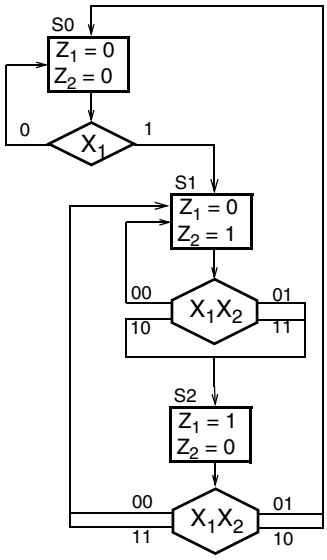
Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 3rd Edition

Chapter 8

8-1.*(Updated 9/21/06)

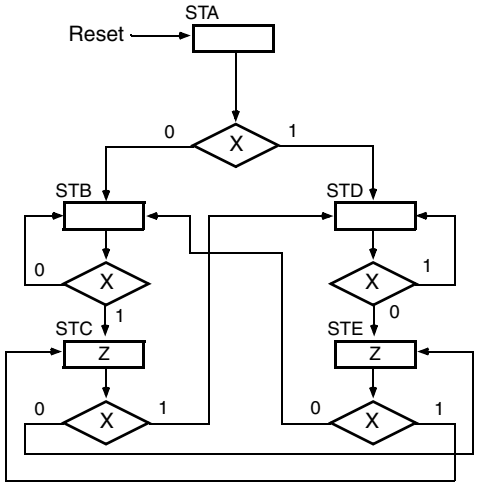


Inputs: X_1X_2
Outputs: Z_1Z_2

8-2.*

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

8-5.*



8-8.*

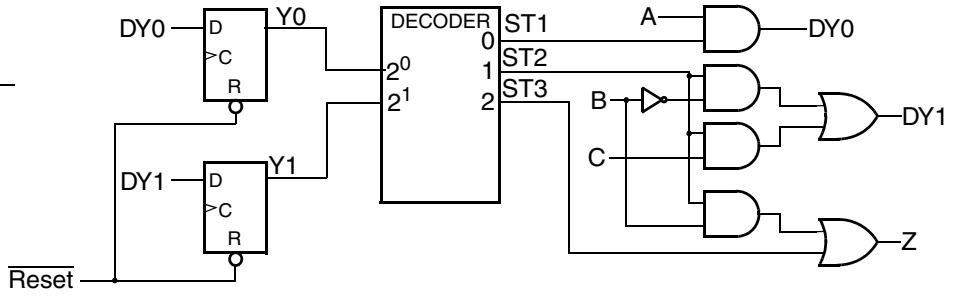
$ST1(t+1) = ST1\cdot\overline{A} + ST2\cdot B\cdot\overline{C} + ST3$, $ST2(t+1) = ST1\cdot A$, $ST3(t+1) = ST2\cdot(\overline{B} + C)$, $Z = ST2\cdot B + ST3$

For the D flip-flops, $D_{STi} = STi(t+1)$ and $STi = Q_{STi}$. Reset initializes the flip-flops: ST1 = 1, ST2 = ST3 = 0.

8-9.*

State Assignment

	Y1	Y0
ST1	0	0
ST2	0	1
ST3	1	0

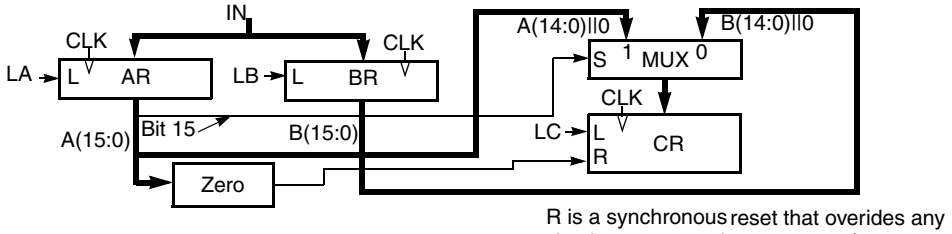


8-11.*

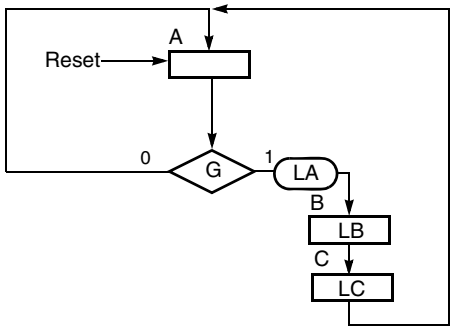
```
    100110    (38)
  × 110101  (× 53)
    100110
   000000
  100110
 000000
100110
100110
11111011110   (2014)
```

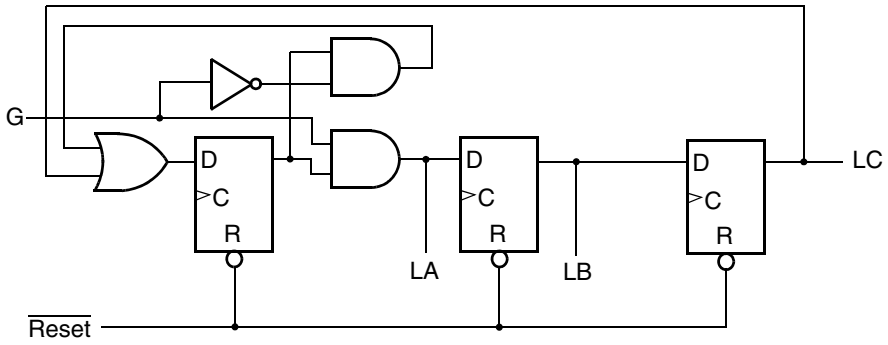
100110	
110101	
000000	Init PP
100110	Add
100110	After Add
0100110	After Shift
00100110	After Shift
100110	Add
10111110	After Add
010111110	After Shift
0010111110	After Shift
100110	Add
1100011110	After Add
01100011110	After Shift
100110	Add
11111011110	After Add
011111011110	After Shift

8-16.*



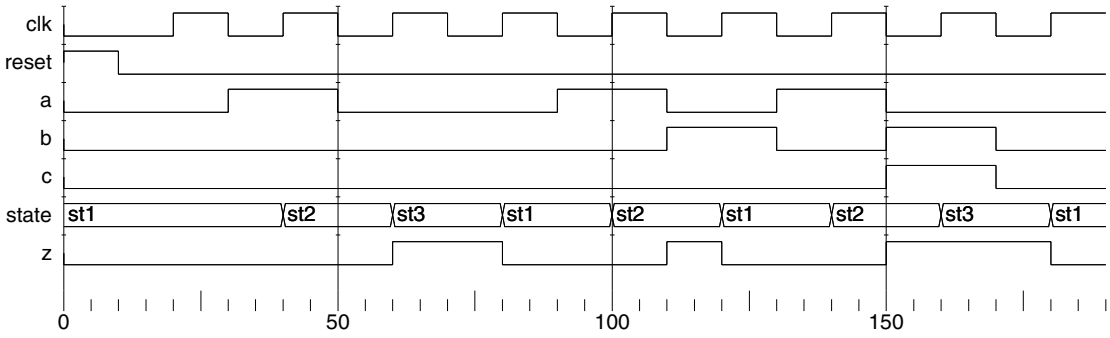
R is a synchronous reset that overides any simultaneous synchronous transfer.





8-19.*

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

entity asm_819 is
      port (
            A, B, C, CLK, RESET: in STD_LOGIC;
            Z: out STD_LOGIC
      );
end asm_819;

architecture asm_819_arch of asm_819 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 = '0')) then
                        next_state <= ST1;
                  else
                        next_state <= ST3;
                  end if;
            when ST3 =>
                  next_state <= ST1;
      end case;
end process;

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_819_arch;
```



8-20.*

```
module asm_820 (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;
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;
          default: next_state <= ST1;
     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;
          default: Z <= 0'b0;
     endcase
end

endmodule
```