King Fahd University Of Petroleum & Minerals
College Of Computer Sciences & Engineering
Computer Engineering Department

COE 405:
Design and Modeling of Digital Systems

Homework # 4

By

987629 Hassan A. Tahhan
987613 Muhammad Amer Araman

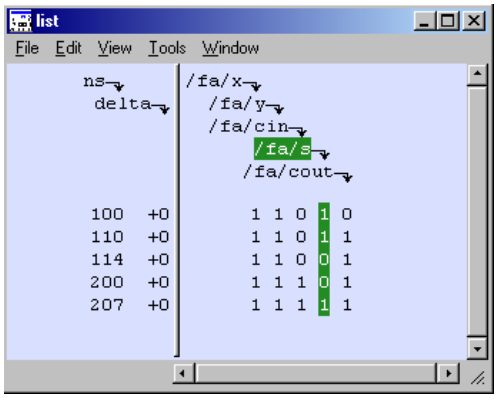
15 Nov, 2002

Q.1. (5.1)

```
entity nor2 is
port (a, b : in bit; o: out bit);
end;
architecture single_delay of nor2 is
begin
o <= a nor b after 4 ns;
end single_delay;

entity xor2 is
port (a, b : in bit; o: out bit);
end;
architecture single_delay of xor2 is
begin
o <= a xor b after 7 ns;
end single_delay;

entity inv is
port (a : in bit; o: out bit);
end;
architecture single_delay of inv is
begin
o <= not a after 4 ns;
end single_delay;

entity nand2 is
port (a, b : in bit; o: out bit);
end;
architecture single_delay of nand2 is
begin
o <= a nand b after 5 ns;
end single_delay;

entity nand3 is
port (a, b, c: in bit; o: out bit);
end;
architecture single_delay of nand3 is
begin
o <= not (a and b and c) after 6 ns;
end single_delay;
```

Q.2. (5.5)

```
entity FA is
port ( x, y, cin : in bit;
       s, cout : out bit);
end FA;

architecture struct of FA is
component n1 port( a, b : in bit; o : out bit);
end component;
component n2 port(a , b: in bit; o : out bit);
end component;
for all: n1 use entity work.nand2 (single_delay);
for all: n2 use entity work.xor2 (single_delay);
signal m1, m2, m3 : bit;
begin
g0: n1 port map (x, y, m2);
g1: n2 port map (x, y, m1);
g2: n1 port map (m1, cin, m3);
g3: n2 port map (m1, cin, s);
g4: n1 port map (m3, m2, cout);
end struct;
```



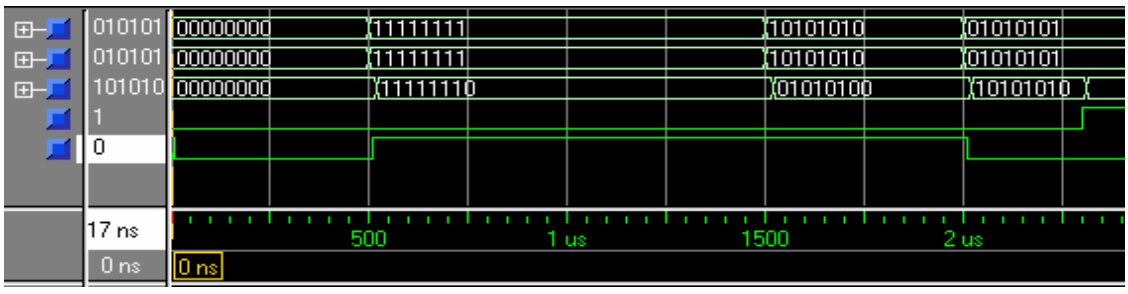
The worst-case delay is 17 ns.

Q.3. (5.11)

```
entity adder8 is
port( a, b: in bit_vector (7 downto 0);
cin : in bit;
s: out bit_vector (7 downto 0);
cout : out bit);
end;

architecture struct of adder8 is
component comp1 port (x, y, cin :in bit; s, cout: out bit);
end component;
for all: comp1 use entity work.FA (struct);
constant n: INTEGER := 8 ;
signal m: bit_vector(6 downto 0);
begin
c_all: for i in 0 to n-1 generate

first: if i =0 generate
least: comp1 port map (a(i), b(i), cin, s(i), m(i));
end generate;

last: if i = n-1 generate
most: comp1 port map (a(i), b(i), m(i-1), s(i),cout);
end generate;

mid: if i > 0 and i < n-1 generate
rest: comp1 port map (a(i), b(i), m(i-1), s(i), m(i));
end generate;

end generate;
end struct;
```

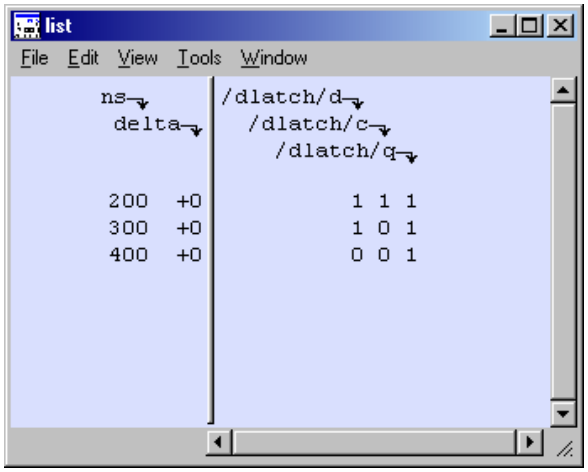
The worst-case delay is 87 ns = 17 + 7 * 10

```
entity adder_test_bench is
end adder_test_bench;

architecture testing of adder_test_bench is
component comp8 port
(a, b: in bit_vector(7 downto 0); cin: in bit;
s: out bit_vector(7 downto 0); cout: out bit);
end component;
for all : comp8 use entity work.adder8(struct);
signal a, b, s: bit_vector(7 downto 0);
signal cin, cout: bit;
begin
   a1: comp8 port map (a, b, cin, s, cout);
   a2: a <= "00000000",
            "11111111" after 0500 ns,
            "10101010" after 1500 ns,
            "01010101" after 2000 ns,
            "10101010" after 2500 ns,
            "01010101" after 3000 ns;
   a3: b <= "00000000",
            "11111111" after 0500 ns,
            "10101010" after 1500 ns,
            "01010101" after 2000 ns,
            "10101010" after 2500 ns,
            "01010101" after 3000 ns;
   a4: cin <= '0',
              '1' after 2300 ns;
end testing;
```



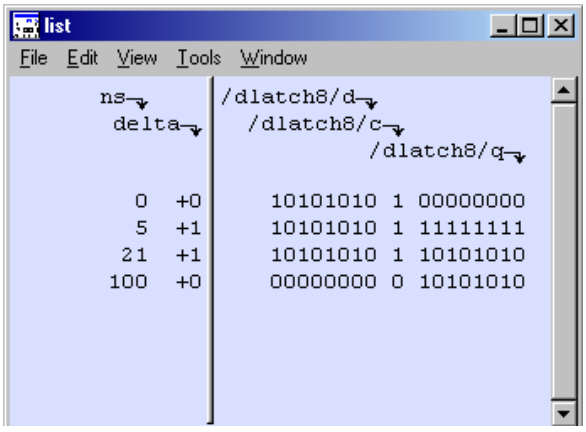
Q.4. (5.13)

```
entity DLatch is
   port ( d, c: in bit;      q: out bit);
end DLatch;

architecture struct of DLatch is
component n1 port( a: in bit; o: out bit);end component;
component n2 port( a, b: in bit; o: out bit);end
component;
for g0: n1 use entity work.nand2(single_delay) port
map(a, a, o);
for g1, g2, g3: n2 use entity work.nand2(single_delay)
port map(a, b, o);
for g4: n2 use entity work.nand3(single_delay) port
map(a, a, b, o);
signal m0, m1, m2, m3, m4 : bit;
begin
   g0: n1 port map (d, m0);
   g1: n2 port map (d, c, m1);
   g2: n2 port map (m0, c, m2);
   g3: n2 port map (m1, m4, m3);
   g4: n2 port map (m2, m3, m4);
   q <= m3;
end struct;
```



Q.5. (5.14)

```
entity DLatch8 is
port( d: in bit_vector (7 downto 0);
c: in bit;
q: out bit_vector (7 downto 0));
end;

architecture struct of DLatch8 is
component comp1 port (d, c:in bit; q: out bit); end
component;
for all: comp1 use entity work.DLatch(struct);
begin

c_all: for i in 0 to 7 generate
rest: comp1 port map (d(i), c, q(i));
end generate;

end struct;
```



Q.6. (5.17)

```
entity function_f is port (a, b, c, d, e: in bit ; f:out bit);
end function_f;

architecture configurable of function_f is
component n1 port (w: in bit; z: out bit); end component;
component n2 port (w, x: in bit; z: out bit); end component;
component n3 port (w, x, y: in bit; z: out bit); end component;

for all: n1 use entity work.nand3(single_delay) port map
(w,w,w,z);
for all: n2 use entity work.nand3(single_delay) port map
(w,w,x,z);
for all: n3 use entity work.nand3(single_delay) port map
(w,x,y,z);

signal i1, i2, i3, i4: bit;
begin
   g0: n1 port map (d, i1);
   g1: n1 port map (e, i2);
   g2: n2 port map (i1, c, i3);
   g3: n2 port map (a, b, i4);
   g4: n3 port map (i2, i3, i4, f);
end configurable;
```

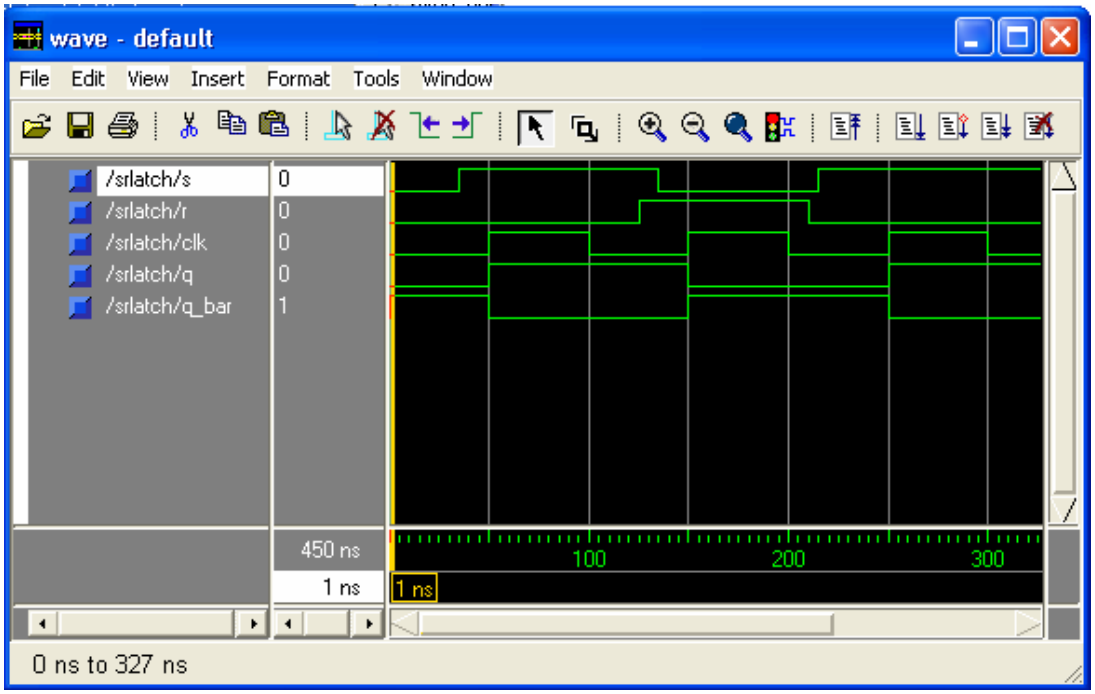
Q.7. (5.20)

```
entity SRlatch is
port (s, r, clk: in bit; q, q_bar :buffer bit);
end;
architecture behave of SRlatch is
begin
process(clk)
begin
if(clk = '1' and clk'event) then
        if (s = '0' and r = '0') then
                q <= q;
        else if(s = '1' and r = '0') then
                q <= '1' ;
            else if(s = '0' and r='1') then
                        q <= '0';
                  end if;
            end if;
        end if;
end if;
end process;
process(q)
begin
q_bar <= not q;
end process;
end behave;
```

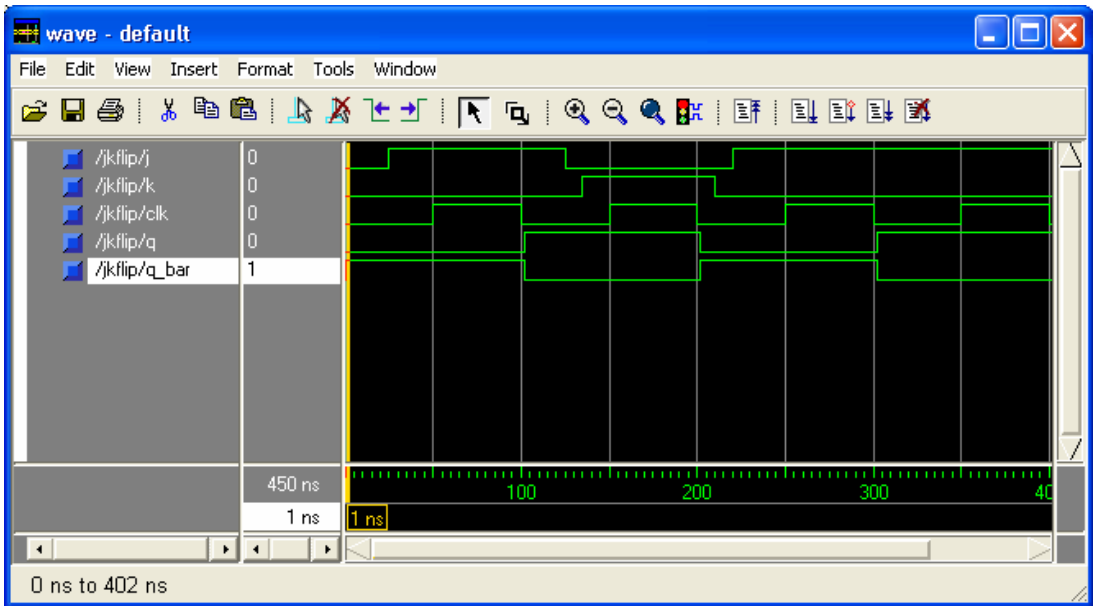
```
entity jkflip is
port (j, k, clk : in bit; q, q_bar : buffer bit);
end;
architecture sr_struct of jkflip is
component SRlatch port (s, r, clk: in bit; q, q_bar : buffer bit); end component;
component inv port (i : in bit; o: out bit); end component;
component and2 port (i1, i2 :in bit; o: out bit); end component;
signal m1, m2, m3, m4, m5: bit;
begin
jand: and2 port map( j, q_bar, m1);
kand: and2 port map( k, q, m2);
msr: SRlatch port map (m1, m2, clk, m3, m4);
minv: inv port map (clk, m5);
ssr: SRlatch port map (m3, m4, m5, q, q_bar);
end sr_struct;
```

```
entity counter1 is
port (en, clk: in bit; q: buffer bit; co: out bit);
end;
architecture struct of counter1 is
component jkflip port (j,k, clk: in bit; q,q_bar: buffer bit); end component;
component and2 port(i1,i2: in bit; o: out bit); end component;
signal q_bar: bit;
begin
jk: jkflip port map (en, en, clk, q, q_bar);
coand: and2 port map (q, en, co);
end struct;
```

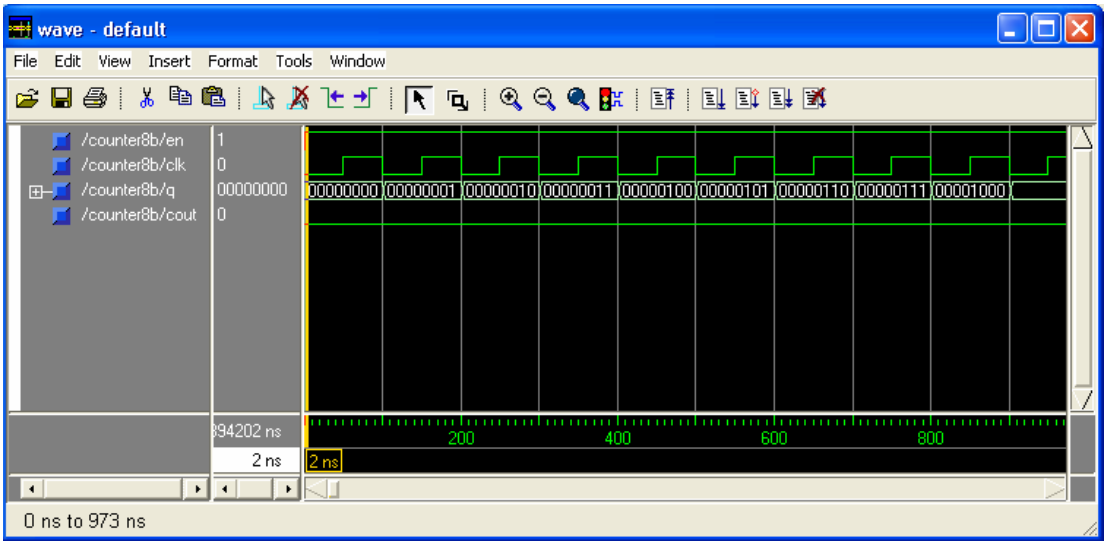
```
entity counter8b is
port (en, clk: in bit; q : out bit_vector(7 downto 0); cout: out bit);
end;

architecture struct of counter8b is
component counter1 port (en, clk: in bit; q: buffer bit; co: out bit); end
component;
signal m : bit_vector(6 downto 0);
constant n: INTEGER := 8;
begin

c_all: for i in 0 to 7 generate

one: if i = 0 generate
first: counter1 port map (en, clk, q(i), m(i));
end generate;

last: if i = n-1 generate
most: counter1 port map(m(i-1), clk, q(i), cout);
end generate;

mid: if i<n-1 and i>0 generate
rest: counter1 port map(m(i-1), clk, q(i), m(i));
end generate;

end generate;
end struct;
```



SR latch simulation



JK flip flop Simulation



8-bit counter simulation