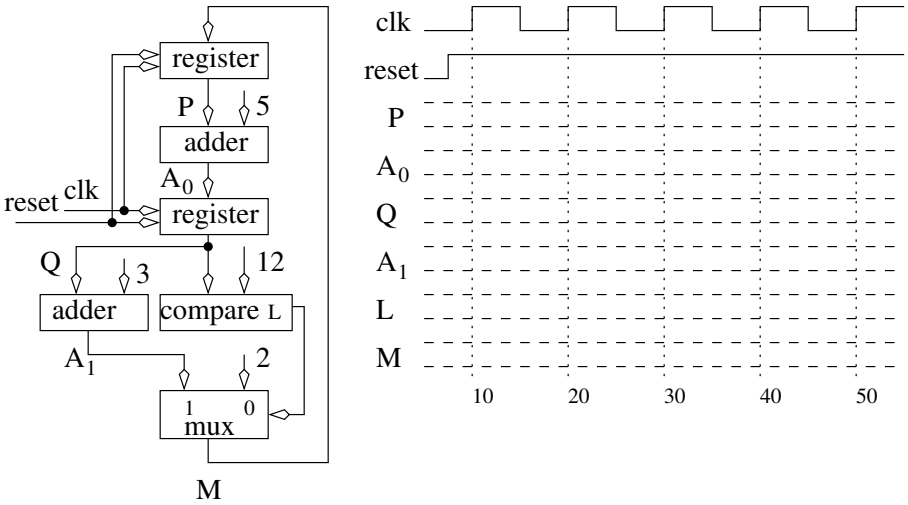




Complete the timing diagram for the following circuit. Note the labels of the signals.

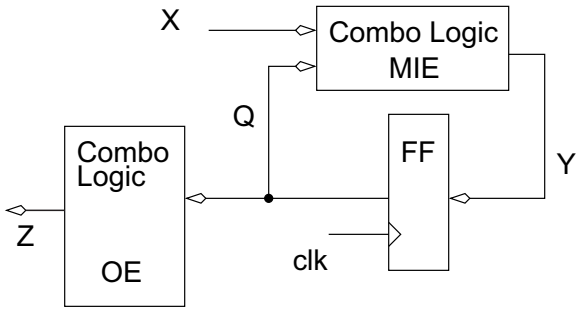


Complete the timing diagram for the following circuit. Note the labels of the signals.



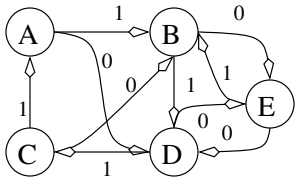
0.3 Finite State Machines

Helpful Stuff



Problems

Convert the state diagram into a state table. The input variable is X.



CURRENT STATE	X=0	X=1
A		
B		
C		
D		
E		

entries should be next state

Convert the state table into a transistion kmap. Use the state assignment provided.

State	$Q_2Q_1Q_0$
A	010
B	101
C	100
D	001
E	011

$Q_2Q_1 \backslash Q_0X$	00	01	11	10
00				
01				
11				
10				

entries should be $Q_2^+Q_1^+Q_0^+$

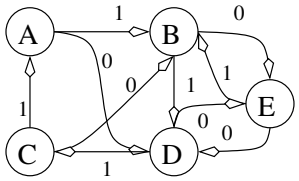
Determine the memory input equations. Use the transition kmap given in the previous problem.

$Q_2Q_1 \backslash Q_0X$	00	01	11	10
00				
01				
11				
10				

$Q_2Q_1 \backslash Q_0X$	00	01	11	10
00				
01				
11				
10				

$Q_2Q_1 \backslash Q_0X$	00	01	11	10
00				
01				
11				
10				

Write the MIEs Assume a 1's hot encoding of the states.



$D_A =$
$D_B =$
$D_C =$
$D_D =$
$D_E =$
$Z =$

Write the MIEs Use the state Assume a 1's hot encoding of the states.

Q	X=0	X=1
A	C	C
B	D	A
C	D	E
D	A	E
E	C	B

$D_A =$
$D_B =$
$D_C =$
$D_D =$
$D_E =$
$Z =$

Given the state table and state assignment determine the MIEs.

State	ND=00	ND =
0	0	10
5	5	15
10	10	20
15	15	25
20	20	30
25	25	35
30	30	35
35	0	0

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 0$

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 1$

Entries are $Q_2^+Q_1^+Q_0^+$

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 0$

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 1$

Entries are $Q_2^+ = D_2$

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 0$

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 1$

Entries are $Q_1+ = D_1$

$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 0$

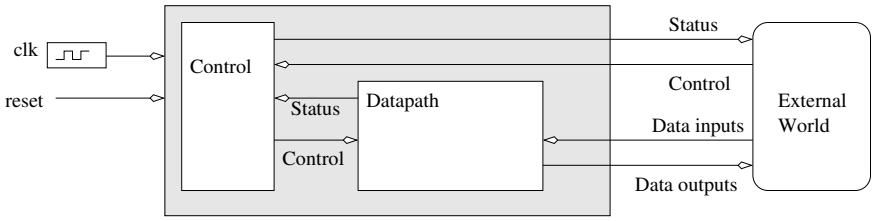
$Q_1Q_0\backslash ND$	00	01	11	10
00				
01				
11				
10				

$Q_2 = 1$

Entries are $Q_0+ = D_0$

0.4 Datapath and Control

Helpful Stuff

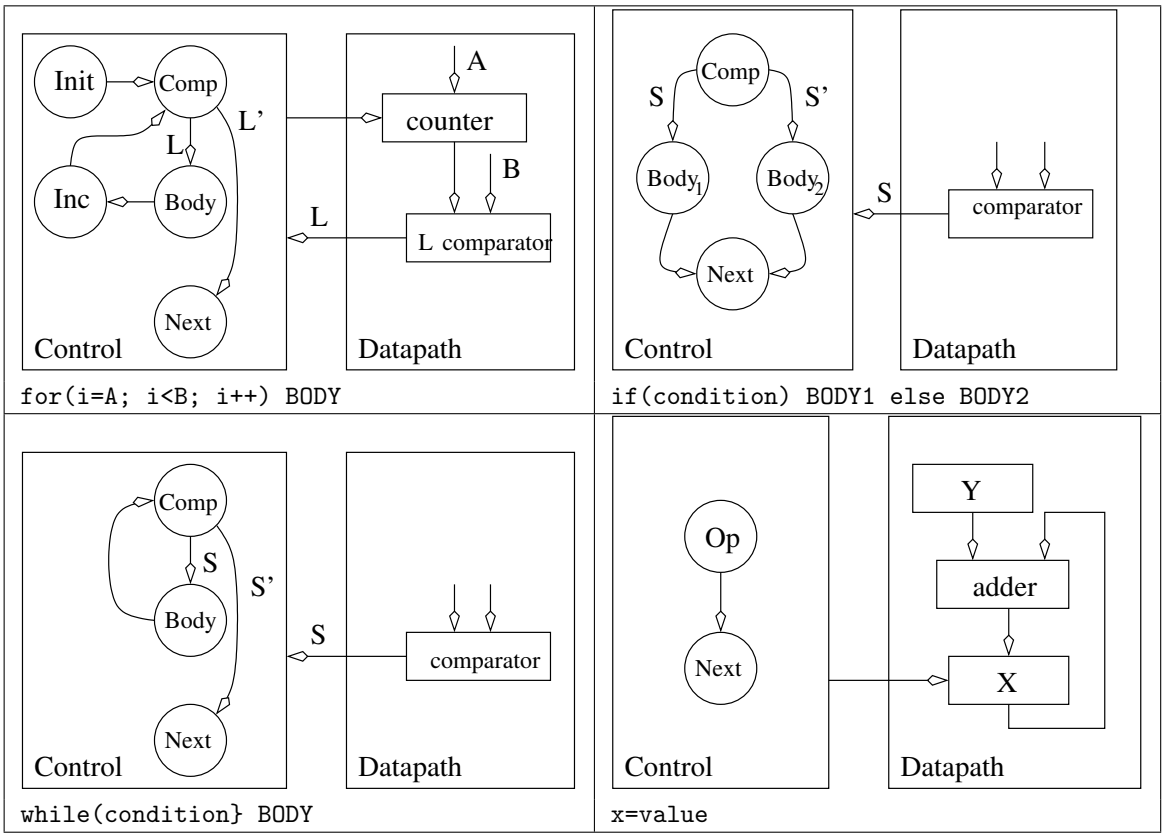


Mini-C statements

- if (condition) then BODY_1 else BODY_2
- for (i=A; i<B; i + = 1) BODY
- while(condition) BODY
- X = value

where BODY contains 0 or more statements.

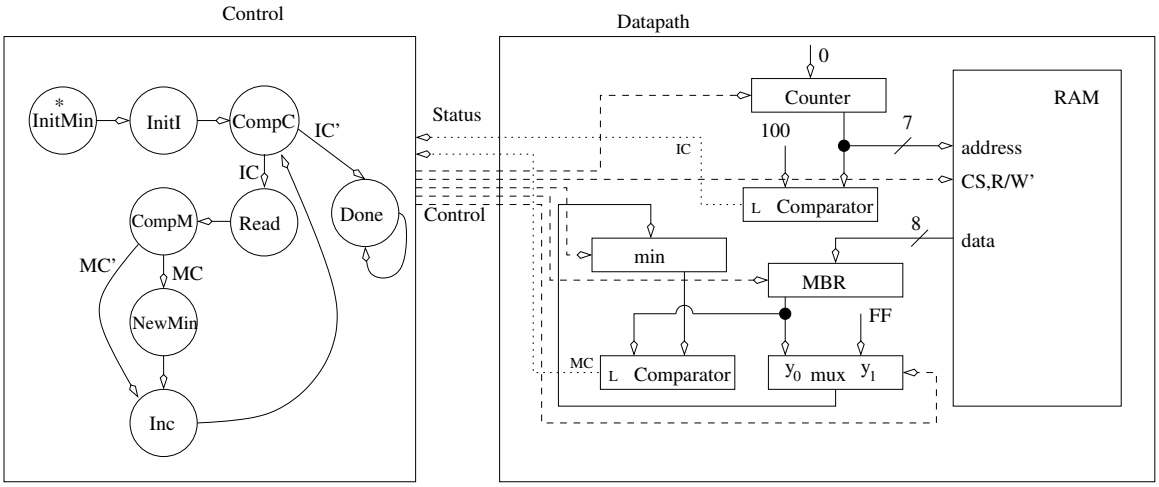
Device	Data in	Data out	Status	Control
N:M Decoder	1 bit	M bits		N bits
N:1 Mux	N bits	1 bit		$\log_2(N)$ bits
MxNx1 Mux	N, each M bits	M bits		$\log_2(N)$ bits
N bit adder	2, each N bits	N bits	Overflow	
N bit add/sub	2, each N bits	N bits	Overflow	1 bit
N bit comparator	2, each N bits		3 bits	
BCD to 7-segment	4-bits	7-bits		
N-bit priority encoder	N-bits	$\log(N)$-bits		
N bit register	N bits	N bits		1 bit
N bit shift register	N bits	N bits		2 bits
N bit counter	N bits	N bits		2 bits
Three state buffer	N bits	N bits		1 bit
N:M RAM	$\log_2(N)$ bits, M bits	M bit		3 bits
N-bit Bus transceiver	N bits	N bits		2 bit



Problems

Minimum Search Design a digital circuit that looks for the smallest 8-bit integer in a 128x8 bit RAM. The numbers are stored at addresses $0 \ldots 99$, you may assume that the RAM is preloaded with data.

```
1.  min = 0xFF;              // Set the min reg to largest value
2.  for (i=0; i<100; i++) {  // Search through the entire array
3.      MBR=RAM[i];          // read an 8-bit value from the RAM
4.      if (MBR<min) then    // If MBR is smaller than min
5.          min = MBR;       //   then set min to the smallest value
6   } // end for
```

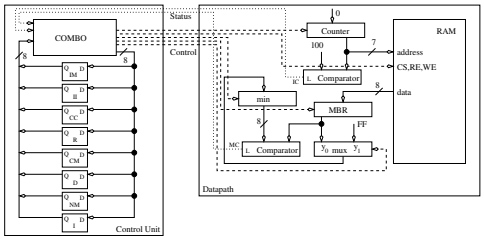


State	CS	RE	WE	Reg Min	Min mux	Counter	MBR
	0 off	0 idle	0 idle	0 hold	0 load FF	00 hold	0 hold
	1 active	1 read	1 write	1 load	1 load MBR	01 load	1 load
						10 count	
						11 reset	
InitMin							
InitI							
CompC							
Read							
CompM							
NewMin							
Inc							
Done							

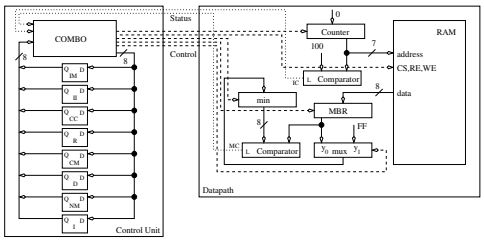
$D_{IM} =$
$D_{II} =$
$D_{CC} =$
$D_R =$
$D_{CM} =$
$D_{NM} =$
$D_I =$
$D_D =$

$Z_{CS} =$
$Z_{RE} =$
$Z_{WE} =$
$Z_{RM} =$
$Z_{MM} =$
$Z_{C1} =$
$Z_{C0} =$
$Z_{MBR} =$

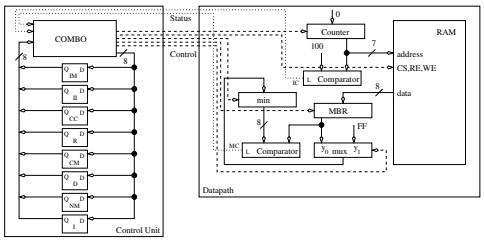
Shade the active FF and any BBBs which are read or written.



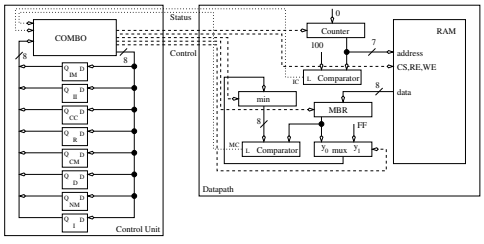
1. State **InitMin**



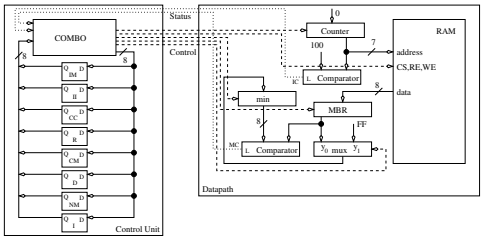
2. State **InitI**



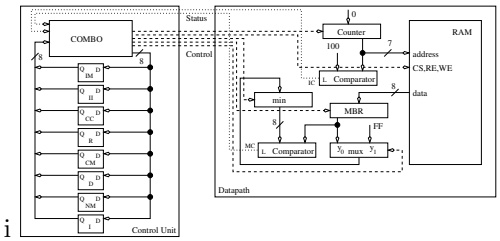
3. State **CompC**



4. State **Read**

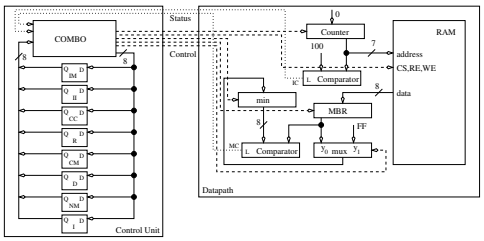


5. State **CompM**

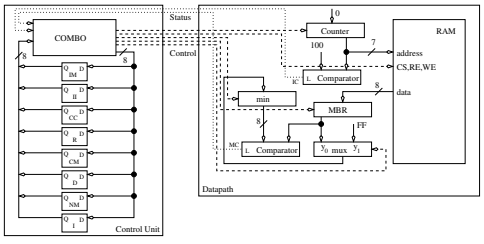


i

6. State **NewMin**



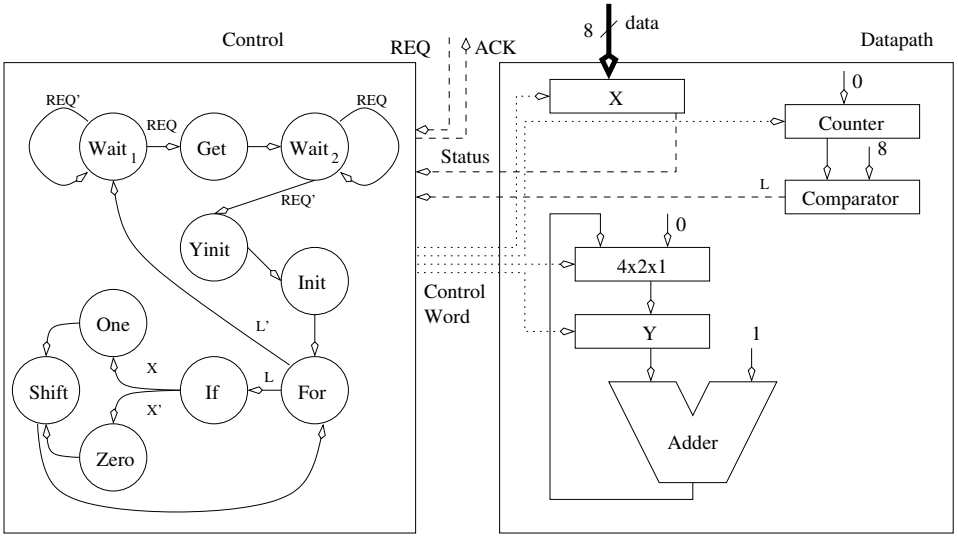
7. State **Inc**



8. State **CompC**

Bit Counter Design a digital circuit that counts the number of 1's in an 8 bit number X and puts the result into a register Y. The 8 bit number is provided by an external user using a two-line handshake protocol. The digital circuit is to play the role of the passive consumer. Your circuit should do this task forever. That is, after counting the number of 1's, you circuit should read in a new value of X.

```
1.  while(true) {                // Forever
2.      while(REQ==0);           // Wait for the REQ signal
3.      X = data;                // When we get a REQ latch data
4.      ACK = 1;
5.      while(REQ == 1);         // Wait for REQ to go low
6.      ACK = 0;                 // then drop the ACK
7.      Y = 0;                   // Clear the bit count
8.      for (i=0; i<8; i++) {    // For each bit of X
9.          if (X(0) == 1) then // If the LSB of X is a 1
10             Y = Y + 1;       //   then increment Y
11         X = X >> 1;          // Shift X to the right 1 bit
12     } // end for
13  } // end while
```



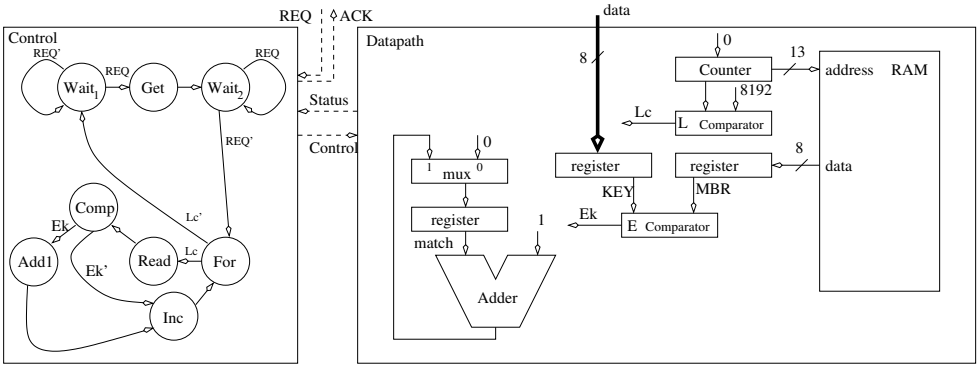
State	ACK	X	Reg Y	Y mux	Counter
		00 hold	0 hold	0 load 0	00 hold
		01 lsr			01 load
		10 lsl	1 load	1 load Add	10 count
		11 load			
Wait1					
Get					
Wait2					
Yinit					
Init					
For					
If					
One					
Zero					
Shift					

$D_{W1} =$
$D_G =$
$D_{W2} =$
$D_{YI} =$
$D_I =$
$D_F =$
$D_{If} =$
$D_{One} =$
$D_{Zer} =$
$D_{Sft} =$

$Z_{ACK} =$
$Z_X =$
$Z_{RegY} =$
$Z_{Ymux} =$
$Z_{C1} =$
$Z_{C0} =$

RAM counter Build a circuit that reads in an 8 bit value, KEY, using a two line handshake; your circuit is a passive consumer. Your circuit should search an 8kx8 RAM counting the number of words that match KEY. It takes one full clock cycle to read the RAM. You may assume that the RAM is pre-loaded with data. Your circuit should do this task forever.

```
1.    while(1) {
2.        while(REQ == 0);
3.        KEY = data;
4.        ACK = 1;
5.        while(REQ == 1);
6.        ACK = 0;
7.        match = 0;
8.        for(i=0; i<8191; i++) {
9.            MBR = RAM[i];
10.            if (MBR == KEY) {
11.                match=match+1;
12.            } // end if
13.        } // end for
14.    } // end while
```



State	ACK	mux	Reg match	Reg KEY	Counter	MBR	CS	RE	WE
	0	0 pass 0	0 hold	0 hold	00 hold	0 hold	0 NULL	0 NULL	0 NULL
					01 load				
	1	1 match+1	1 load	1 load	10 count	1 load	1 active	1 read	1 write
Wait1									
Get									
Wait2									
For									
Read									
Comp									
Add1									
Inc									

Extra Design a circuit that moves M consecutive words from address S (source) to address D (destination). For example, if $M = 4$, $S = 3EA$ and $D = 1FE$ then the circuit would move words 3EA, 3EB, 3EC and 3ED to address 1FE, 1FF, 200 and 201. Each of M, S, D is preloaded into a register. While this problem appears simple, its really rather treacherous. The circuit will have to handle cases where $S + M > D$. In such a case the order of the data movement must be carefully planned. In order to simplify the design, assume that $S < D$. Turn in an algorithm, datapath and control, the control word, MIEs and OEs. These circuit will need a three-state buffer to be able to both read and write to the RAM. Do not worry about the sizes of the registers or RAM.