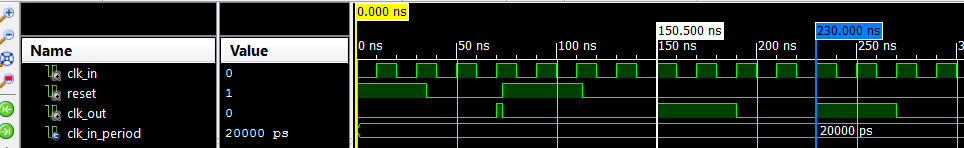
**ECE102 Lecture 22 simulation results**

1. **Asynchronous reset**



----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity clk\_div\_top is

Port (

reset : in std\_logic;

clk\_in : in STD\_LOGIC;

clk\_out : out STD\_LOGIC);

end clk\_div\_top;

architecture Behavioral of clk\_div\_top is

signal clk\_temp : std\_logic;

---- internal signal: clk\_temp

----Question: why do you need an internal signal?

signal count : integer range 0 to 15 :=0;

----32-bit count (default).

-----Question: Why do you specify the range?

begin

process(clk\_in,reset,count)

begin

---- case #1 : Asychronous reset

if (reset = '1') then---- reset condition is HIGH

clk\_temp <= '0';

count <= 0; ---reset the counter

----rising edge ====== active edge of clk\_in

elsif (clk\_in = '1' and clk\_in'event) then

count <= count + 1; --- update the counter

if (count = 1) then

----Question: the conditional checking is based on past value or current value of count?

----Question: what's the frequency of clk\_temp?

---- count = N ==> fclk/2\*(N+1)

clk\_temp <= not clk\_temp;

count <= 0; ---- reset the count to zero every half cycle of clk\_temp

end if;

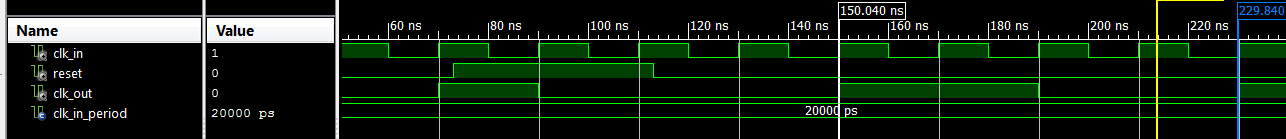
end if;

end process;

clk\_out <= clk\_temp;

end Behavioral;

1. **Synchronous reset**



----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity clk\_div\_top is

Port (

reset : in std\_logic;

clk\_in : in STD\_LOGIC;

clk\_out : out STD\_LOGIC);

end clk\_div\_top;

architecture Behavioral of clk\_div\_top is

signal clk\_temp : std\_logic;

---- internal signal: clk\_temp

----Question: why do you need an internal signal?

signal count : integer range 0 to 15 :=0;

----32-bit count (default).

-----Question: Why do you specify the range?

begin

process(clk\_in,reset,count)

begin

----- case #2 : syncrhonous reset

----rising edge ====== active edge of clk\_in

if (clk\_in = '1' and clk\_in'event) then

if (reset = '1') then

clk\_temp <= '0';

count <= 0;

else

count <= count + 1;

if (count = 1) then -----

clk\_temp <= not clk\_temp;

count <= 0;----reset the counter to 0

end if;

end if;

end if;

end process;

clk\_out <= clk\_temp;

end Behavioral;