#1 Given: What follows is the VMLAB project information and AVR code for an 8-bit up/down counter, where the count direction is controlled by the state of PIN B, bit 0…

Find: Modify this code to create a 3-bit up counter that stops the count when PIN D, bit 1, is LOW. Use the 3 least significant bits of R16.

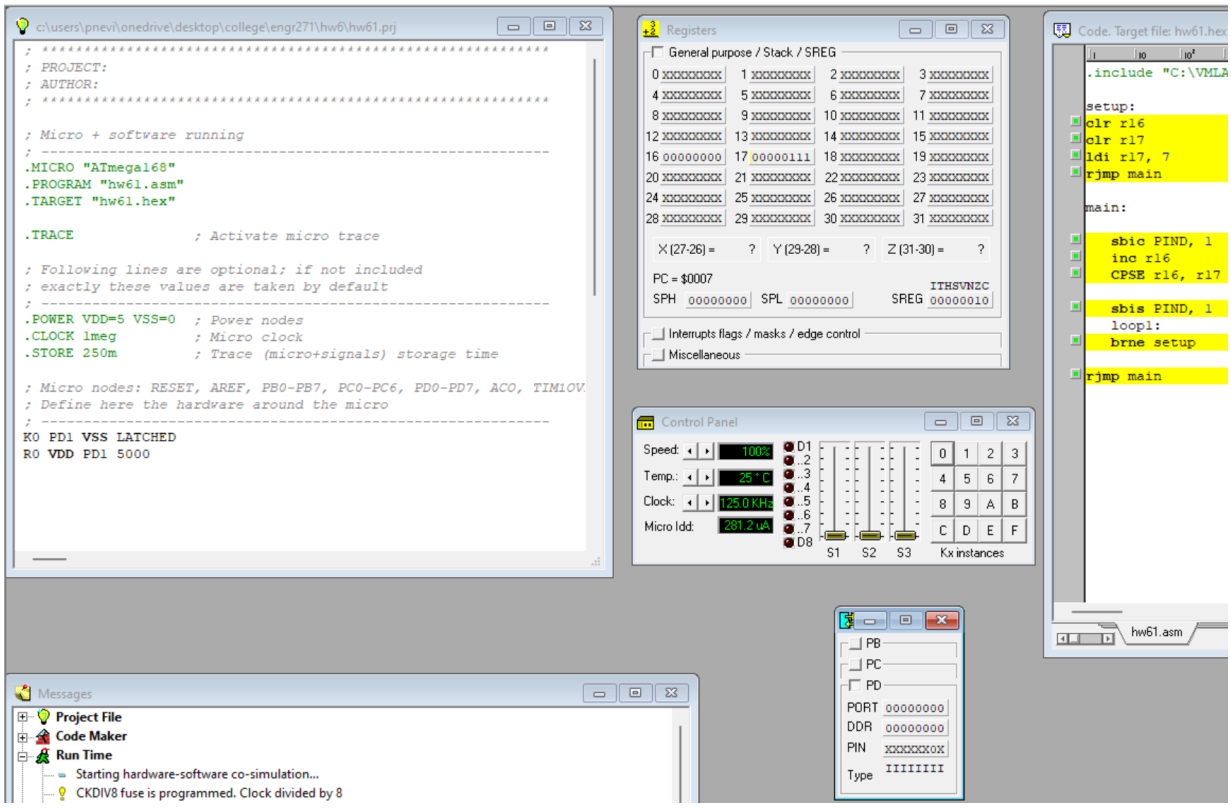
Solution:

.prj file:

```
K0 PD1 VSS LATCHED
R0 VDD PD1 5000
```

.hex file:

```
.include "C:\VMLAB\include\m168def.inc"

setup:
clr r16
clr r17
ldi r17, 7

rjmp main

main:
        sbic PIND, 1
        inc r16
        CPSE r16, r17

  sbis PIND, 1
        loop1:
        brne setup

rjmp main
```


c:\users\pnevi\onedrive\desktop\college\engr271\hw6\hw61.prj
; PROJECT:
; AUTHOR:
; Micro + software running
.MICRO "ATmega168"
.PROGRAM "hw61.asm"
.TARGET "hw61.hex"
.TRACE ; Activate micro trace
; Following lines are optional; if not included
; exactly these values are taken by default
.POWER VDD=5 VSS=0 ; Power nodes
.CLOCK 1meg ; Micro clock
.STORE 250m ; Trace (micro+signals) storage time
; Micro nodes: RESET, AREF, PB0-PB7, PC0-PC6, PD0-PD7, ACO, TIM1OV
; Define here the hardware around the micro
K0 PD1 VSS LATCHED
R0 VDD PD1 5000
Registers
General purpose / Stack / SREG
PC = $0007
SPH 00000000 SPL 00000000 SREG 00000010
ITHSVNZC
Interrupts flags / masks / edge control
Miscellaneous
Control Panel
Speed: 100%
Temp.: 25 °C
Clock: 125.0 KHz
Micro Idd: 281.2 uA
S1 S2 S3 Kx instances
Code. Target file: hw61.hex
.include "C:\VMLA
setup:
clr r16
clr r17
ldi r17, 7
rjmp main
main:
sbic PIND, 1
inc r16
CPSE r16, r17
sbis PIND, 1
loop1:
brne setup
rjmp main
hw61.asm
PB
PC
PD
PORT 00000000
DDR 00000000
PIN XXXXXXXX
Type IIIIIIII
Messages
Project File
Code Maker
Run Time
Starting hardware-software co-simulation...
CKDIV8 fuse is programmed. Clock divided by 8

#2 Given: Single 4-bit Johnson counter. Assume the counter is reset to all 0's before the sequence starts.

Find: Design an 8-bit one-hot counter. What would be the problem with trying to build a 16-bit one-hot counter from a single 4-bit Johnson counter?

Solution:

Truth Table

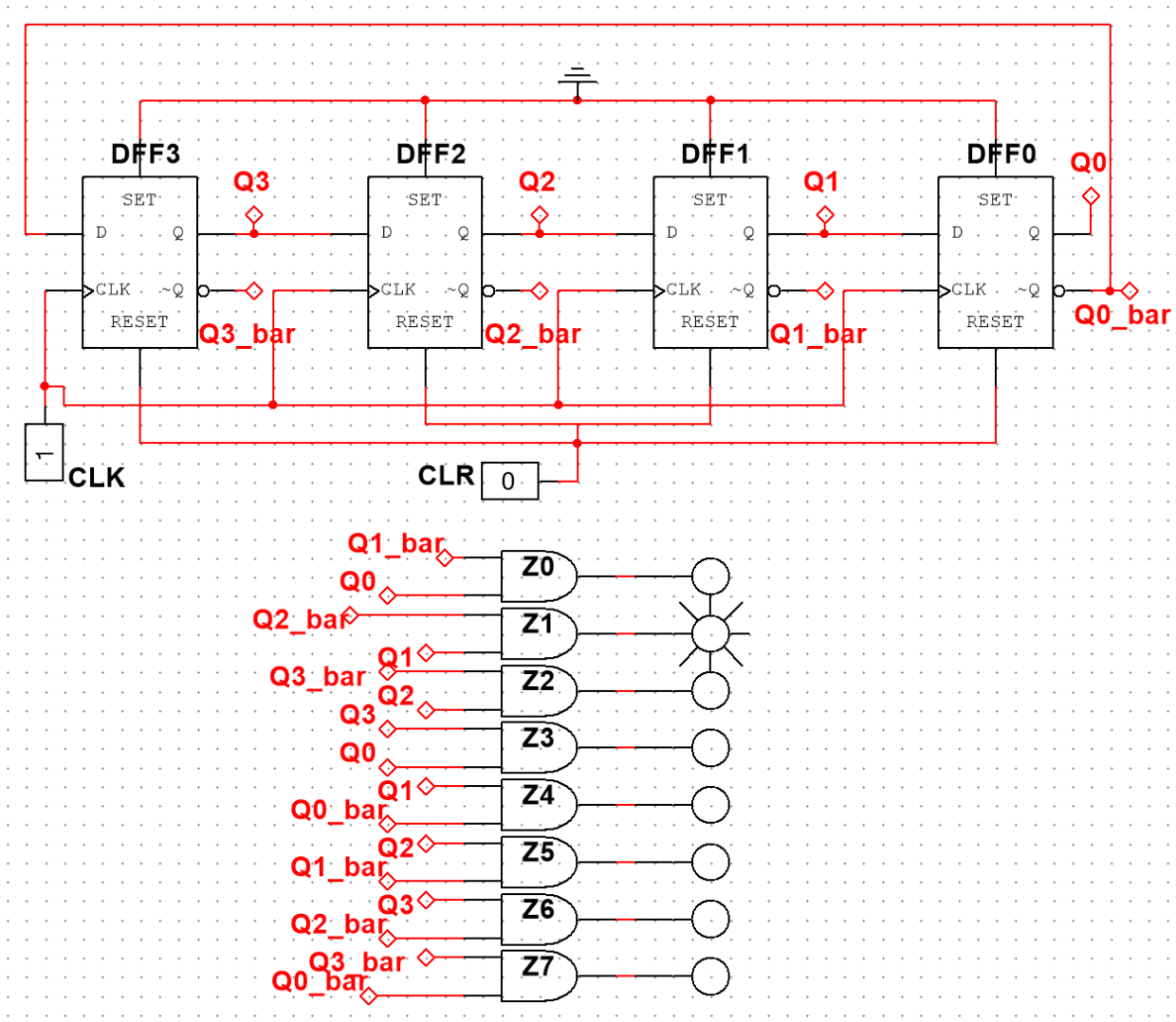
Qx = output from DFFs (Johnson Counter)
Zx = 8-bits of the One Hot Counter

State	Q3	Q2	Q1	Q0	Z7	Z6	Z5	Z4	Z3	Z2	Z1	Z0
0	0	0	0	0	1	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0	0	0
2	1	1	0	0	0	0	1	0	0	0	0	0
3	1	1	1	0	0	0	0	1	0	0	0	0
4	1	1	1	1	0	0	0	0	1	0	0	0
5	0	1	1	1	0	0	0	0	0	1	0	0
6	0	0	1	1	0	0	0	0	0	0	1	0
7	0	0	0	1	0	0	0	0	0	0	0	1

Using a state machine solver, we have:

Z7 = (Q3' Q0')
Z6 = (Q3 Q2')
Z5 = (Q2 Q1')
Z4 = (Q1 Q0')
Z3 = (Q3 Q0)
Z2 = (Q3' Q2)
Z1 = (Q2' Q1)
Z0 = (Q1' Q0)

There is no problem building a 16-bit One Hot Counter from a single 4-bit Johnson Counter. With the Johnson Counter, we have 2^4 = 16 states, and we need 16 states for each of the 16 bits in the 16-bit One Hot Counter. So there is no problem.



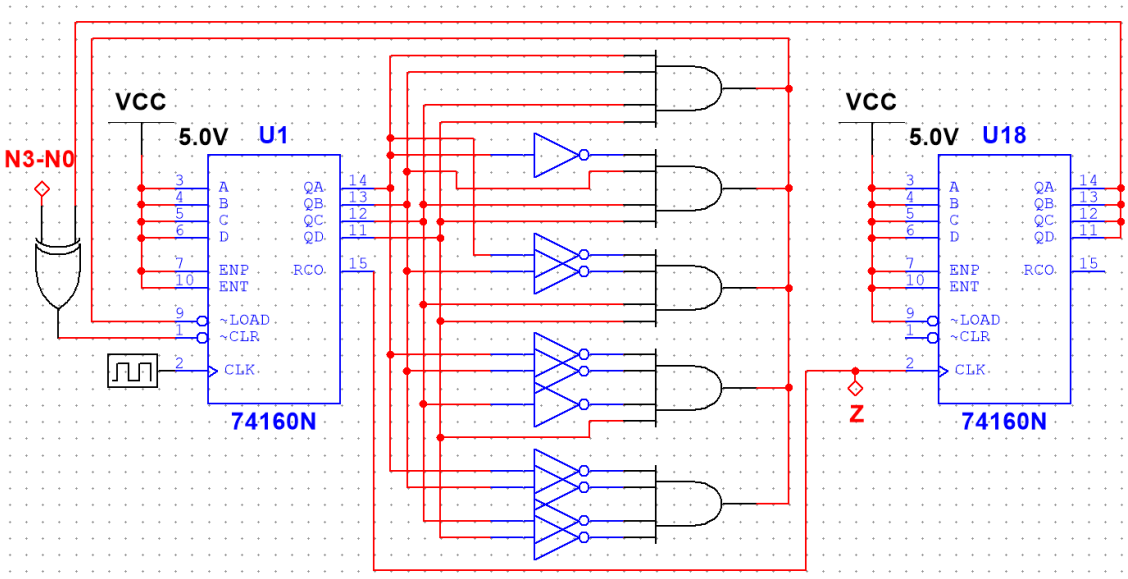
#3 Given: Circuit from Homework 5, #5

Find: Modify this circuit so that Z produces N transitions in each 16-tick interval. The resulting circuit is called a binary rate multiplier and was once sold as a TTL MSI part, the 7497.

Solution:

This is not a "functioning" circuit in Multisim, this is just to display the schematic. You would need a BUS between the outputs of the AND4 gates and the LOAD' of U1 for it to work properly. I could not get it to work correctly in Multisim, so I just removed the BUS and added this disclaimer.

The output Z asserts when the count is 2, 4, 8 and 16. There will be N number of transitions. The second binary counter is being used to count the transitions, and the output Z is not asserted once the values of N3-0 and QD-A are equal. This is how we easily change the circuit so that Z produces N transitions in each 16-tick interval.

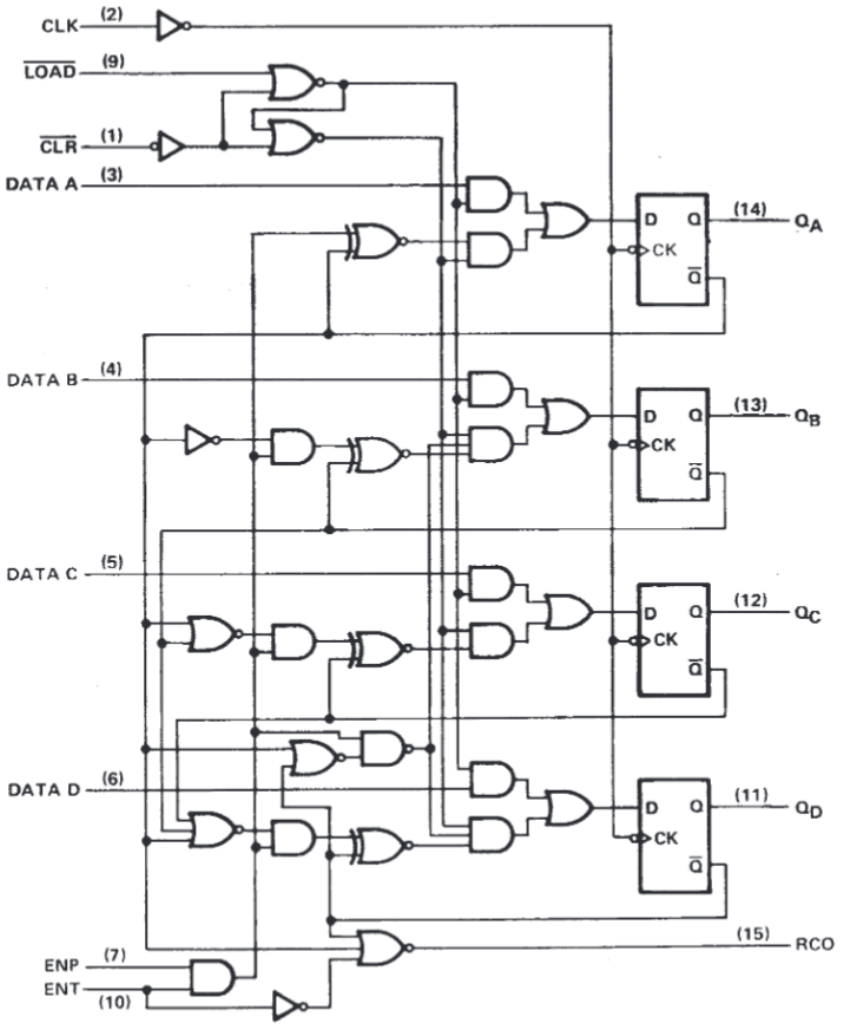


#4 Given: Datasheet with the internal logic diagram for a 74162 synchronous decade counter

Find: Write a state table and include its counting behavior in the normally unused states 10-15

Solution: I used this specific synchronous decade counter, the SN54S162

SN54S162, SN74S162 SYNCHRONOUS DECADE COUNTER



Pin numbers shown are for D, J, N, and W packages.

Present State	Next State	Behavior
1000	0100	Normal Count -4
0100	1100	Normal Count +8
1100	0010	Normal Counter -12
0010	1010	Normal Count +8
1010	0110	Normal Count -4
0110	1110	Normal Count +8
1110	0001	Normal Count -13
0001	1001	Normal Count + Ripple Carry Out +8
1001	0000	Normal Count -9
0000	1000	Normal Count +8
0011	x	Invalid State
0101	x	Invalid State
0111	x	Invalid State
1011	x	Invalid State
1101	x	Invalid State
1111	1000	Clear enabled

#5 Given: (From the Wakerly textbook, problems 11.62) Prove that X0 must appear on the right hand side of any LFSR feedback equation that generates a maximum-length sequence. (*Note:* Assume the LFSR bit ordering and shift direction are as given in the text; that is, the LFSR counter shifts right, toward the X0 stage.)

Find: Prove that X0 must appear on the right hand side of any LFSR feedback equation that generates a maximum-length sequence

Solution:

0 in the X0 spot can signify an invalid state, so we must always consider what is in the X0 position. Having X0 on the right side of the LFSR equation equates to always checking if the "first" decimal number is an invalid state or not; if it's a 0, it is paired with another X1-x to ensure it is indeed not an invalid state. Since X0 is the LSB of the LFSR, it must always be considered to have maximum length, since it denotes the decimal number 1. If you discount this, you will have skipped states without the 1 representation. IE: Taking the 1 out of the right side of the LFSR equation, you would have the states (decimal #) 2, 4, 8, 12, 16, etc.