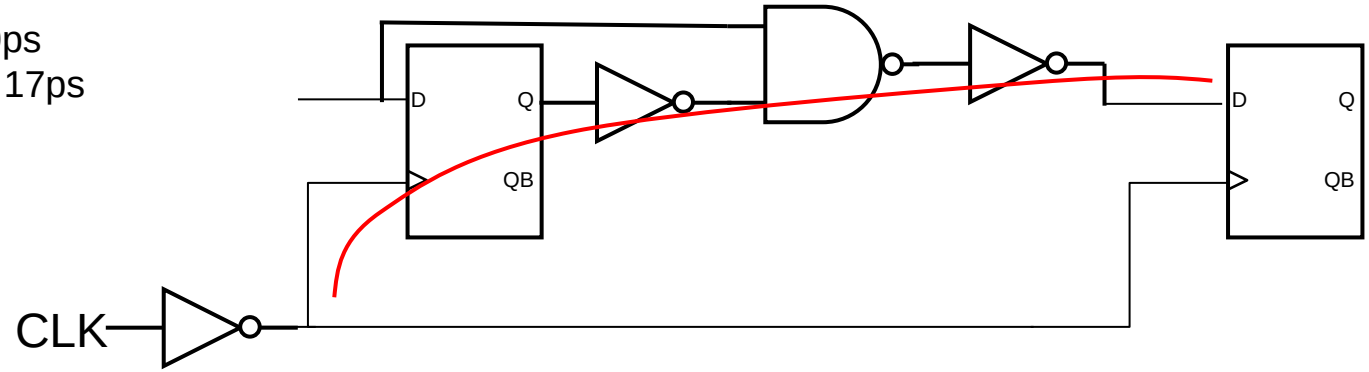
EE 3921

Dr. Johnson

Homework 1

1 – Calculate the fastest possible clock frequency 20pts

t_{PD} INV = 10ps
t_{PD} NAND = 17ps
t_{CQ} = 22ps
t_{setup} = 5ps
t_{hold} = 2ps



critical path = clk to valid D input on second FF

$= t_{CQ} + T_{inv} + T_{nand} + T_{inv} + T_{SU}$

= 22ps + 10ps + 17ps + 10ps + 5ps = 64ps

→ F max = 1/64ps = 15.6GHz

2 – Word Match, Identify the best matching number on the left for each item on the right. (no duplicates) 10pts

1	Analysis and Elaboration	6	View Schematic
2	Analyze Current File	9	State machine schematic/logic
3	Analysis and Synthesis	4	Creates the physical design
4	Start Fitter	3	Creates logic
5	Pin Planner	10	Allows access to the physical design
6	RTL Viewer	2	Checks for syntax errors
7	Signal Tap II	7	Debug Tool
8	Model Sim	8	Simulates RTL
9	State Machine Viewer	1	Creates RTL
10	Chip Planner	5	Assign/view pins

3) Given the VHDL code below and the indicated inputs, provide the expected signal values

30pts

a=1, b=0, c=0, d=1
e=0110, f=1001
g=0111, h=1110

i= 1

j= 1

p= 1100 0111

q= 1001 (-7)

r= 0111 (7)

s= 0000 (0)

u= 1111 (-1)

v= 0000 (0)

aa= 1001 (9)

bb= 0111 (7)

cc= 0000 (0)

ee= 0001 (1)

ff= 0010 (2)

gg= 1100 1111(-49)

hh= 0011 1111(63)

0 For MOD only!
-7/7 → -1
leaves remainder = 0

```
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

-- IGNORING NAMING CONVENTION --

entity arithHW is
generic( NNN:  INTEGER  := 4);
   port  (  a,b,c,d:     in STD_LOGIC;
            e,f,g,h:     in STD_LOGIC_VECTOR(NNN-1 downto 0);
            i,j,k,l:     out STD_LOGIC;
            m,n:         out STD_LOGIC_VECTOR(NNN-1 downto 0);
            o,p,z:       out STD_LOGIC_VECTOR(2*NNN-1 downto 0);

            foo:         out std_logic_vector(65 downto 0)

         );
end entity;

Architecture behavioral of arithHW is

   signal q,r,s,t,u,v:        SIGNED((NNN-1) downto 0);
   signal aa,bb,cc,dd,ee,ff:  UNSIGNED((NNN-1) downto 0);
   signal gg:                 SIGNED((2*NNN-1) downto 0);
   signal hh:                 UNSIGNED((2*NNN-1) downto 0);

   begin
      q <= SIGNED(f);
      r <= SIGNED(g);
      aa <= UNSIGNED(f);
      bb <= UNSIGNED(g);

      i <= ((a or b) and c) xor d;
      j <= a and (e(1) or e(2));

      s <= q + r;
      gg <= q * r;
      u <= q / r;
      v <= q mod r;

      cc <= aa + bb;
      hh <= aa * bb;
      ee <= aa / bb;
      ff <= aa mod bb;

      p <= ('1' & h(2) & "00" & g(3 downto 1) & '1');

end architecture;
```

don't change anything
define how to interpret

only keeps the whole part
keeps the remainder with sign of mod (r)

4 – Write your own behavioral VHDL code for a JK Flip-Flop
Provide code and a simulation 40pts

```
--------------------------------
--
-- JK_ff.vhdl
--
-- created 2/1/2018
-- tj
--
-- rev 0
--------------------------------
--
-- JK flip-flop
--
--------------------------------
--
-- Inputs: clk, rstb, j, k
-- Outputs: q, qb
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity JK_ff is
    port (
          i_clk :  in std_logic;
          i_rstb : in std_logic;
          i_J :    in std_logic;
          i_K :    in std_logic;

          o_Q :    out std_logic;
          o_Qb :   out std_logic
    );
end entity;
```

```
architecture behavioral of JK_ff is
begin
    process(i_clk, i_rstb)
    begin
        if (i_rstb = '0') then
            o_Q  <= '0';
            o_Qb <= '1';
        elsif (rising_edge(i_clk)) then
            if ((i_J = '0') and (i_K = '0')) then
                o_Q  <= o_Q;
                o_Qb <= o_Qb;
            elsif ((i_J = '0') and (i_K = '1')) then
                o_Q  <= '0';
                o_Qb <= '1';
            elsif ((i_J = '1') and (i_K = '0')) then
                o_Q  <= '1';
                o_Qb <= '0';
            elsif ((i_J = '1') and (i_K = '1')) then
                o_Q  <= not o_Q;
                o_Qb <= not o_Qb;
            else
                o_Q  <= o_Q;
                o_Qb <= o_Qb;
            end if;
        end if;
    end process;
end behavioral;
```

Note: This design has the same delay for Q and Qb
if you just create Q and then say ‘Qb <= not Q’ they
will have different delays