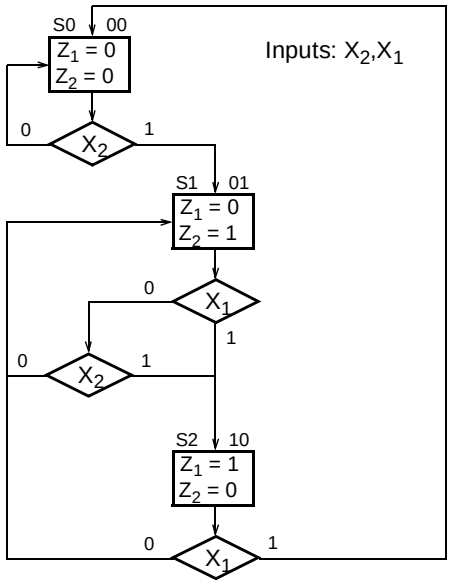
Problem Solutions to Problems Marked With a * in

Logic Computer Design Fundamentals, Ed. 2

C H A P T E R 8

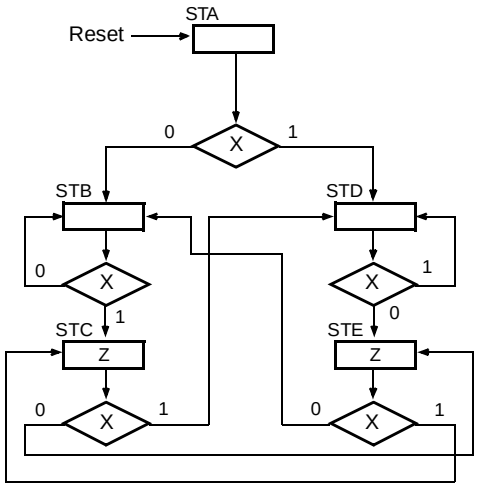
8-1.



8-2.

A:	0	1	1	0	1	1	
B:	1	1	1	1	0	0	
C:	0	1	0	1	0	1	
State:	ST1	ST1	ST2	ST3	ST1	ST2	ST3
Z:	0	0	1	1	0	0	

8-5.

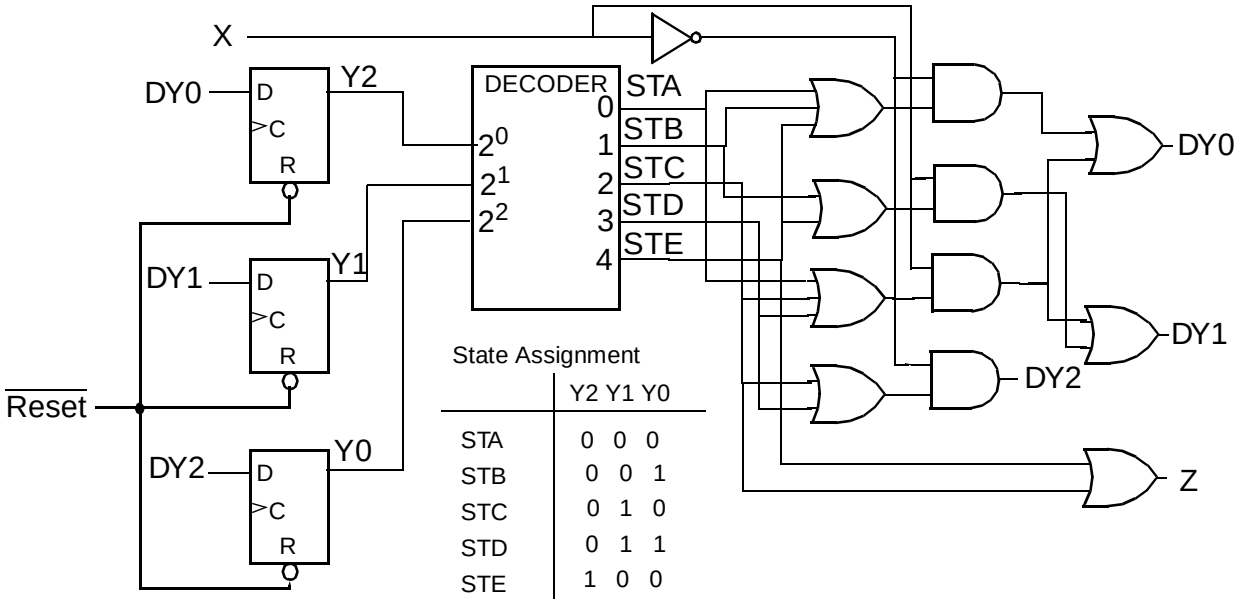


8-8.

ST1(t + 1) = ST1·$\overline{A}$ + ST2·B·C + ST3, ST2(t + 1) = ST1·A, ST3(t + 1)= ST2·($\overline{B}$ + $\overline{C}$), Z = ST2·B + ST3

For the D flip-flops, DSTi = STi(t + 1) and STi = Q(STi).Reset initializes the flip-flops: ST1 = 1, ST2 = ST3 = 0.

8-9.



State Assignment

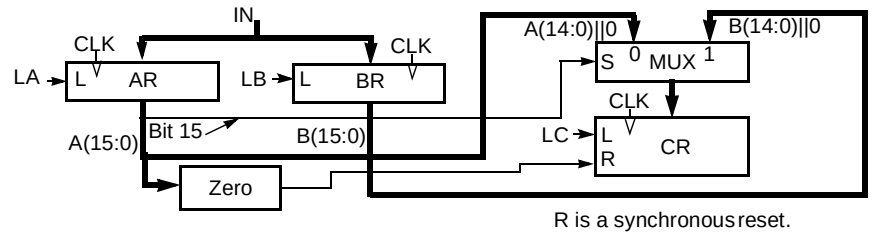
	Y2 Y1 Y0
STA	0 0 0
STB	0 0 1
STC	0 1 0
STD	0 1 1
STE	1 0 0

8-12.

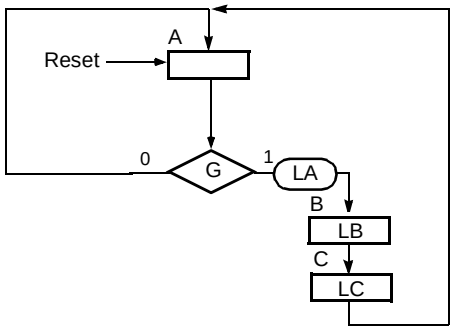
```
   100110    (38)
 × 110101  (× 53)
   100110
  000000
 100110
000000
100110
100110
11111011110   (2014)
```

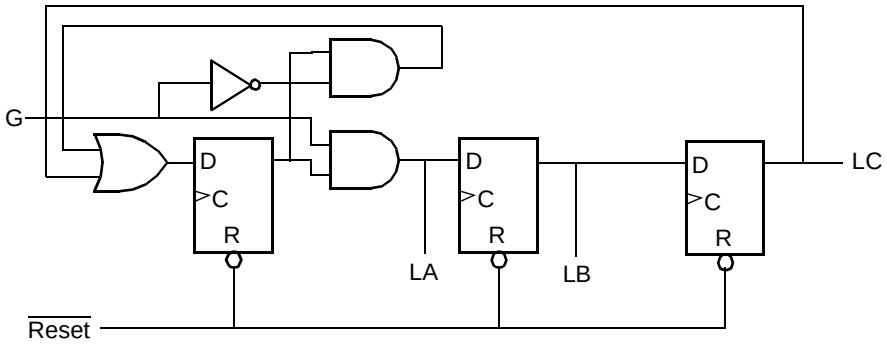
100110	
110101	
000000	Init PP
100110	Add
100110	After Add
0100110	After Shift
00100110	After Shift
100110	Add
10111110	After Add
010111110	After Shift
0010111110	After Shift
100110	Add
1100011110	After Add
01100011110	After Shift
100110	Add
11111011110	After Add
011111011110	After Shift

8-17.



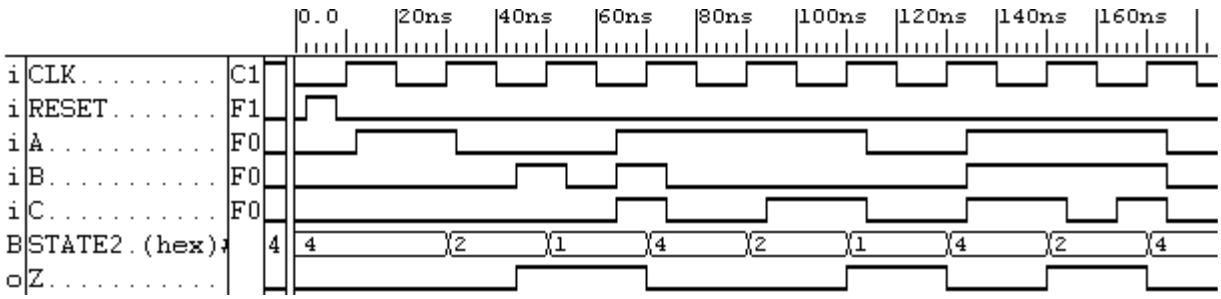
R is a synchronous reset.





8-20.

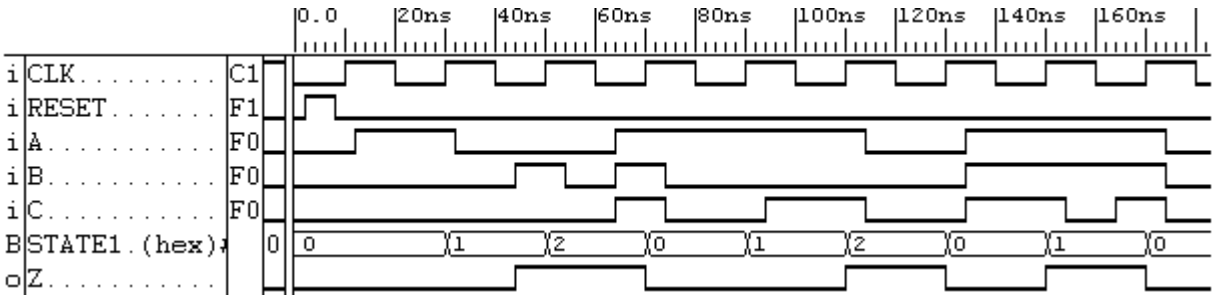
```
library IEEE;
use IEEE.std_logic_1164.all;

entity asm_820 is
	port (
		A, B, C, CLK, RESET: in STD_LOGIC;
		Z: out STD_LOGIC
	);
end asm_820;

architecture asm_820_arch of asm_820 is
	type state_type is (ST1, ST2, ST3);
	signal state, next_state : state_type;
begin

state_register: process (CLK, RESET)
begin
	if RESET='1' then --asynchronous RESET active High
		state <= ST1;
	elsif (CLK'event and CLK='1') then  --CLK rising edge
		state <= next_state;
	end if;
end process;

next_state_func: process (A, B, C, state)
begin
	case (state) is
		when ST1 =>
			if A = '0' then
				next_state <= ST1;
			else
				next_state <= ST2;
			end if;
		when ST2 =>
			if ((B = '1') and (C = '1')) then
				next_state <= ST1;
			else
				next_state <= ST3;
			end if;
		when ST3 =>
			next_state <= ST1;
	end case;
end process;

--Output Z only depends on the state and input 'B'
output_func: process (B, state)
begin
	case (state) is
		when ST1 =>
			Z <= '0';
		when ST2 =>
			if (B = '1') then
				Z <= '1';
			else
				Z <= '0';
			end if;
		when ST3 =>
			Z <= '1';
	end case;
end process;

end asm_820_arch;
```



NOTE: State hex value 4 cooresponds to ST1, 2 to ST2 and 1 to ST3.

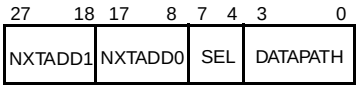
8-21. Errata: A, B, C should be ST1, ST2, ST3.

```
module asm_821 (CLK, RESET, A, B, C, Z) ;

input CLK, RESET, A, B, C ;
output Z ;
reg [1:0] state, next_state;
parameter ST1=2'b00, ST2=2'b01, ST3=2'b10, ST4=2'b11;
reg Z;

//State register
always @(posedge CLK or posedge RESET)
begin
  if (RESET)  //asynchronous RESET active High
    state <= ST1;
  else          //use CLK rising edge
    state <= next_state;
end

//Next state function
always @(A or B or C or state)
begin
      case (state)
            ST1: next_state <= A ? ST2: ST1;
            ST2: next_state <= (B && C) ? ST1: ST3;
            ST3: next_state <= ST1;
            ST4: next_state <= ST1; // Next state ST1 is assigned to unused state ST4.
      endcase
end

//Output function
always @(B or state)
begin
      case (state)
            ST1: Z <= 1'b0;
            ST2: Z <= B ? 1'b1: 1'b0;
            ST3: Z <= 1'b1;
            ST4: Z <= 0'b0;
      endcase
end

endmodule
```



NOTE: State hex value 0 cooresponds to ST1, 1 to ST2 and 2 to ST3.

8-28.



28 bits/word

Total = 1024 words x 28 bits/word = 28,672 bits

8-32.

a) Opcode = 8 bits , b) 16 bits c) 65,536 d) –32768 to +32767

8-39. Errata: Problem 8-39(d): C ← 0 should be C = 0.

	ADDR	NXT	MS	MC	IL	PI	PL	TD	TA	TB	MB	FS	MD	RW	MM	MW
a)	—	17	NXT	NXA	NLI	NLP	NLP	DR	SA	SB	Register	F = A – B	FnUt	WR	—	NW
	—	17	001	0	0	0	0	0	0	0	0	00101	0	1	0	0
DR = 3, SA = 1, SB = 2																
b)	—	—	CNT	—	NLI	NLP	NLP	DR	SA	—	Register	F = lsr A	FnUt	WR	—	NW
	—	—	000	0	0	0	0	0	0	0	0	10100	0	1	0	0
DR = 5, SA = 5																
c)	—	21	BNZ	NXA	NLI	NLP	NLP	—	—	—	—	—	—	—	—	—
	—	21	111	0	0	0	0	0	0	0	0	00000	0	0	0	0
d)	—	—	CNT	—	NLI	NLP	NLP	DR	SA	—	Register	F = A	FnUt	WR	—	NW
	—	—	000	0	0	0	0	0	0	0	0	00000	0	1	0	0
DR = 6, SA = 6																

8-47.

Pipeline Fill	3 cycles
Execution Write-Backs	22 cycles
Final Register Load	1 cycle
TOTAL	26 cycles or $26 \times 5 = 130$ ns