Clock Dividers

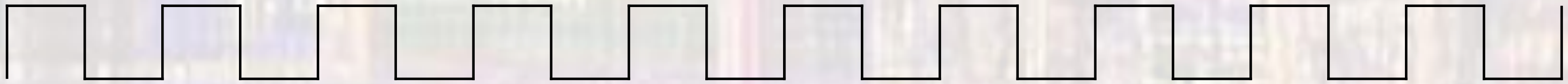
Last updated 2/8/24

Clock Dividers

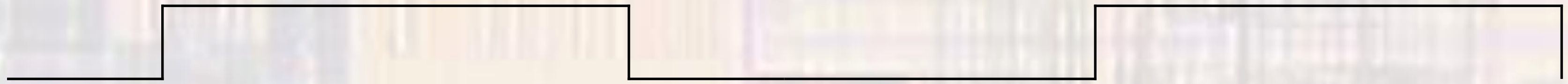
- Clock division
  - Normally a clock divider should never be used for internal circuits
    - Effectively a gated clock
    - Leads to timing issues
    - Use a PLL instead (we will cover this later)
  - A clock divider can be used for an external clock
    - Only if the external/internal relative clock edge is not critical
  - In order to see operation of designs on the DE10 we need to slow down the system clock (50MHz)
    - Run the entire system off of the slow clock
      - Removes timing issues
    - Typically dividing down to 1-30Hz
      - Approximately as fast as the human eye can see

Clock Dividers

- Clock Division Concept
  - Reference Clock (Fclk)



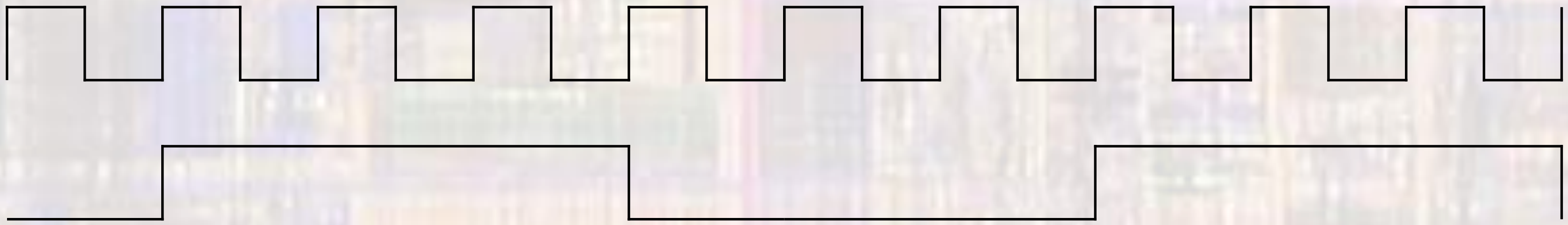
  - Divided Clock (Fclk/n)
    - n = 6



  - Need a way to cause the divided clock to go high and low

Clock Dividers

- Clock Division Concept



- Each ½ clock period requires n/2 reference clock cycles
  - Toggle the divided clock after counting to n/2
- Odd divisions not allowed
  - Much more complex
  - Asymmetrical – no longer a square wave

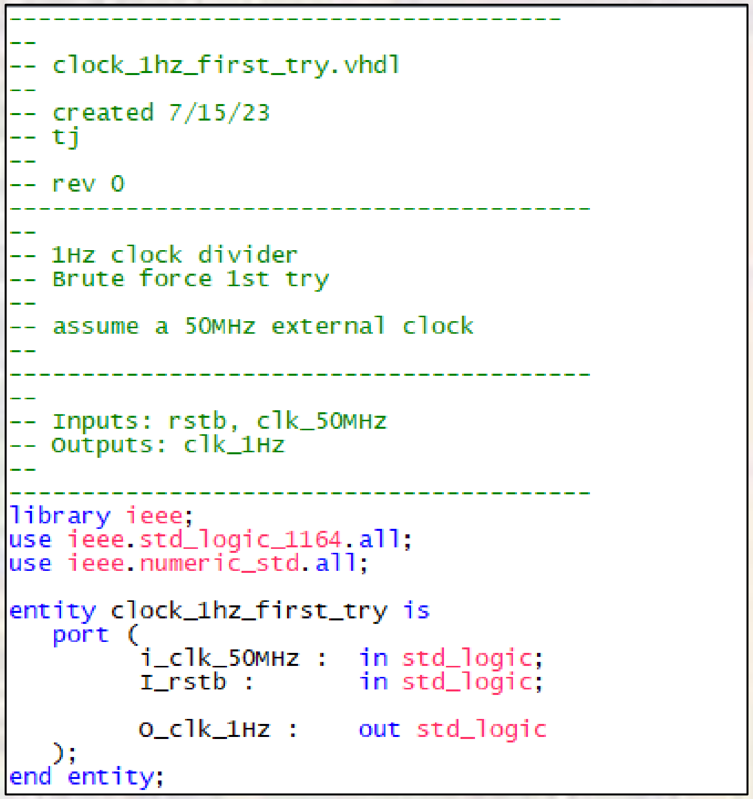
Clock Dividers

- Clock Division Concept
  - 50MHz reference clock
    - 1Hz clock → divide by 50,000,000 → count 25,000,000 clocks
      - Toggle output
    - 10Hz clock → divide by 5,000,000 → count 2,500,000 clocks
      - Toggle output
  - Note: there may be overhead associated with the counter that modifies the count value

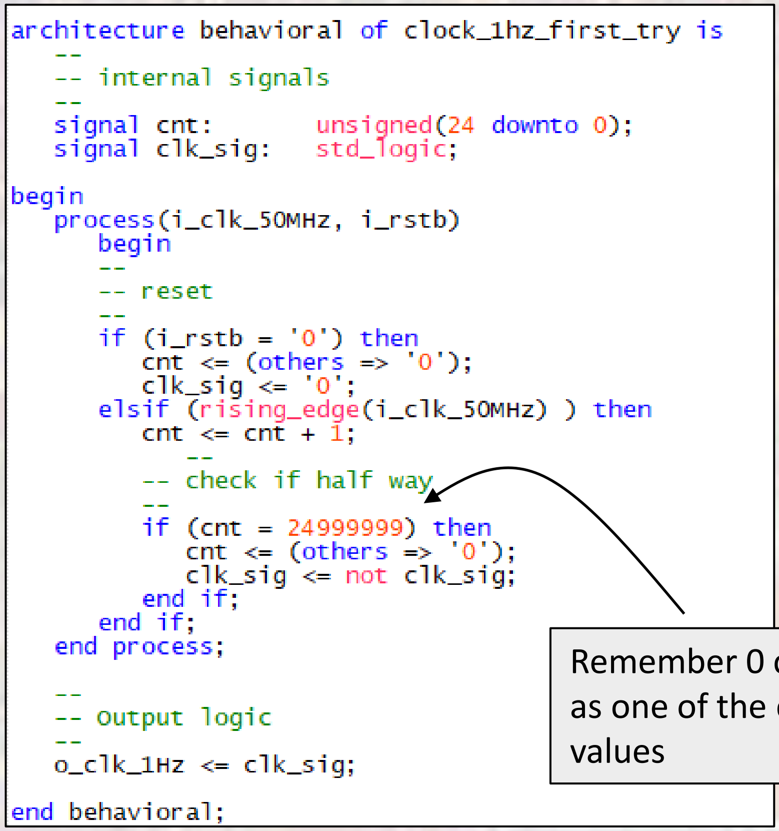
Clock Dividers

- Clock Divider – First try
  - 1Hz example

Counter sizing:
  Count 25,000,000 clocks → requires $\log_2 25{,}000{,}000$ = 25 bits



```
----------------------------------------
--
-- clock_1hz_first_try.vhdl
--
-- created 7/15/23
-- tj
--
-- rev 0
------------------------------------------
--
-- 1Hz clock divider
-- Brute force 1st try
--
-- assume a 50MHz external clock
--
------------------------------------------
--
-- Inputs: rstb, clk_50MHz
-- Outputs: clk_1Hz
--
------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_1hz_first_try is
   port (
         i_clk_50MHz :  in std_logic;
         I_rstb :       in std_logic;

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



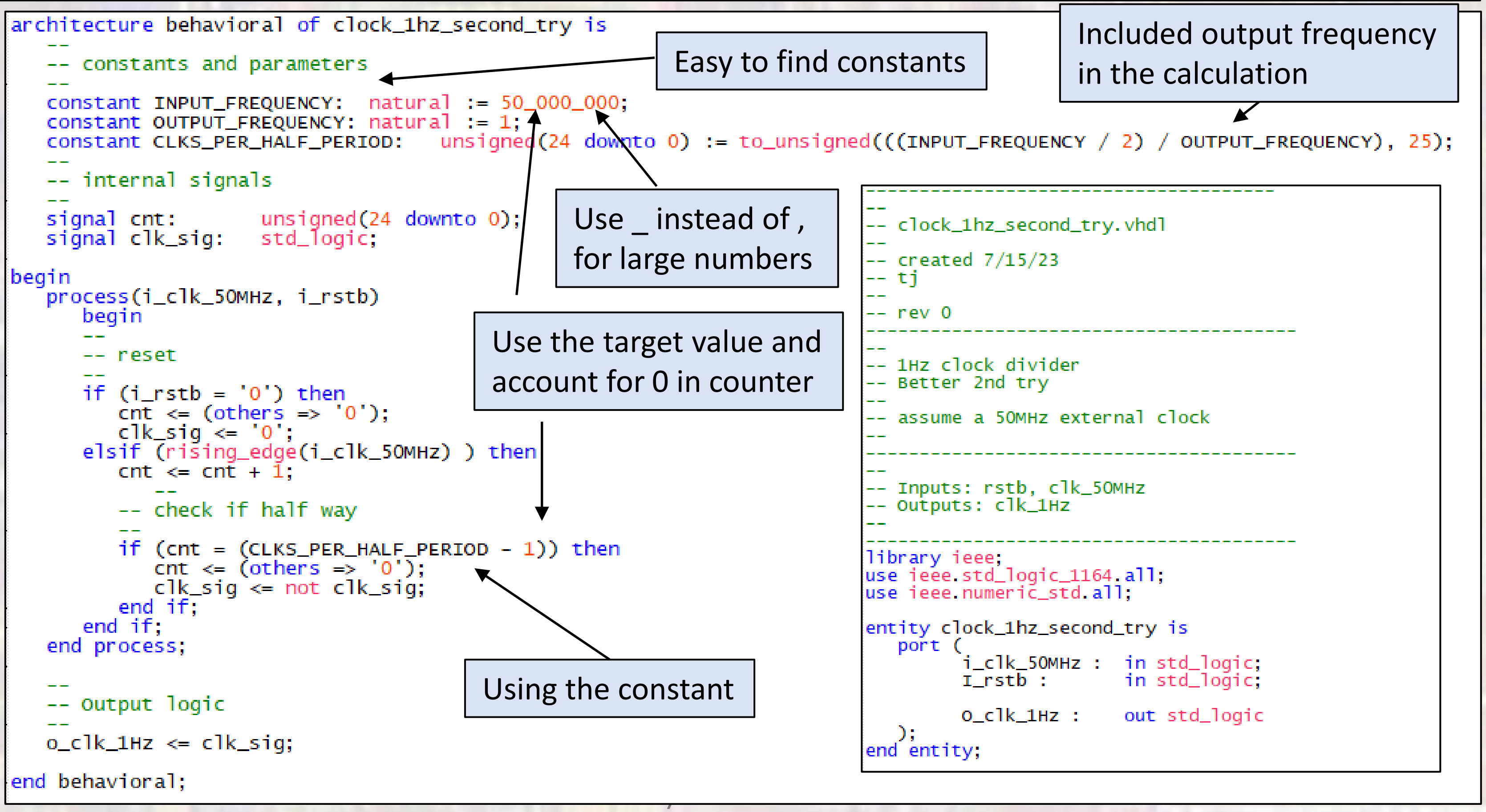
```
architecture behavioral of clock_1hz_first_try is
   --
   -- internal signals
   --
   signal cnt:       unsigned(24 downto 0);
   signal clk_sig:   std_logic;

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 = 24999999) then
            cnt <= (others => '0');
            clk_sig <= not clk_sig;
         end if;
      end if;
   end process;


   --
   -- output logic
   --
   o_clk_1Hz <= clk_sig;

end behavioral;
```

Remember 0 counts as one of the count values

Clock Dividers

- Clock Divider – Second try
  - 1Hz example

Clean up the code to make it easier to modify for different input/output frequencies



```vhdl
architecture behavioral of clock_1hz_second_try is
   --
   -- constants and parameters
   --
   constant INPUT_FREQUENCY:  natural := 50_000_000;
   constant OUTPUT_FREQUENCY: natural := 1;
   constant CLKS_PER_HALF_PERIOD:   unsigned(24 downto 0) := to_unsigned(((INPUT_FREQUENCY / 2) / OUTPUT_FREQUENCY), 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 - 1)) then
            cnt <= (others => '0');
            clk_sig <= not clk_sig;
         end if;
      end if;
   end process;

   --
   -- output logic
   --
   o_clk_1Hz <= clk_sig;

end behavioral;
```

```vhdl
-----------------------------------------
--
-- clock_1hz_second_try.vhdl
--
-- created 7/15/23
-- tj
--
-- rev 0
----------------------------------------
--
-- 1Hz clock divider
-- Better 2nd try
--
-- assume a 50MHz external clock
--
----------------------------------------
--
-- Inputs: rstb, clk_50MHz
-- Outputs: clk_1Hz
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_1hz_second_try is
   port (
         i_clk_50MHz :   in std_logic;
         I_rstb :        in std_logic;

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

Easy to find constants

Included output frequency in the calculation

Use _ instead of , for large numbers

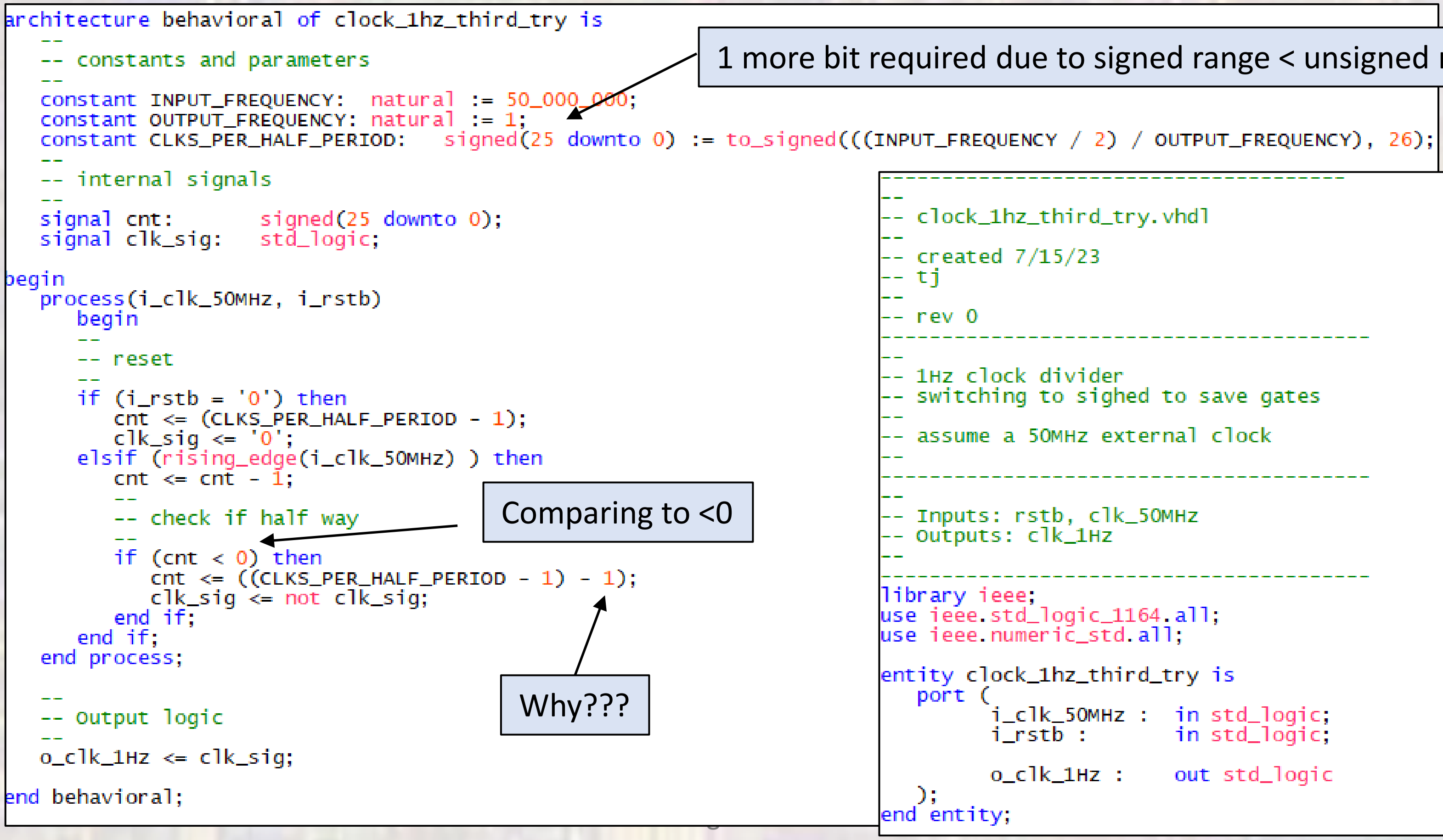
Use the target value and account for 0 in counter

Using the constant

Clock Dividers

- Clock Divider – Third try
  - 1Hz example

Use 'best practices' and compare to <0 or >=0

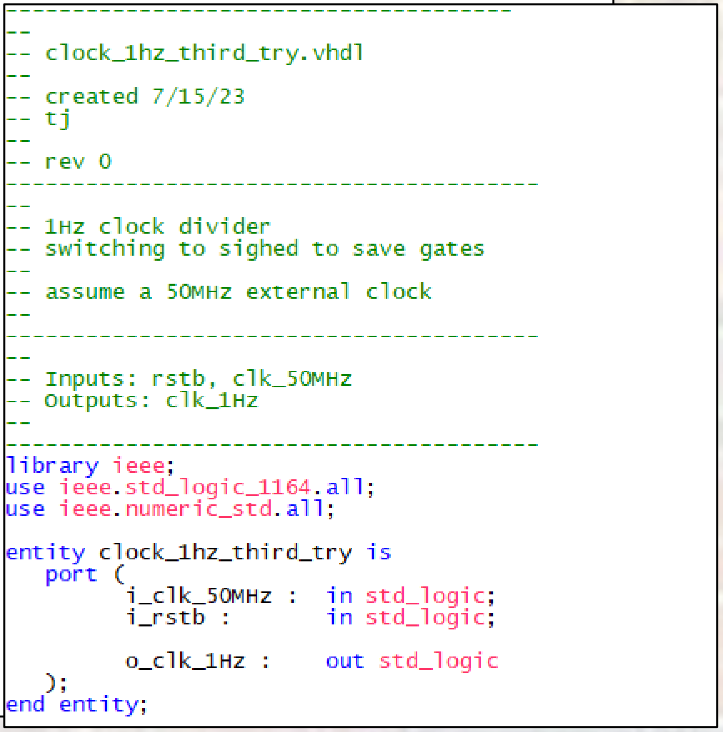


```
architecture behavioral of clock_1hz_third_try is
   --
   -- constants and parameters
   --
   constant INPUT_FREQUENCY:  natural := 50_000_000;
   constant OUTPUT_FREQUENCY: natural := 1;
   constant CLKS_PER_HALF_PERIOD:   signed(25 downto 0) := to_signed(((INPUT_FREQUENCY / 2) / OUTPUT_FREQUENCY), 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 <= (CLKS_PER_HALF_PERIOD - 1);
         clk_sig <= '0';
      elsif (rising_edge(i_clk_50MHz) ) then
         cnt <= cnt - 1;
         --
         -- check if half way
         --
         if (cnt < 0) then
            cnt <= ((CLKS_PER_HALF_PERIOD - 1) - 1);
            clk_sig <= not clk_sig;
         end if;
      end if;
   end process;

   --
   -- output logic
   --
   o_clk_1Hz <= clk_sig;

end behavioral;
```

1 more bit required due to signed range < unsigned range

Comparing to <0

Why???



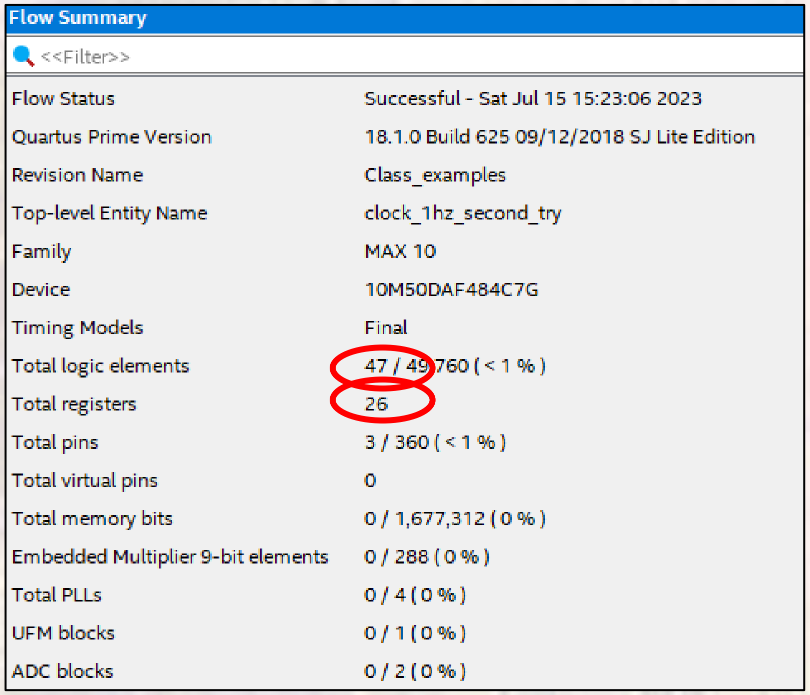
```
----------------------------------------
--
-- clock_1hz_third_try.vhdl
--
-- created 7/15/23
-- tj
--
-- rev 0
----------------------------------------
--
-- 1Hz clock divider
-- switching to sighed to save gates
--
-- assume a 50MHz external clock
--
----------------------------------------
--
-- Inputs: rstb, clk_50MHz
-- Outputs: clk_1Hz
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity clock_1hz_third_try is
   port (
         i_clk_50MHz :   in std_logic;
         i_rstb :        in std_logic;

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

Clock Dividers

- Clock Divider – Comparison
  - 1Hz example

‘best practices’ saved 19 logic elements at the cost of 1 FF

Try 2

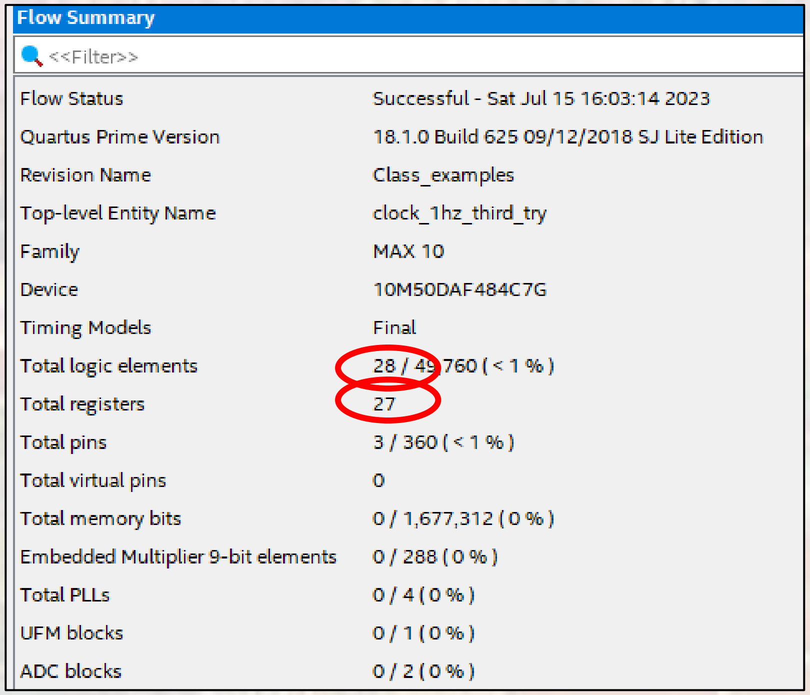


Flow Summary

<<Filter>>

Flow Status	Successful - Sat Jul 15 15:23:06 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Class_examples
Top-level Entity Name	clock_1hz_second_try
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 %)

Try 3



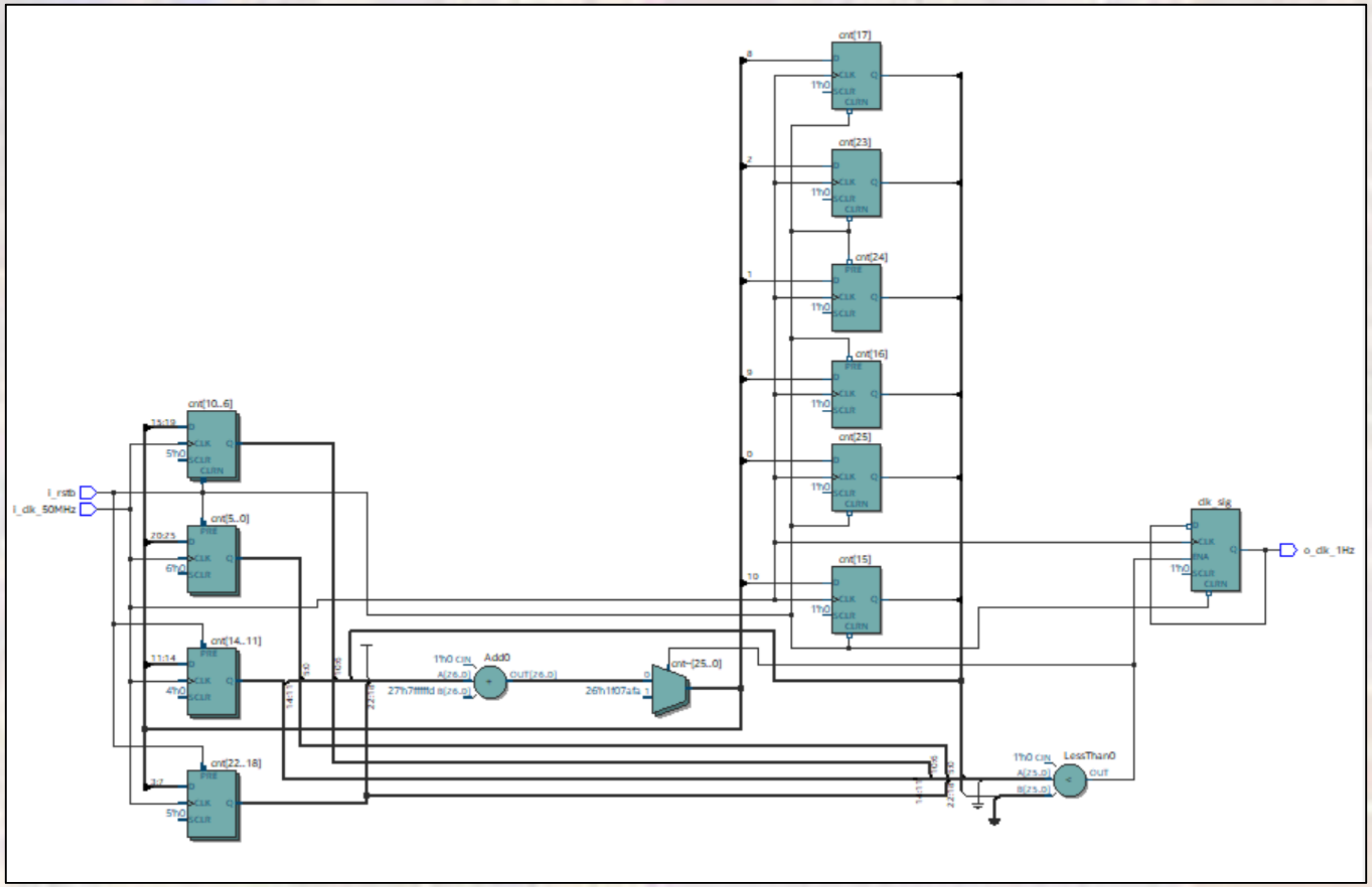
Flow Summary

<<Filter>>

Flow Status	Successful - Sat Jul 15 16:03:14 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	Class_examples
Top-level Entity Name	clock_1hz_third_try
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 %)

Clock Dividers

- Clock Divider – Third try
  - 1Hz example - RTL

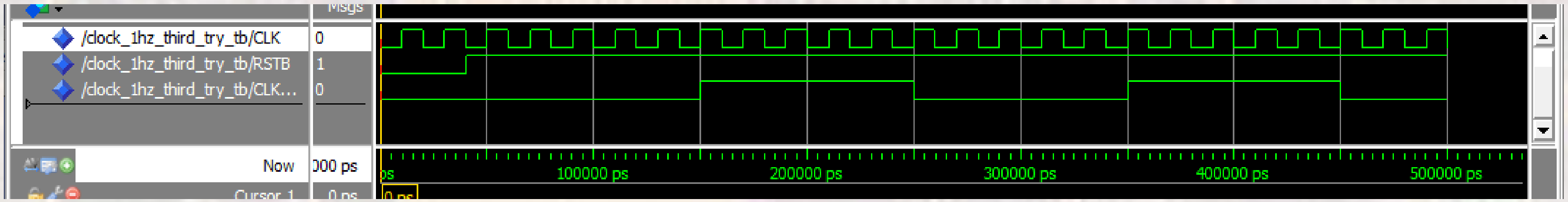


Clock Dividers

- Clock Divider – Third try
  - 1Hz example - Simulation

Modified to do a divide by 10 (5MHz) to reduce simulation time

5 clks / ½ cycle


Msgs
/clock_1hz_third_try_tb/CLK
/clock_1hz_third_try_tb/RSTB
/clock_1hz_third_try_tb/CLK...
0
1
0
Now
000 ps
Cursor 1
0 ps
ps
100000 ps
200000 ps
300000 ps
400000 ps
500000 ps
0 ps

5 MHz

Clock Dividers

- Clock Divider – Fourth try
  - Further optimization – HW/Lab assignment