Clock Dividers

Last updated 5/15/20

Counters

These slides review the design for several types of clock dividers

Upon completion: You should be able to design an efficient clock divider

Clock Dividers

- Clock Divider

  - Normally we do not use a clock divider
    Effectively a gated clock

  BUT

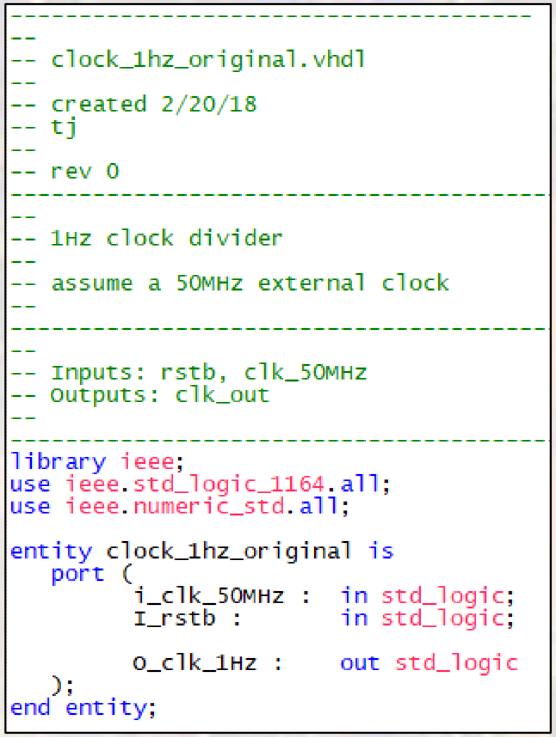
  - In order to see operation of designs on the DE10 we need to slow down the system clock (50MHz)

  - To operate at 1 HZ → divide by 50,000,000

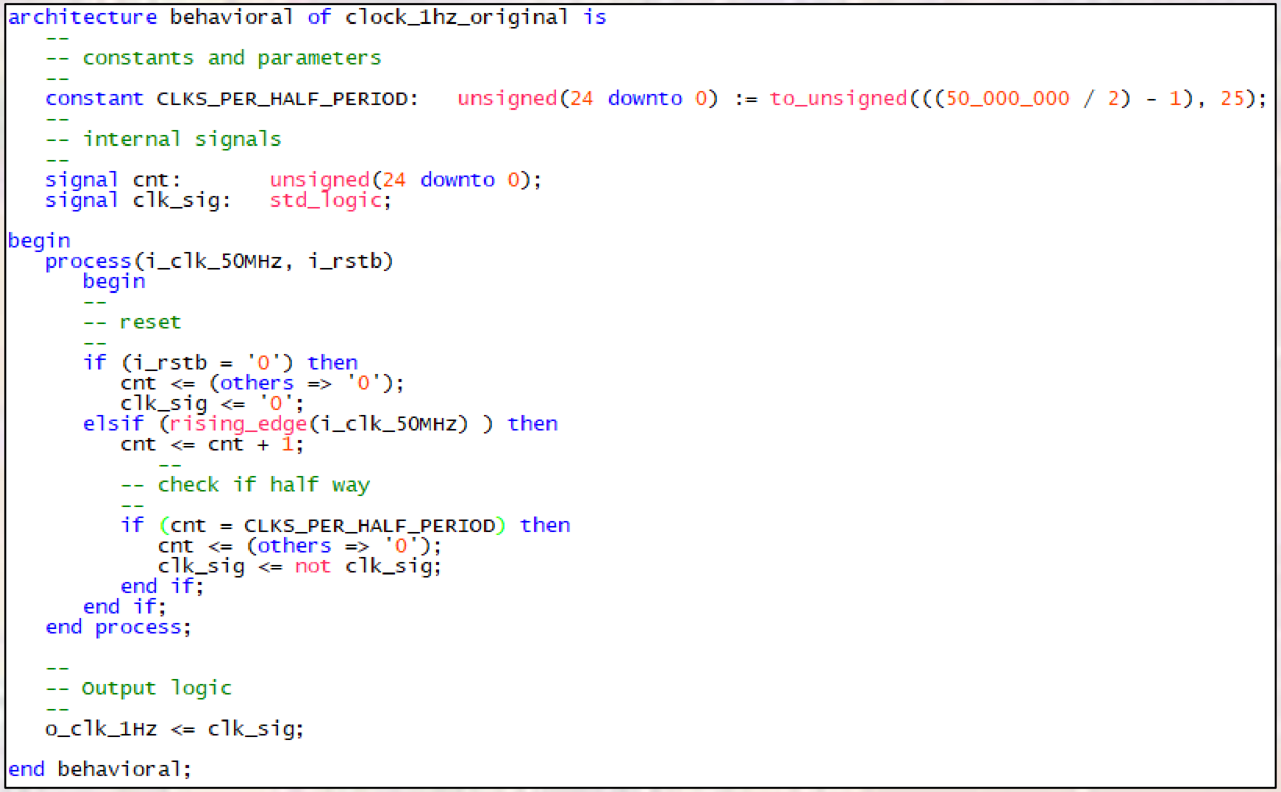
* Note – we should really do this another way – but with small designs and all of our design running on the same clock we can get by

Clock Dividers

- Clock Divider

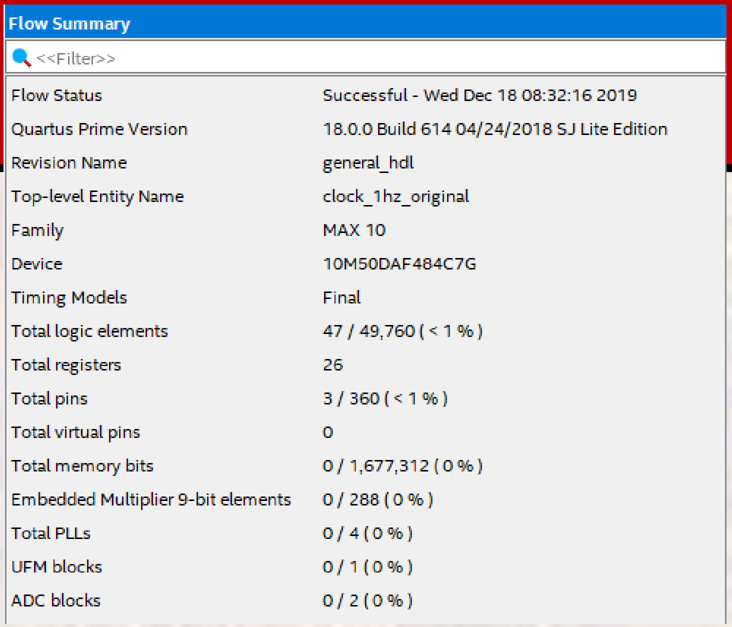


```
---------------------------------------
--
-- clock_1hz_original.vhdl
--
-- created 2/20/18
-- tj
--
-- rev 0
----------------------------------------
--
-- 1Hz clock divider
--
-- assume a 50MHz external clock
--
----------------------------------------
--
-- Inputs: rstb, clk_50MHz
-- Outputs: clk_out
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_1hz_original is
   port (
         i_clk_50MHz :   in std_logic;
         I_rstb :        in std_logic;

         o_clk_1Hz :     out std_logic
   );
end entity;
```



```
architecture behavioral of clock_1hz_original is
   --
   -- constants and parameters
   --
   constant CLKS_PER_HALF_PERIOD:   unsigned(24 downto 0) := to_unsigned(((50_000_000 / 2) - 1), 25);
   --
   -- internal signals
   --
   signal cnt:       unsigned(24 downto 0);
   signal clk_sig:   std_logic;

begin
   process(i_clk_50MHz, i_rstb)
      begin
      --
      -- reset
      --
      if (i_rstb = '0') then
         cnt <= (others => '0');
         clk_sig <= '0';
      elsif (rising_edge(i_clk_50MHz) ) then
         cnt <= cnt + 1;
            --
         -- check if half way
         --
         if (cnt = CLKS_PER_HALF_PERIOD) then
            cnt <= (others => '0');
            clk_sig <= not clk_sig;
         end if;
      end if;
   end process;

   --
   -- Output logic
   --
   o_clk_1Hz <= clk_sig;

end behavioral;
```

Clock Dividers

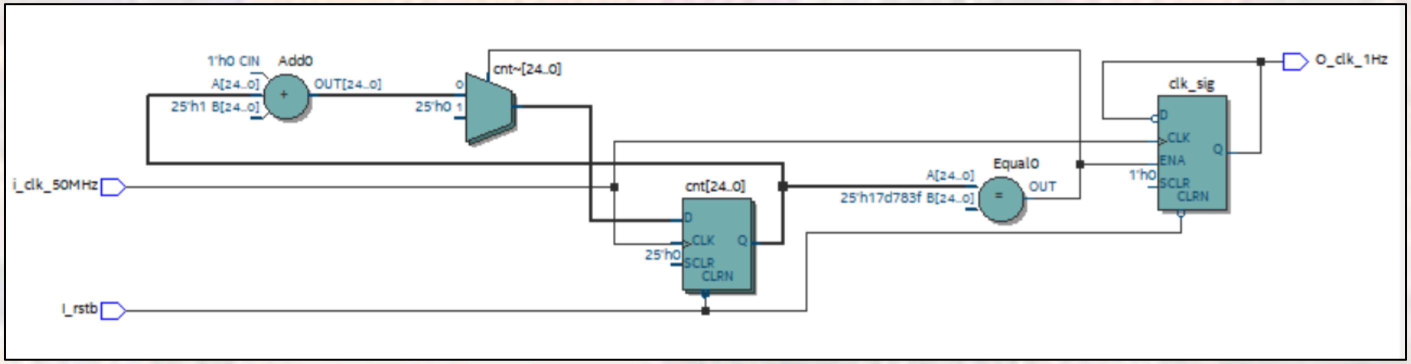


Flow Summary

<<Filter>>

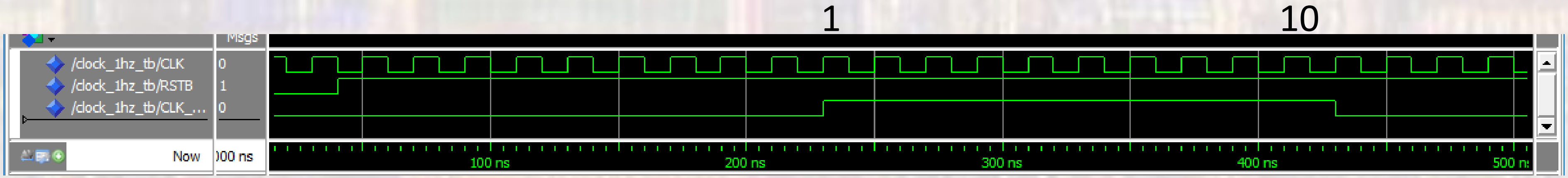
Flow Status	Successful - Wed Dec 18 08:32:16 2019
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	general_hdl
Top-level Entity Name	clock_1hz_original
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	47 / 49,760 (< 1 %)
Total registers	26
Total pins	3 / 360 (< 1 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

- Clock Divider



Clock Dividers

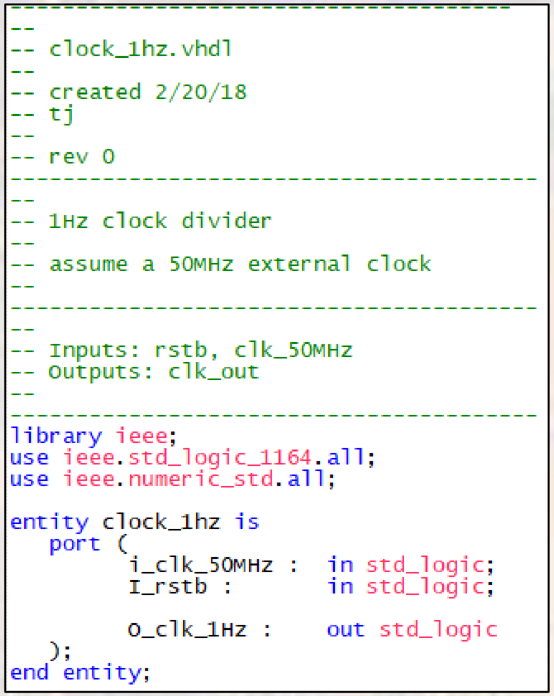
- Clock Divider
  - original
  - modified to CLOCKS_PER_HALF_PERIOD = (10 – 1)



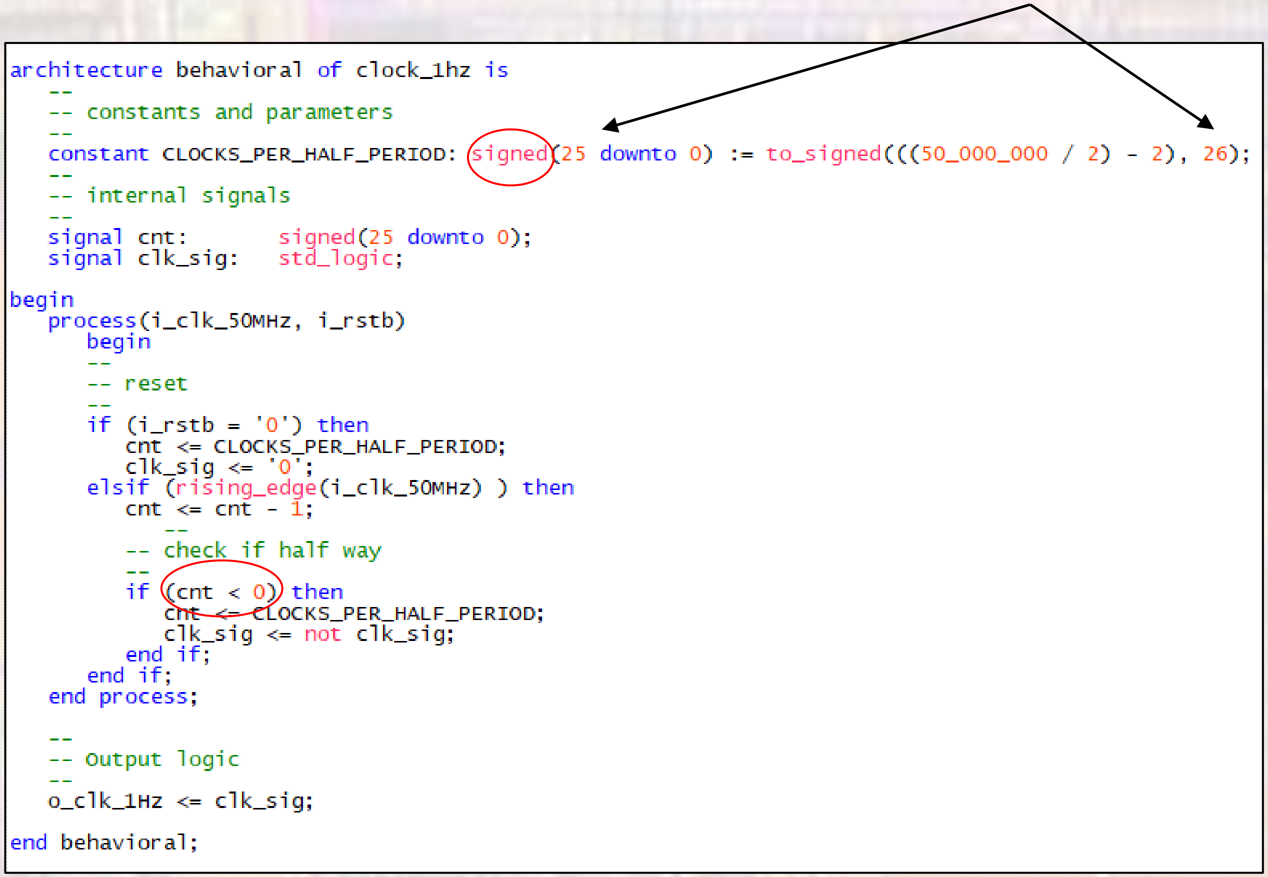
Clock Dividers

- Clock Divider – enhanced (count down)

why is the bit width larger?



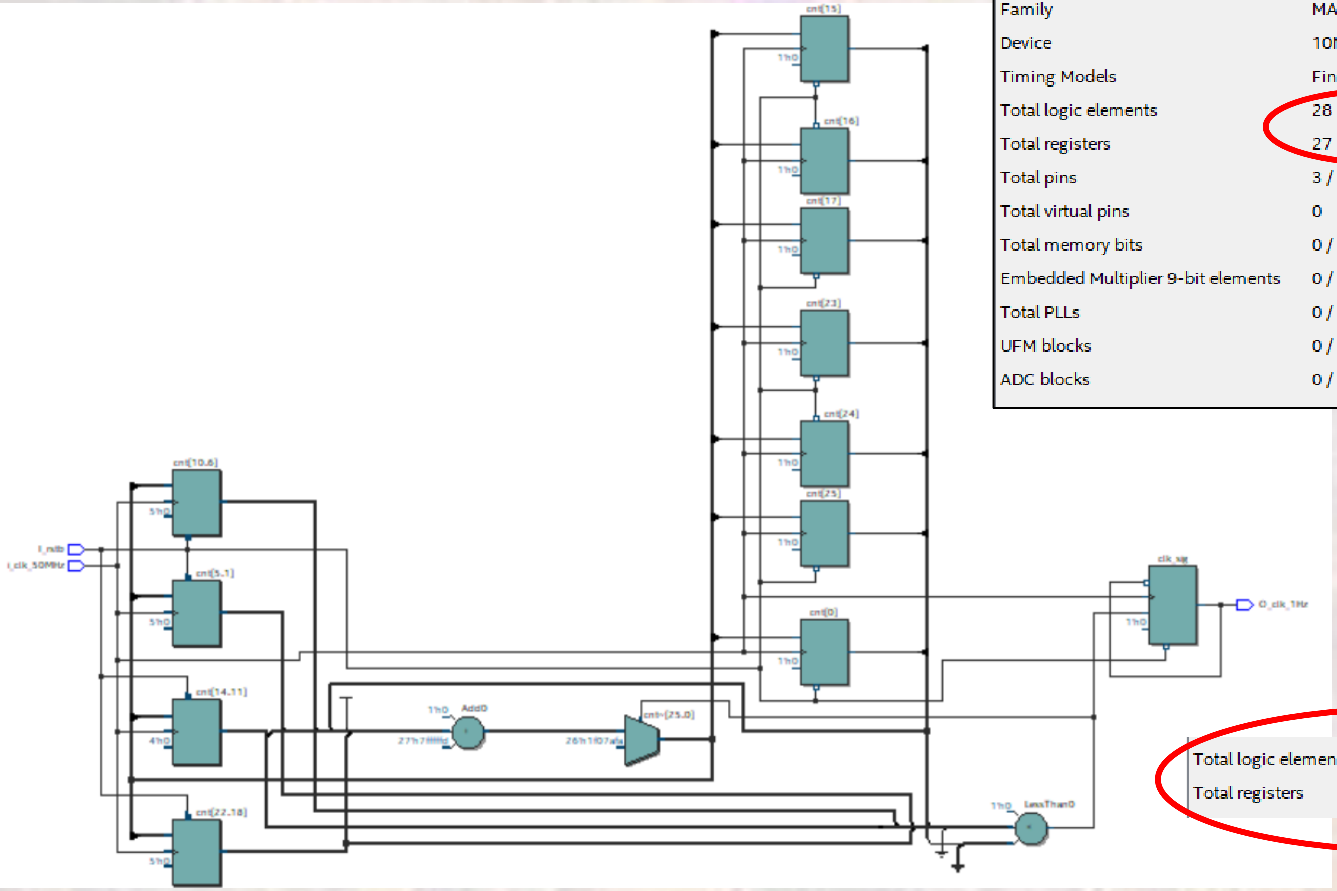
```
----------------------------------------
--
-- clock_1hz.vhdl
--
-- created 2/20/18
-- tj
--
-- rev 0
----------------------------------------
--
-- 1Hz clock divider
--
-- assume a 50MHz external clock
--
----------------------------------------
--
-- Inputs: rstb, clk_50MHz
-- Outputs: clk_out
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_1hz is
   port (
         i_clk_50MHz :   in std_logic;
         I_rstb :        in std_logic;

         o_clk_1Hz :     out std_logic
   );
end entity;
```



```
architecture behavioral of clock_1hz is
   --
   -- constants and parameters
   --
   constant CLOCKS_PER_HALF_PERIOD: signed(25 downto 0) := to_signed(((50_000_000 / 2) - 2), 26);
   --
   -- internal signals
   --
   signal cnt:       signed(25 downto 0);
   signal clk_sig:   std_logic;

begin
   process(i_clk_50MHz, i_rstb)
      begin
      --
      -- reset
      --
      if (i_rstb = '0') then
         cnt <= CLOCKS_PER_HALF_PERIOD;
         clk_sig <= '0';
      elsif (rising_edge(i_clk_50MHz) ) then
         cnt <= cnt - 1;

         --
         -- check if half way
         --
         if (cnt < 0) then
            cnt <= CLOCKS_PER_HALF_PERIOD;
            clk_sig <= not clk_sig;
         end if;
      end if;
   end process;

   --
   -- Output logic
   --
   o_clk_1Hz <= clk_sig;

end behavioral;
```

Clock Dividers

- Clock Divider - enhanced



Flow Summary

<<Filter>>

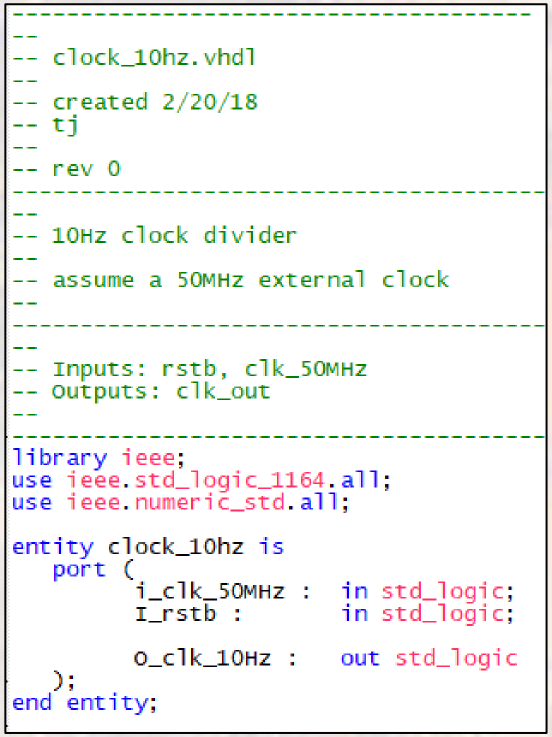
Flow Status	Successful - Wed Dec 18 08:43:43 2019
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	general_hdl
Top-level Entity Name	clock_1hz
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	28 / 49,760 (< 1 %)
Total registers	27
Total pins	3 / 360 (< 1 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

original

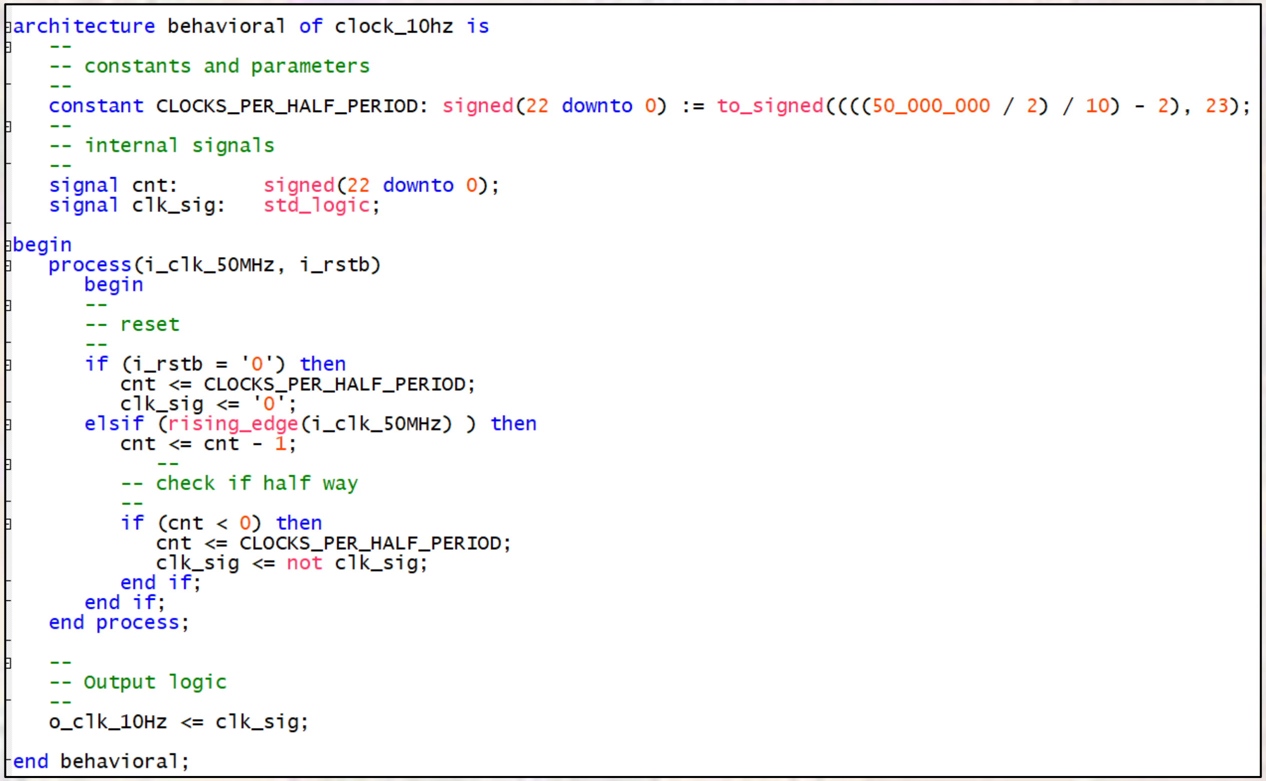
Total logic elements	47 / 49,760 (< 1 %)
Total registers	26

Clock Dividers

- Clock Divider – enhanced → 10Hz



```
---------------------------------------
--
-- clock_10hz.vhdl
--
-- created 2/20/18
-- tj
--
-- rev 0
---------------------------------------
--
-- 10Hz clock divider
--
-- assume a 50MHz external clock
--
---------------------------------------
--
-- Inputs: rstb, clk_50MHz
-- Outputs: clk_out
--
---------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_10hz is
   port (
         i_clk_50MHz :  in std_logic;
         I_rstb :       in std_logic;

         o_clk_10Hz :   out std_logic
   );
end entity;
```



```
architecture behavioral of clock_10hz is
   --
   -- constants and parameters
   --
   constant CLOCKS_PER_HALF_PERIOD: signed(22 downto 0) := to_signed((((50_000_000 / 2) / 10) - 2), 23);
   --
   -- internal signals
   --
   signal cnt:       signed(22 downto 0);
   signal clk_sig:   std_logic;

begin
   process(i_clk_50MHz, i_rstb)
      begin
      --
      -- reset
      --
      if (i_rstb = '0') then
         cnt <= CLOCKS_PER_HALF_PERIOD;
         clk_sig <= '0';
      elsif (rising_edge(i_clk_50MHz) ) then
         cnt <= cnt - 1;
            --
         -- check if half way
         --
         if (cnt < 0) then
            cnt <= CLOCKS_PER_HALF_PERIOD;
            clk_sig <= not clk_sig;
         end if;
      end if;
   end process;

   --
   -- Output logic
   --
   o_clk_10Hz <= clk_sig;

end behavioral;
```