Quartus Counter Example

Last updated 7/1/26

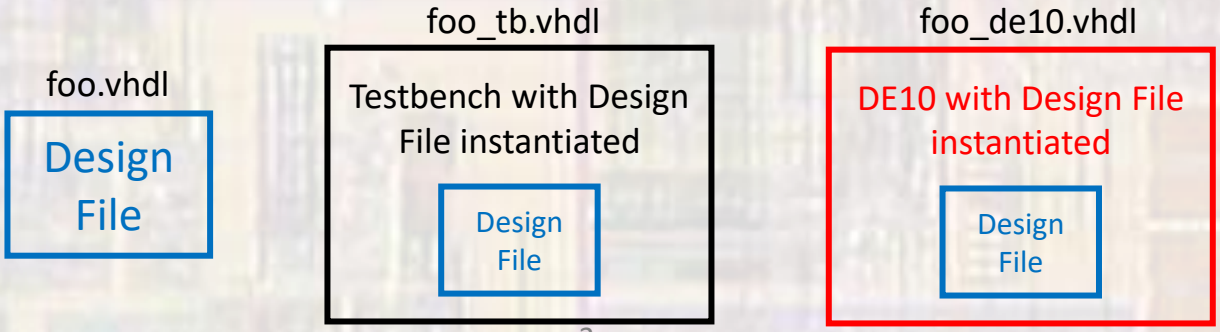
Quartus Counter Example

- Create a logic design from start to a DE10 implementation
  - This example uses “best design practices”
  - This example is not about creating HDL
    - The HDL code will be provided without explanation

© tj

Quartus Counter Example

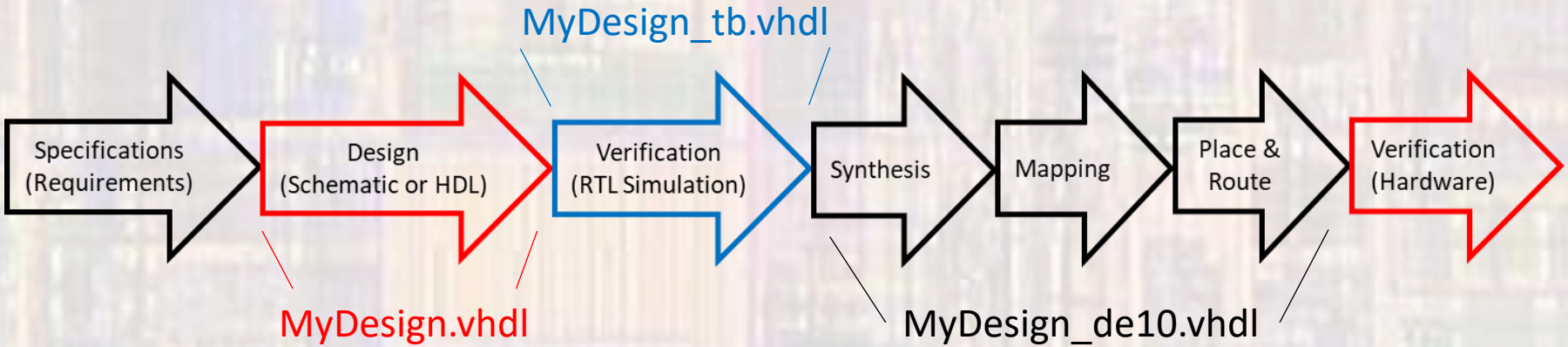
- Design Hierarchy
  - Name your testbench the same as your design file but add _tb to the end
    - counter_8bit.vhdl → counter_8bit_tb.vhdl
    - shift_reg_nbit.vhdl → shift_reg_nbit_tb.vhdl
  - When we start using the DE10 we will do the same thing
    - Name your DE10 implementation the same as your design file but add _de10 to the end
    - counter_8bit.vhdl → counter_8bit_de10.vhdl
    - shift_reg_nbit.vhdl → shift_reg_nbit_de10.vhdl
  - In both cases we never change the base design file
    - Ensures what we designed is what we simulate, and what we simulated is what we build



© tj

Quartus Counter Example

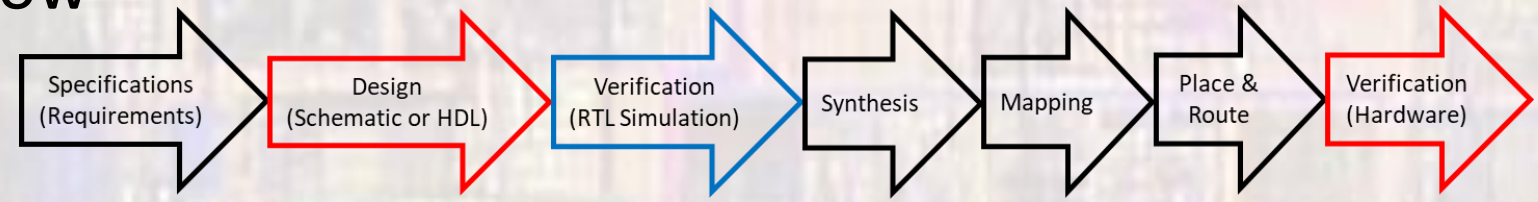
- Flow



© tj

Quartus Counter Example

- Flow

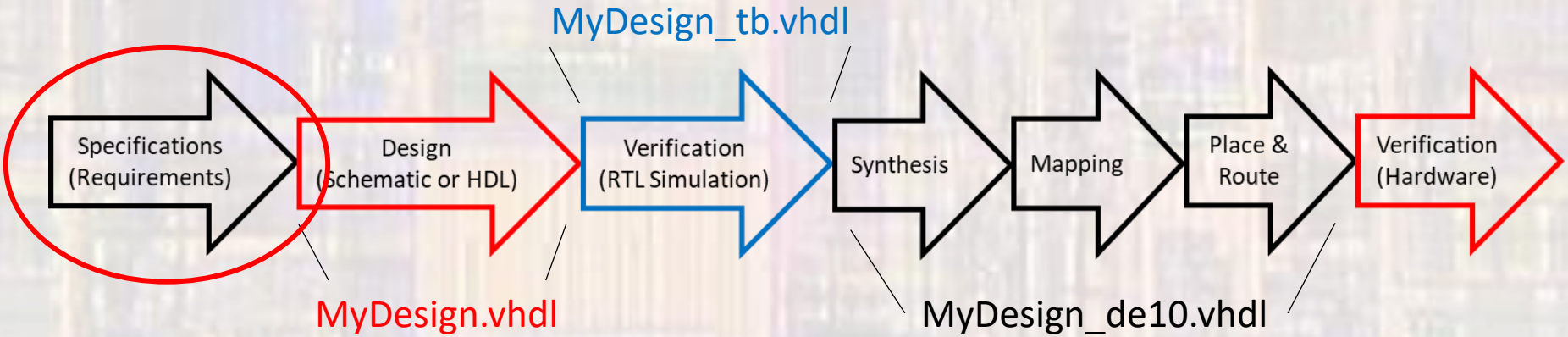


  - Create project
  - Create HDL code
  - Verify RTL visually
  - Create testbench
  - Verify operation
  - Create DE10 implementation HDL
  - Compile
  - Verify RTL visually
  - Program DE10 board
  - Verify operation manually

© tj

Quartus Counter Example

- Flow



© tj

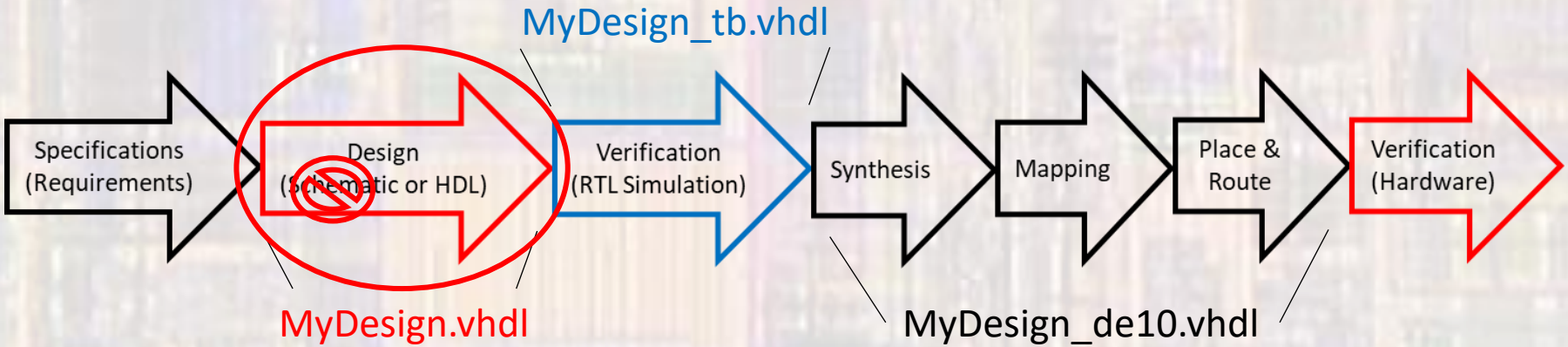
Quartus Counter Example

- Create a counter
  - Binary wrapping count
  - N bits
  - Up/down
- Test the design using Questa
- Implement an 8 bit version on the DE10
  - Reset and direction → switches
  - Count value → LEDs
  - Clock divider → 3Hz operation

© tj

Quartus Counter Example

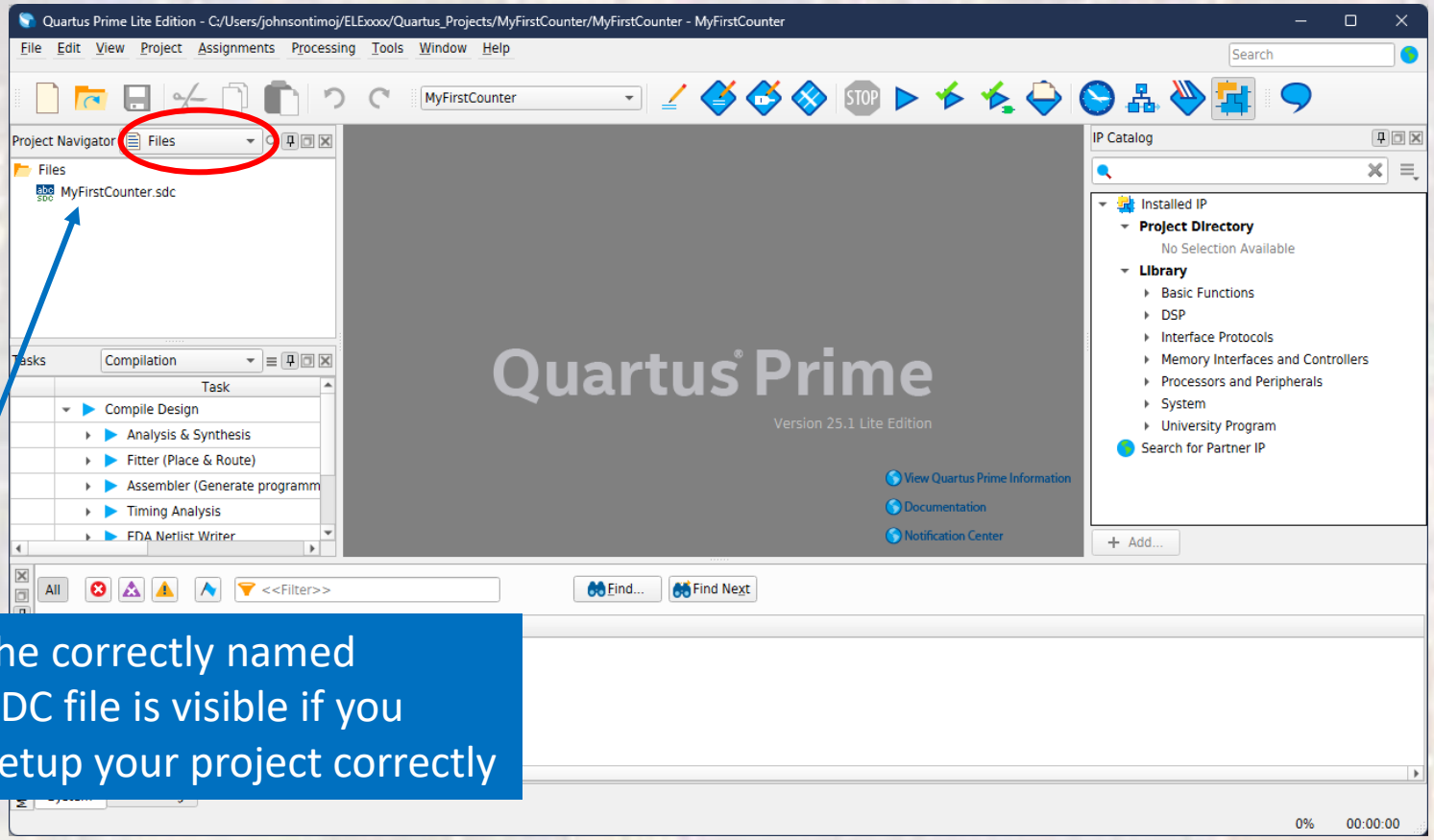
- Flow



© tj

Quartus Counter Example

- Create a new Project
  - Use the Quartus Project Setup notes as a guide
    - Switch to Files view mode

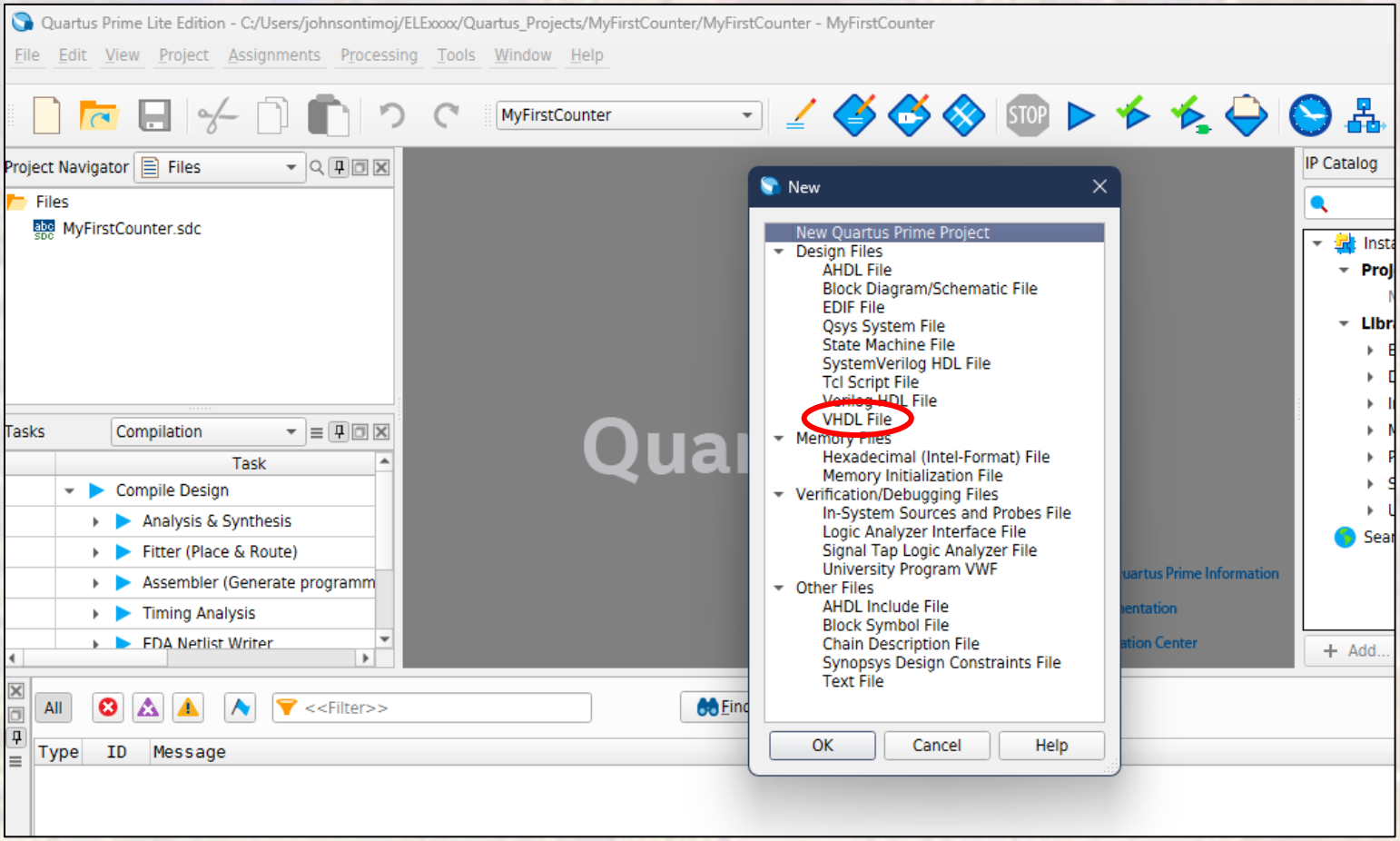


Note: the correctly named SDC file is visible if you setup your project correctly

© tj

Quartus Counter Example

- Create a new HDL file
  - File → New → VHDL File



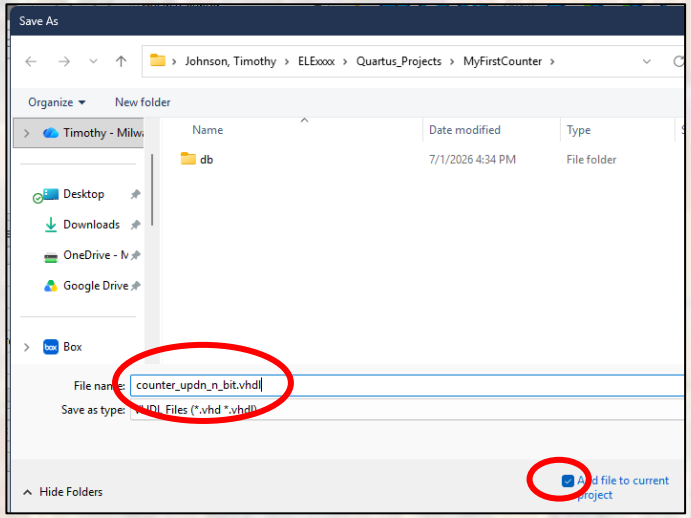
 © tj

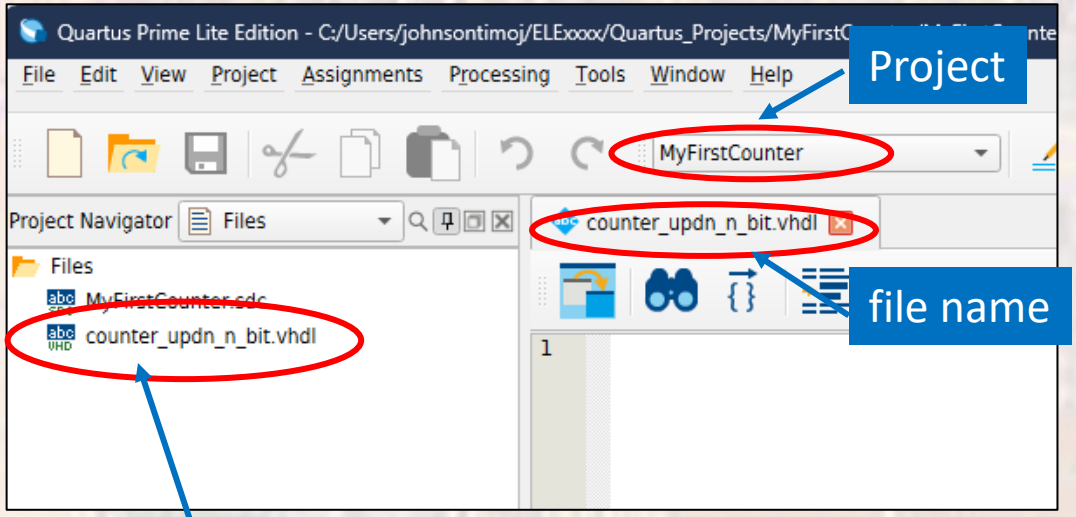
Quartus Counter Example

Use a reverse basis approach so all designs with the same basis are together in the file system making them easier to find

…
counter_mod10
counter_up_nbit
counter_updn_nbit
counter_updn_non-wrap_nbit
…

- Create a new HDL file
  - Note – the default file name is Vhdl.vhd
  - File → Save As and give it an informational file name
  - Change the file extension to vhdl (for VHDL files)
  - Make sure "Add file to current project" is checked



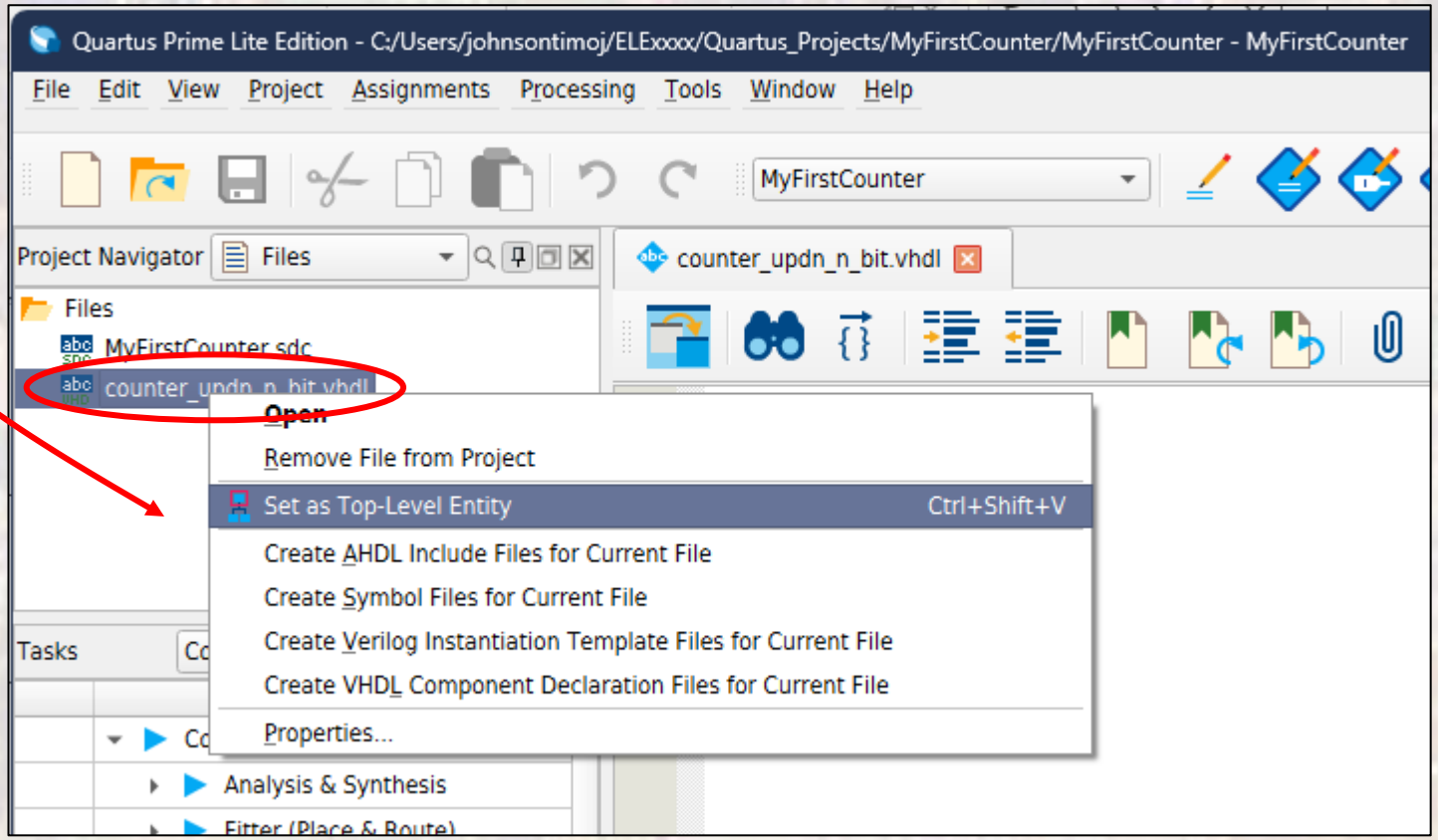


Note: New file is visible

© tj

Quartus Counter Example

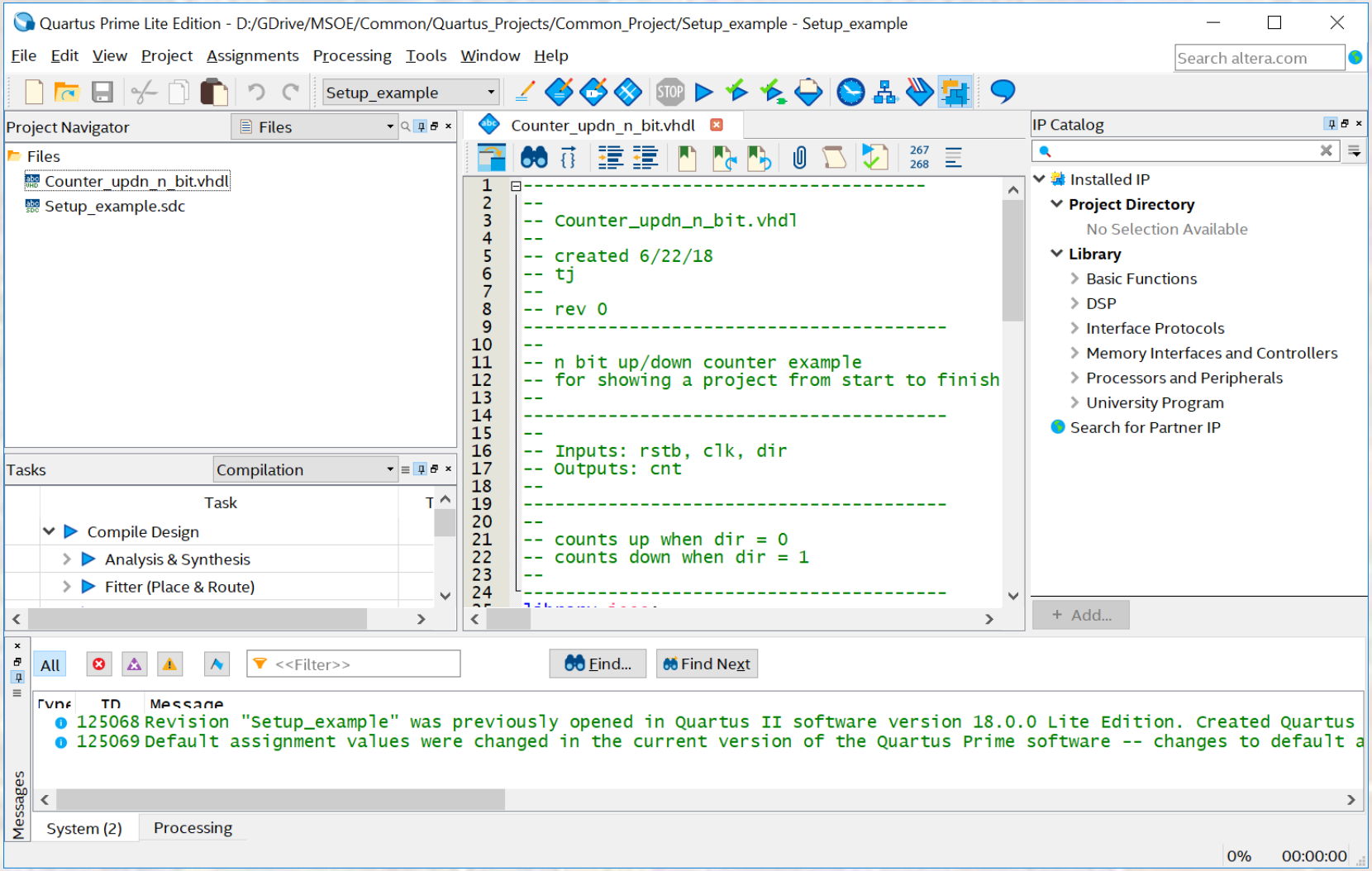
- Set the Top Level Entity
  - Right click on the design file in the Files window
  - Select Set as Top Level Entity



 © tj

Quartus Counter Example

- Enter your design



Quartus Counter Examp

Follow the “best practices” – see web site
- i_, o_ for signals in/out of the block
- std logic for wires - signed, unsigned for signals
- almost everything in a process
- generic whenever possible
- …

```
----------------------------------------
--
-- Counter_updn_n_bit.vhdl
--
-- created 6/22/18
-- tj
--
-- rev 0
----------------------------------------
--
-- n bit up/down counter example
-- for showing a project from start to finish
--
----------------------------------------
--
-- Inputs: rstb, clk, dir
-- Outputs: cnt
--
----------------------------------------
--
-- counts up when dir = 0
-- counts down when dir = 1
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Counter_updn_n_bit is
   generic(
            n: natural := 8
   );
   port (
         i_clk:      in std_logic;
         i_rstb:     in std_logic;
         i_dir:      in std_logic;

         o_cnt :     out std_logic_vector(n-1 downto 0)
   );
end entity;
```

```
architecture behavioral of Counter_updn_n_bit is

   --
   -- internal signals
   --
   signal cnt_sig:   unsigned(n-1 downto 0);

begin

   process(i_clk, i_rstb)
   begin
      --
      -- reset
      --
      if (i_rstb = '0') then
         cnt_sig <= (others => '0');
      --
      -- rising clk edge
      --
      elsif (rising_edge(i_clk)) then
         if(i_dir = '0') then
            cnt_sig <= cnt_sig + 1;
         else
            cnt_sig <= cnt_sig - 1;
         end if;
      end if;
   end process;

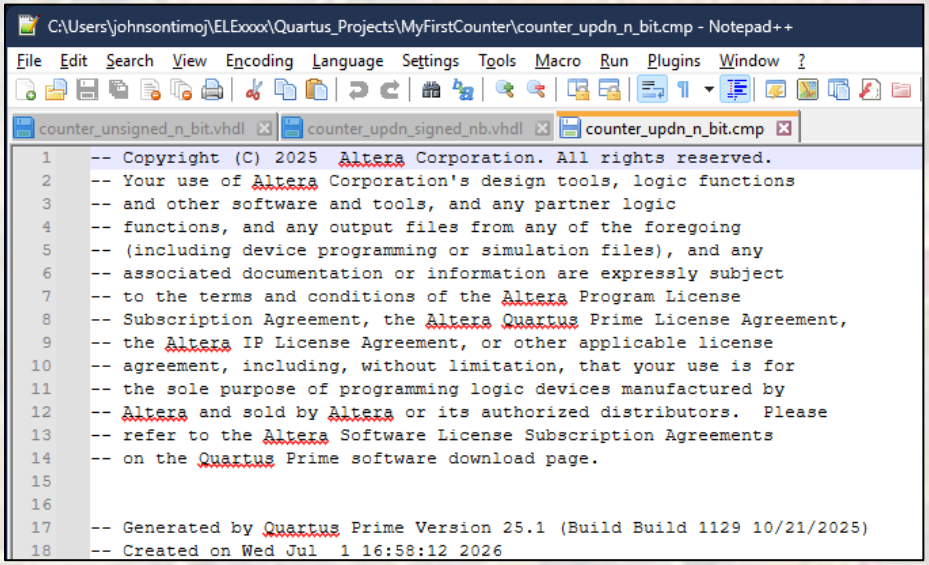
   --
   -- Output logic
   --
   o_cnt <= std_logic_vector(cnt_sig);

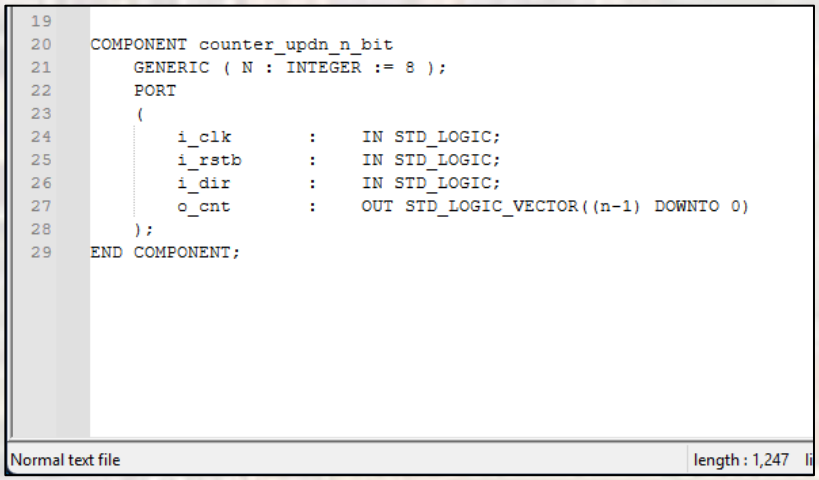
end behavioral;
```


© tj

Quartus Counter Example

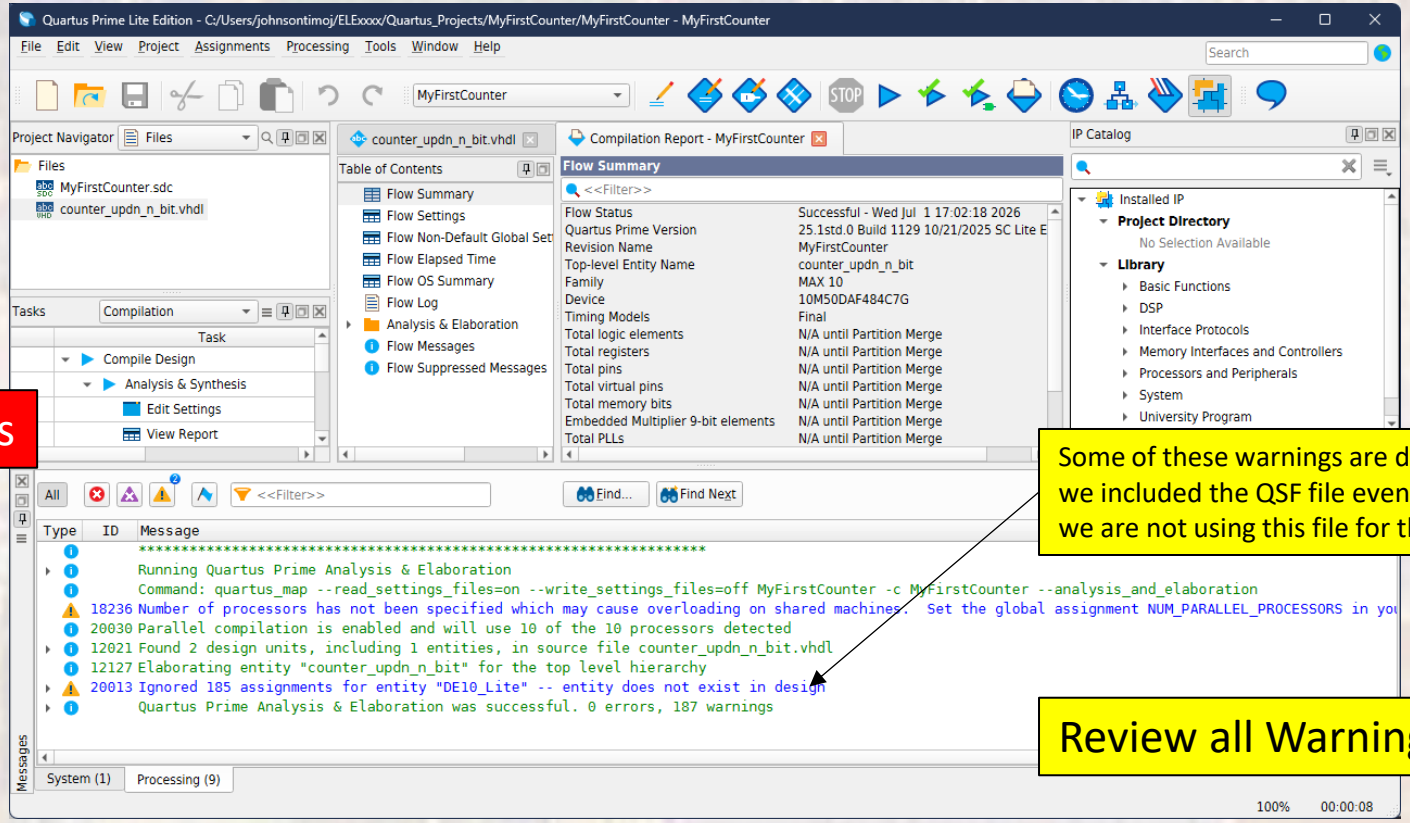
- Enter your design
  - Create a component template for your design (DUT)
    - Used in your testbench and DE10 files
    - Select File → Create/Update → Create VHDL Component Declaration Files from Current File
    - Stored in the project directory as counter_updn_n_bit.cmp





Quartus Counter Example

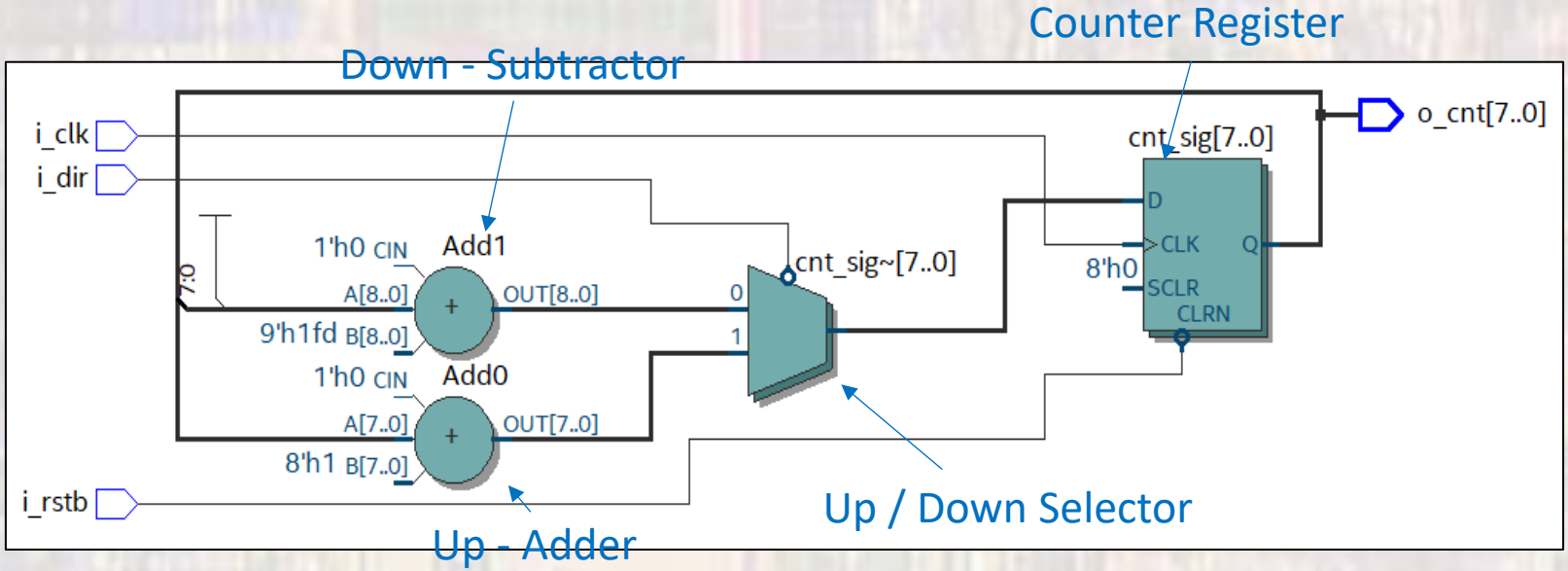
- Elaborate
  - Check the design syntax and create RTL
  - Processing → Start → Start Analysis and Elaboration



 © tj

Quartus Counter Example

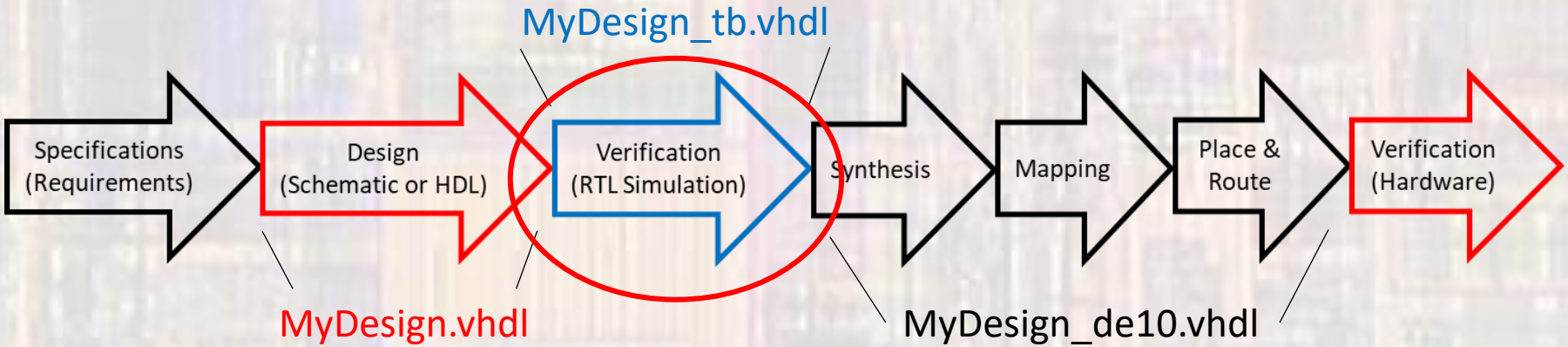
- Verify the RTL
  - Tools → Netlist Viewers → RTL Viewer



 © tj

Quartus Counter Example

- Flow

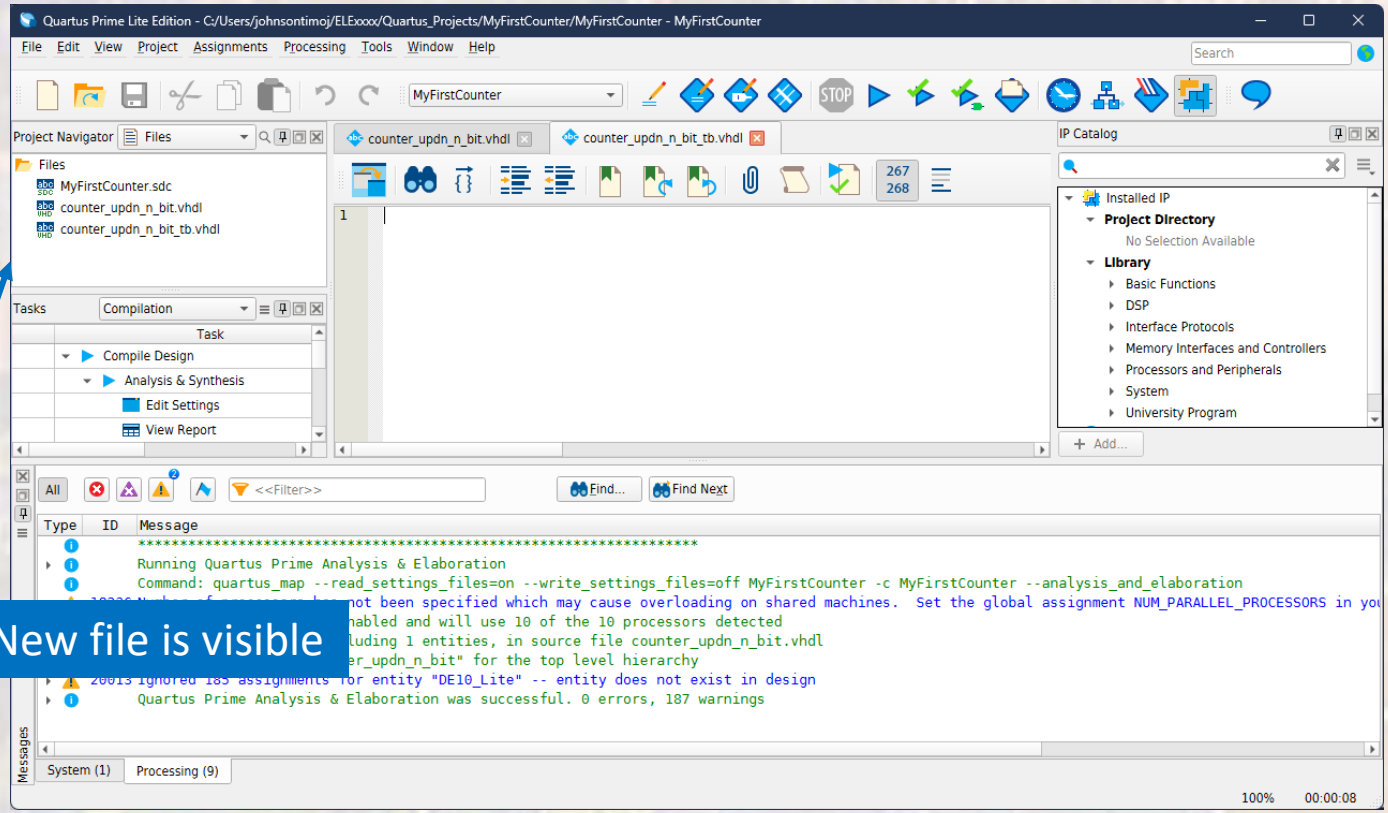


 © tj

Quartus Counter Example

- Create Testbench
  - File → New → VHDL File
  - File → Save As and give it the same name as the top level entity with _tb added

counter_updn_n_bit.vhdl
counter_updn_n_bit_tb.vhdl

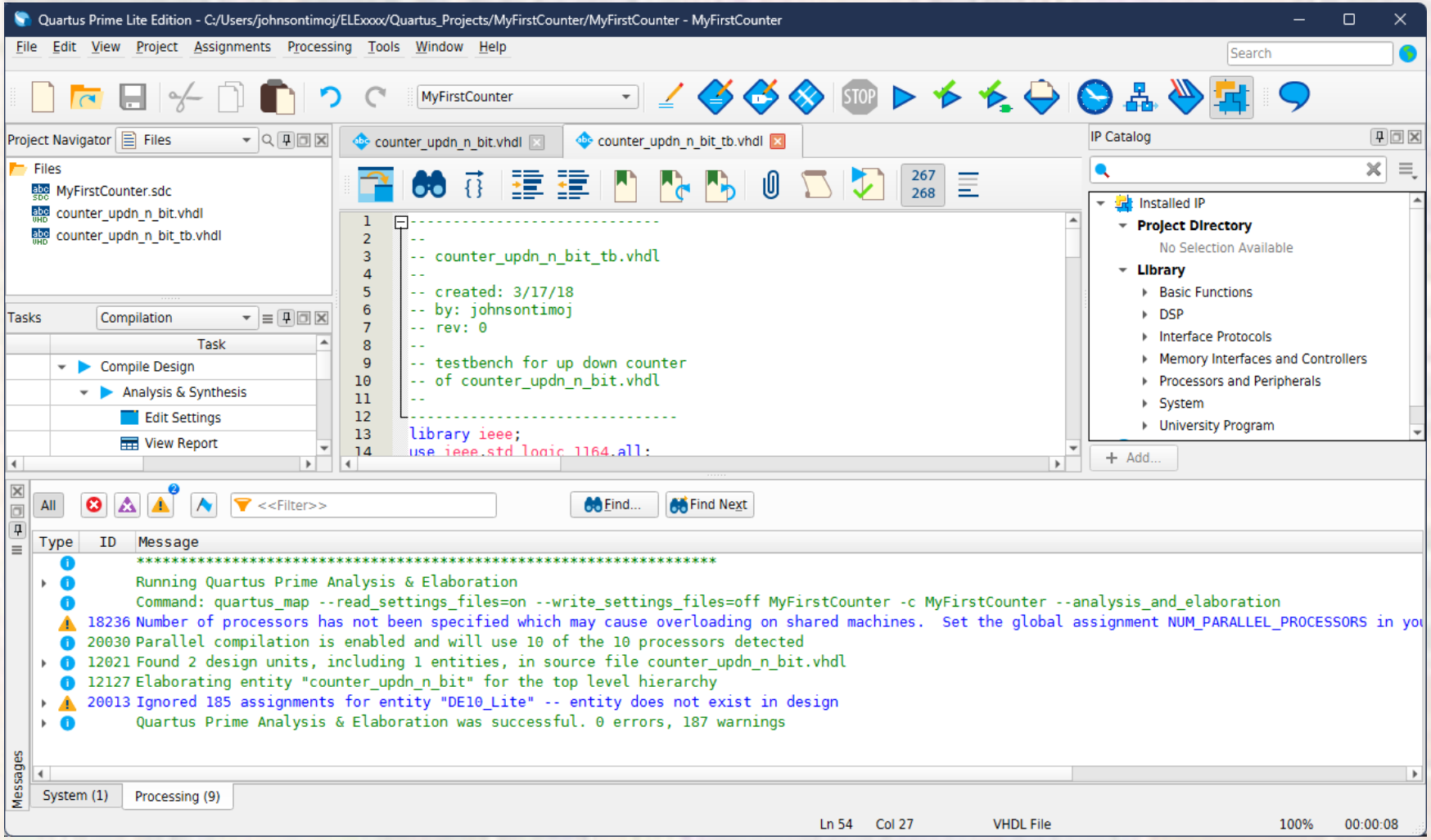


Note: New file is visible

© tj

Quartus Counter Example

- Enter your TB



Quartus Counter Example

Continue to use generics in TB
TB signals in all caps
Define a constant for the clock PER
(no absolute times used)

- Enter your TB

```
------------------------------
--
-- counter_updn_n_bit_tb.vhdl
--
-- created: 3/17/18
-- by: johnsontimoj
-- rev: 0
--
-- testbench for up down counter
-- of counter_updn_n_bit.vhdl
--
------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity counter_updn_n_bit_tb is
   generic(
            NUM_BITS: natural := 6
   );
   -- no port entry - testbench
end entity;
```

```
architecture testbench of counter_updn_n_bit_tb is
   signal   CLK:  std_logic;
   signal   RSTB: std_logic;
   signal   DIR:  std_logic;

   signal   CNT:  std_logic_vector((NUM_BITS - 1) downto 0);

   constant PER:  time:= 20 ns;

   ---------------------------------
   -- Component prototype
   ---------------------------------
   COMPONENT counter_updn_n_bit
      generic(
               N: natural := 8
      );
      port (
            i_clk:   in std_logic;
            i_rstb:  in std_logic;
            i_dir:   in std_logic;

            o_cnt :  out std_logic_vector((N - 1) downto 0)
      );
   END COMPONENT;
   -----------------------------------

   begin

   -----------------------------------
   -- Device under test (DUT)
   -----------------------------------
   DUT: counter_updn_n_bit
      generic map(
                  n => NUM_BITS
                  )
      port map(
               i_clk    => CLK,
               i_rstb   => RSTB,
               i_dir    => DIR,
               o_cnt    => CNT
               );
```


© tj

Quartus Counter Example

Separate clock, reset and run processes
Could be 0, 1, 2, …. Run processes

- Enter your TB

```
  -----------------------------------
  -- Test processes
  -----------------------------------

  -- Clock process
  clock: process    -- no sensitivity list allowed
     begin
        CLK <= '0';
        wait for PER/2;
        infinite: loop
           CLK <= not CLK; wait for PER/2;
        end loop;
  end process clock;

  -- Reset process
  reset: process    -- no sensitivity list allowed
  begin
     RSTB <= '0'; wait for 2*PER;
     RSTB <= '1'; wait;
  end process reset;

  -- Run Process
  run: process      -- no sensitivity list allowed
  begin
     -- initialize inputs
     DIR <= '0';
     -- run code
     wait for 68*PER;
     DIR <= '1';
     wait for 68*PER;
  end process run;

  -----------------------------------------
  -- End test processes
  -----------------------------------------

end architecture;
```


© tj

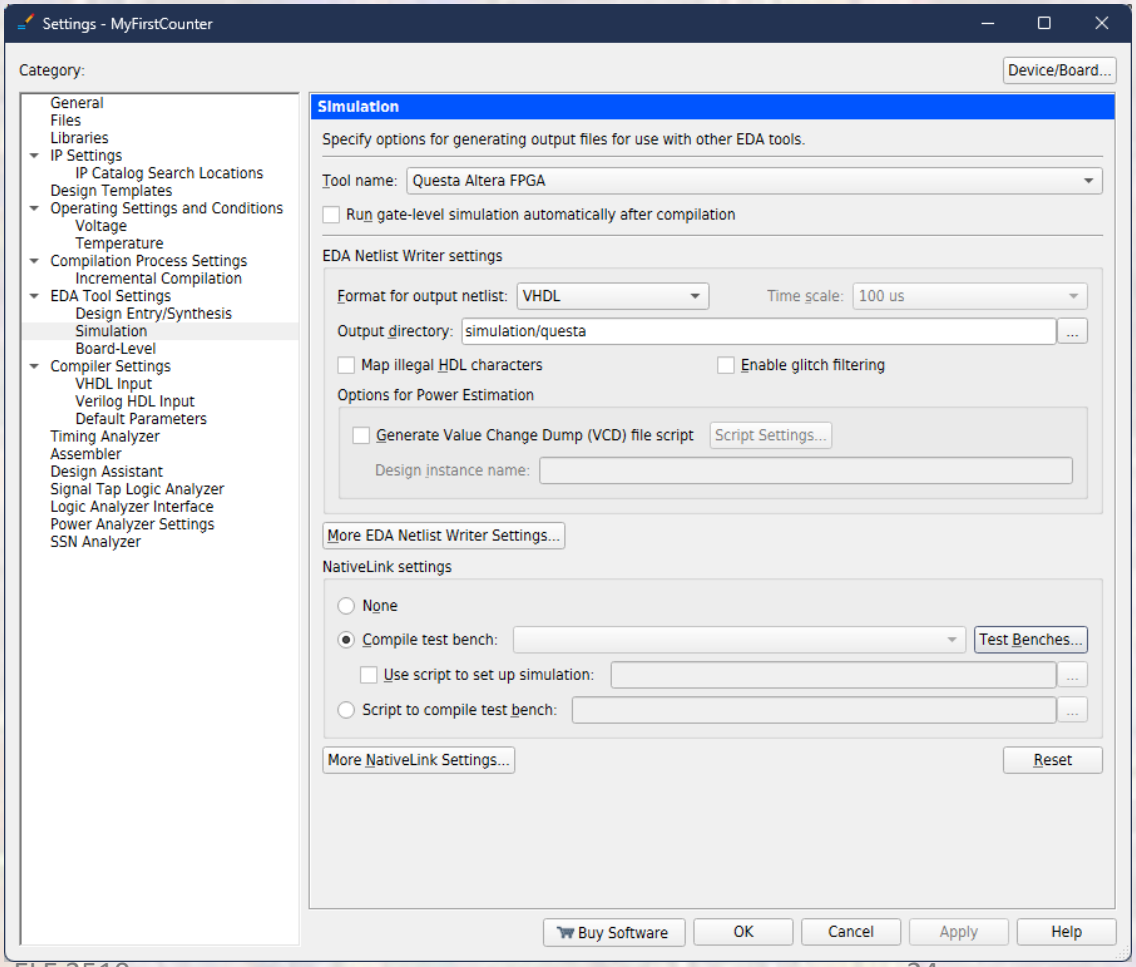
Quartus Counter Example

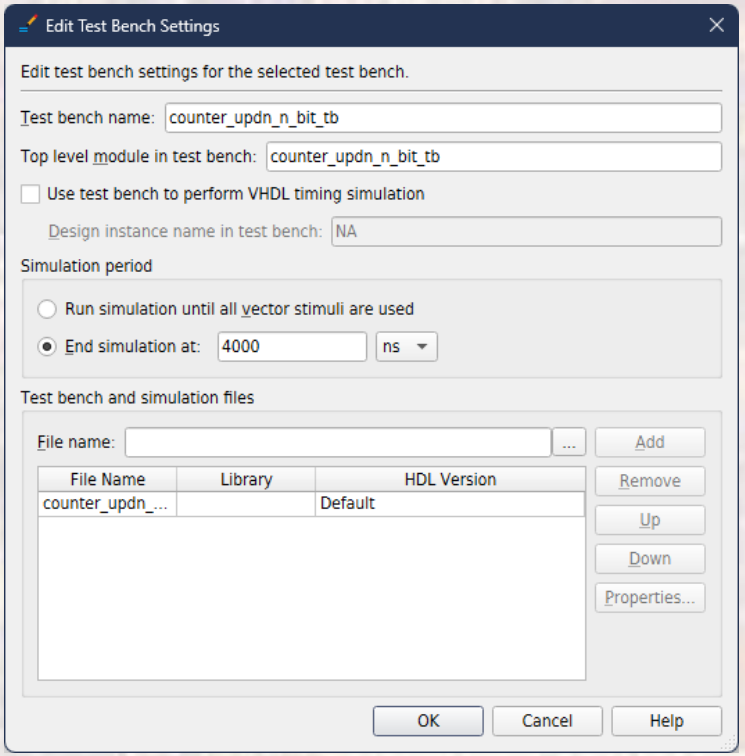
- Enter your TB
  - Elaborate the design
    - With the original vhdl design set as the top level entity (not the xxxx_tb.vhdl design)
    - Select Processing → Start → Start Analysis and Elaboration
  - This causes Quartus to check the Test Bench code along with the original vhdl design

© tj

Quartus Counter Example

- Setup your simulation
  - Use the Questa Testbench Setup notes as a guide

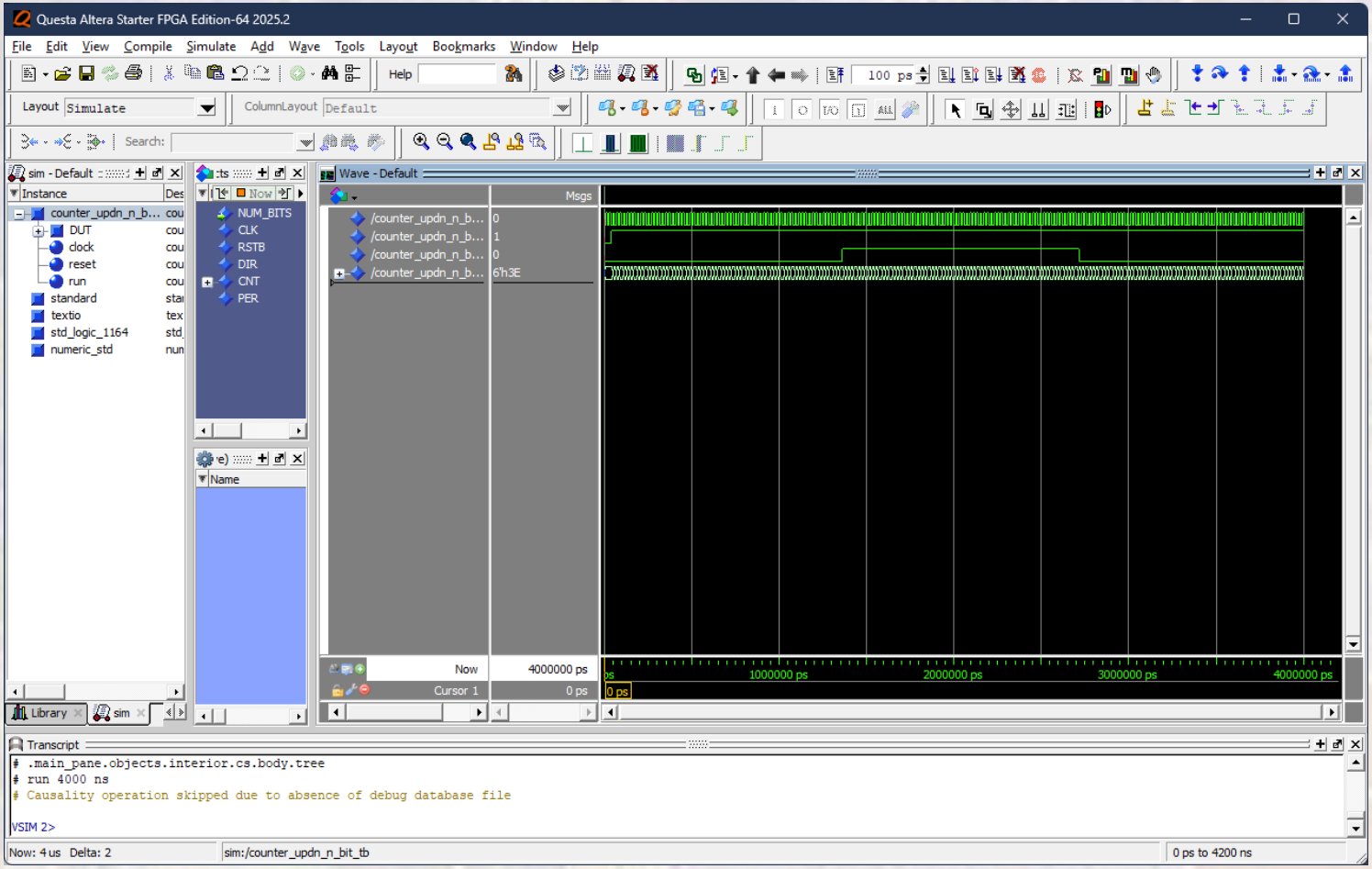




© tj

Quartus Counter Example

- Run your simulation
  - Tools → Run Simulation Tools → RTL Simulation

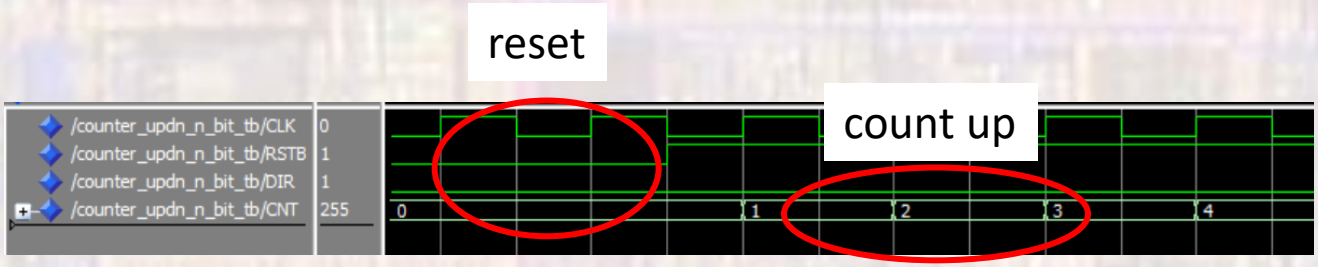


 © tj

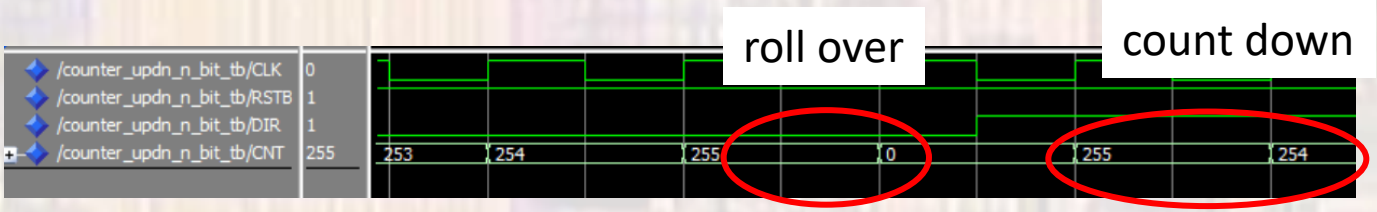
Quartus Counter Example

- Verify your design

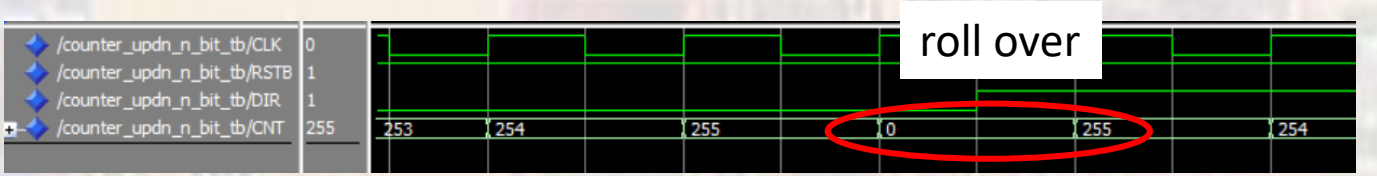
Reset & Count up



Wrap up & Count down



Wrap down



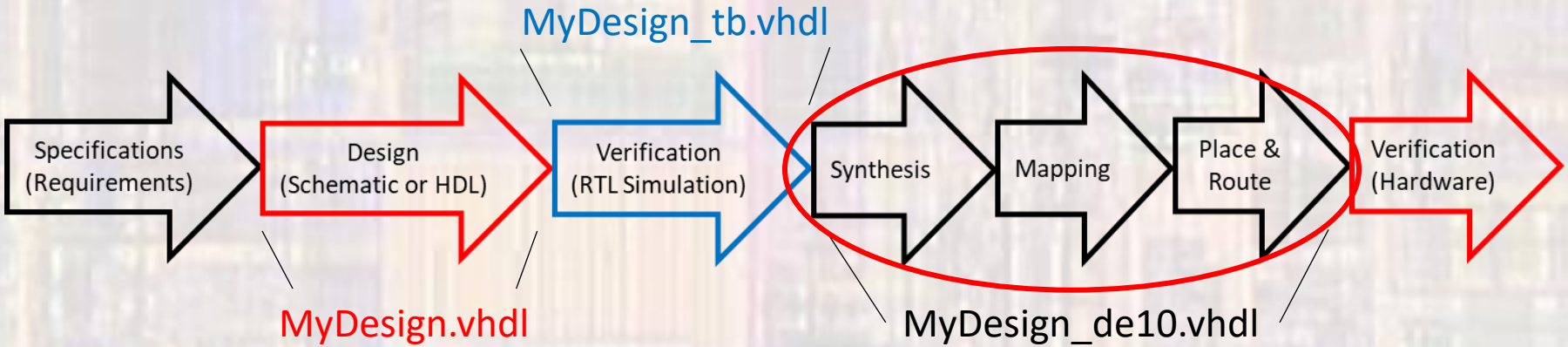


signal names visible

© tj

Quartus Counter Example

- Flow

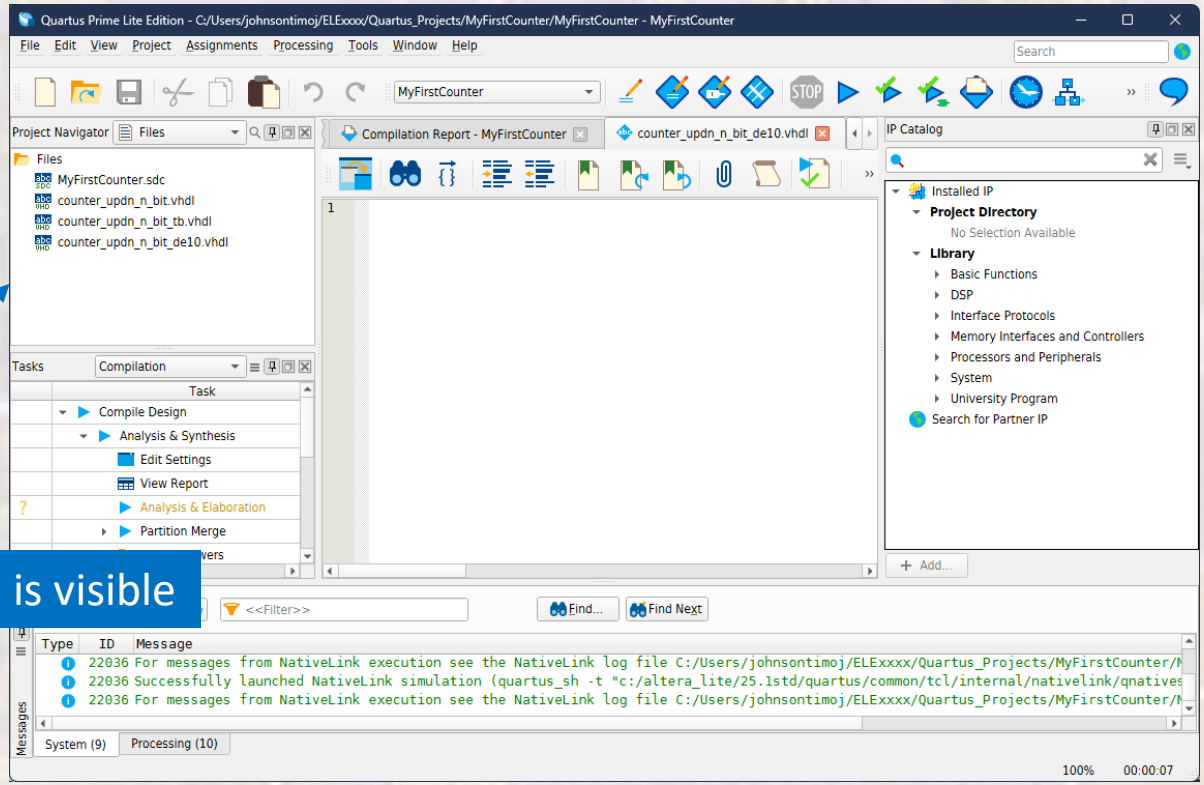


 © tj

Quartus Counter Example

- Create DE10 implementation HDL
  - File → New → VHDL File
  - File → Save As and give it the same name as the top level entity with _de10 added

counter_updn_n_bit.vhdl
counter_updn_n_bit_tb.vhdl
counter_updn_n_bit_de10.vhdl

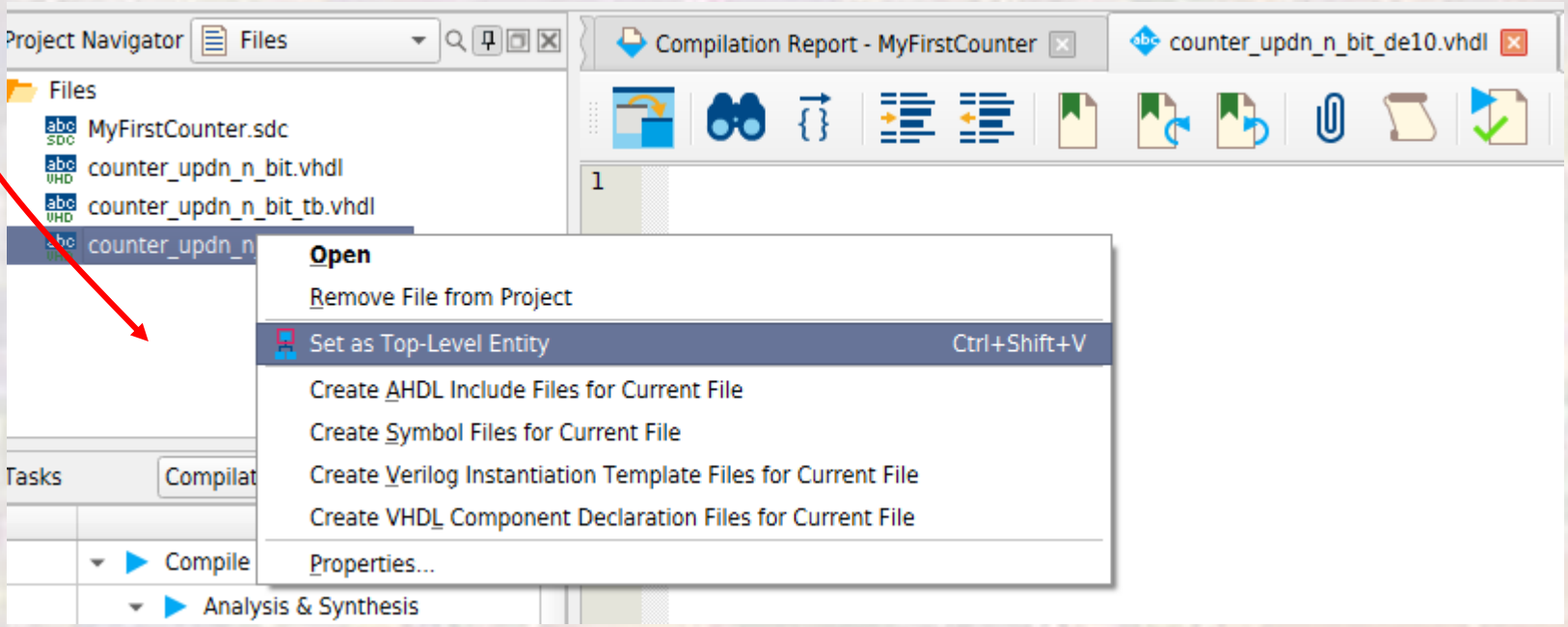


Note: New file is visible

© tj

Quartus Counter Example

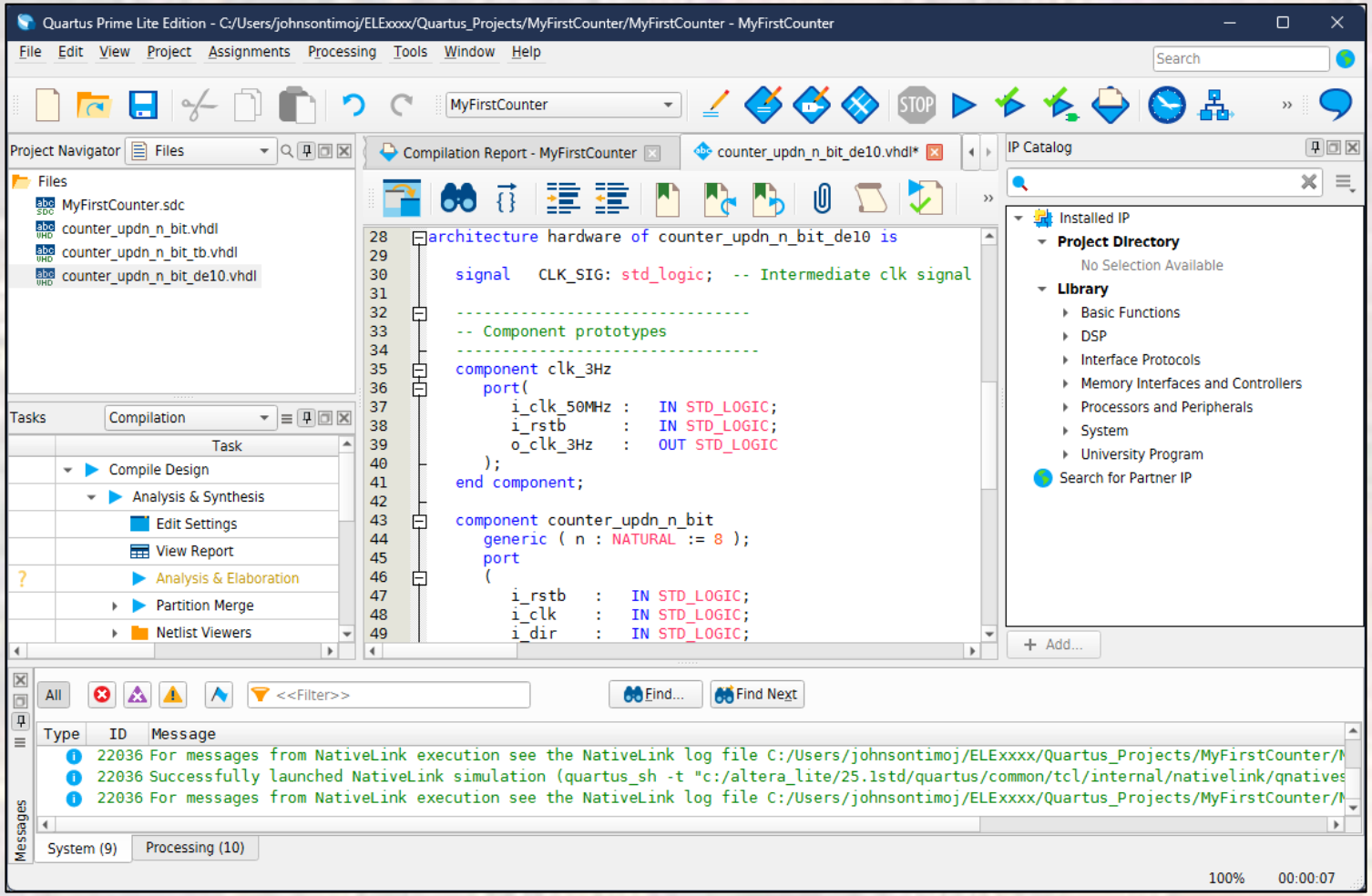
- Set the Top Level Entity
  - Right click on the de10 file in the Files window
  - Select Set as Top Level Entity



 © tj

Quartus Counter Example

- Enter your DE10 design



 © tj

Quartus Counter Example

Use qsf pin names for the top level I/Os

- Enter your DE10 design

```
--------------------------------
--
-- counter_updn_n_bit_de10.vhdl
--
-- created: 3/17/18
-- by: johnsontimoj
-- rev: 0
--
-- DE10 implementation for up down counter
-- of counter_updn_n_bit.vhdl
--
-- Uses a 3 Hz clock divider, sw0 for reset
-- sw1 for the dir input
-- and LEDs for the count output
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity counter_updn_n_bit_de10 is
   port(
         CLOCK_50:    in    std_logic;
         SW:          in    std_logic_vector(1 downto 0);

         LEDR:        out   std_logic_vector(7 downto 0)
   );
end entity;
```

```
architecture hardware of counter_updn_n_bit_de10 is

   signal   CLK_SIG: std_logic;  -- Intermediate clk signal

   ---------------------------------
   -- Component prototypes
   ---------------------------------
   component clk_3Hz
      port(
         i_clk_50MHz :   IN STD_LOGIC;
         i_rstb      :   IN STD_LOGIC;
         o_clk_3Hz   :   OUT STD_LOGIC
      );
   end component;

   component counter_updn_n_bit
      generic ( n : NATURAL := 8 );
      port
      (
         i_rstb   :   IN STD_LOGIC;
         i_clk    :   IN STD_LOGIC;
         i_dir    :   IN STD_LOGIC;
         o_cnt :   OUT STD_LOGIC_VECTOR((n - 1) downto 0)
      );
   end component;
   ---------------------------------

   begin
```


© tj

Quartus Counter Example

- Enter your DE10 design

```
   -------------------------------------
   -- Device under test (DUT)
   -------------------------------------
   CK: clk_3Hz
      port map(
         i_clk_50MHz    =>  CLOCK_50,
         i_rstb         =>  SW(0),
         o_clk_3Hz      =>  CLK_SIG
      );

   DUT: counter_updn_n_bit
      port map(
               i_clk    => CLK_SIG,
               i_rstb   => SW(0),
               i_dir    => SW(1),
               o_cnt    => LEDR(7 downto 0)
      );

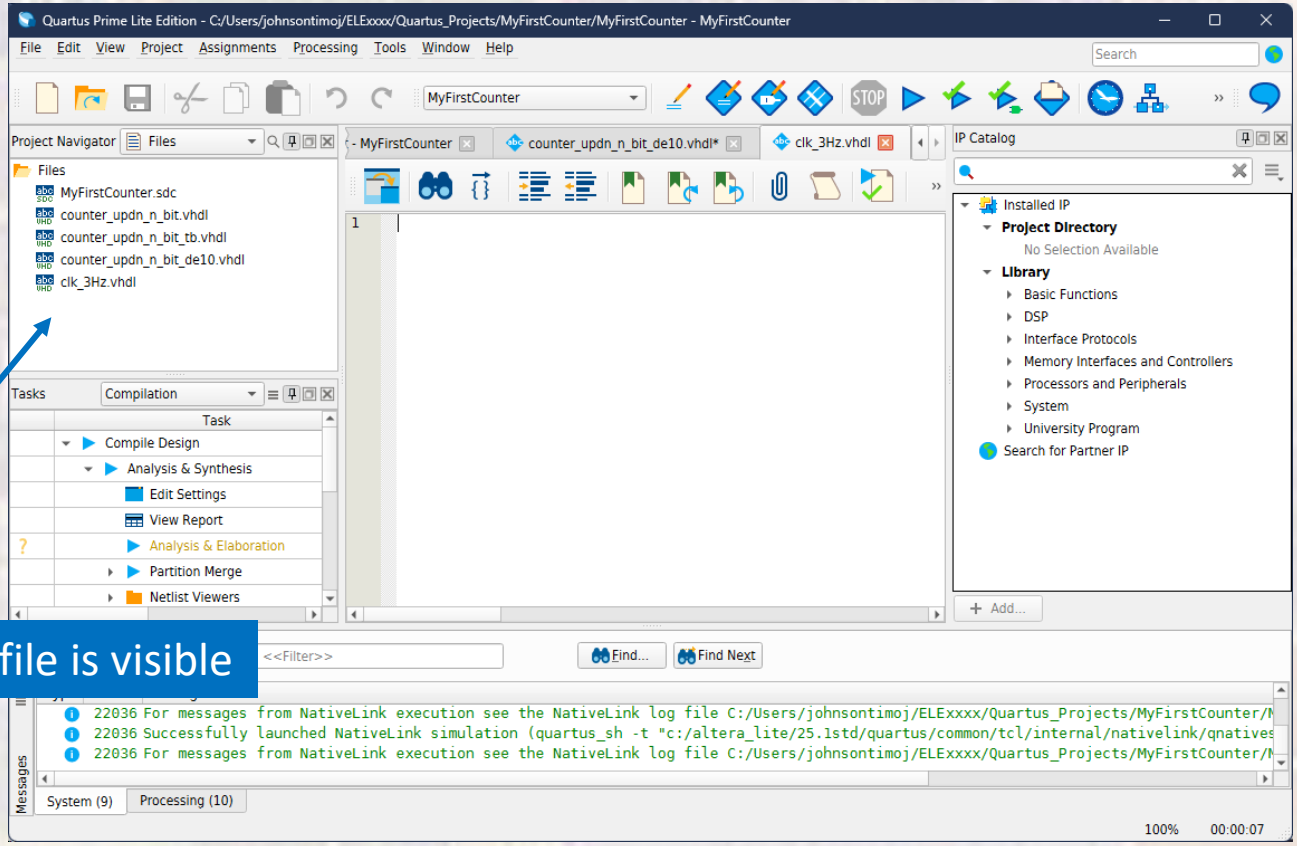
end architecture;
```


© tj

Quartus Counter Example

- Create a clock divider
  - File → New → VHDL File
  - File → Save As and give it an informational file name

clk_3Hz.vhdl



Note: New file is visible

© tj

Quartus Counter Example

- Create 3Hz clock divider

```
----------------------------------------
--
-- clk_3hz.vhdl
--
-- created 2/20/18
-- tj
--
-- rev 0
----------------------------------------
--
-- 3Hz clock divider for use with DE10 Labs
--
-- assumes 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 clk_3hz is
   port (
         i_clk_50MHz :  in std_logic;
         i_rstb :       in std_logic;

         o_clk_3Hz :    out std_logic
   );
end entity;
```

Quartus Counter Example

- Create 3Hz clock divider

```
architecture behavioral of clk_3hz is
   --
   -- constants and parameters
   --
   constant CLOCKS_PER_HALF_PERIOD: signed(24 downto 0) := to_signed((((50_000_000 / 2) /3) - 2), 25);
   --
   -- internal signals
   --
   signal cnt:       signed(24 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;

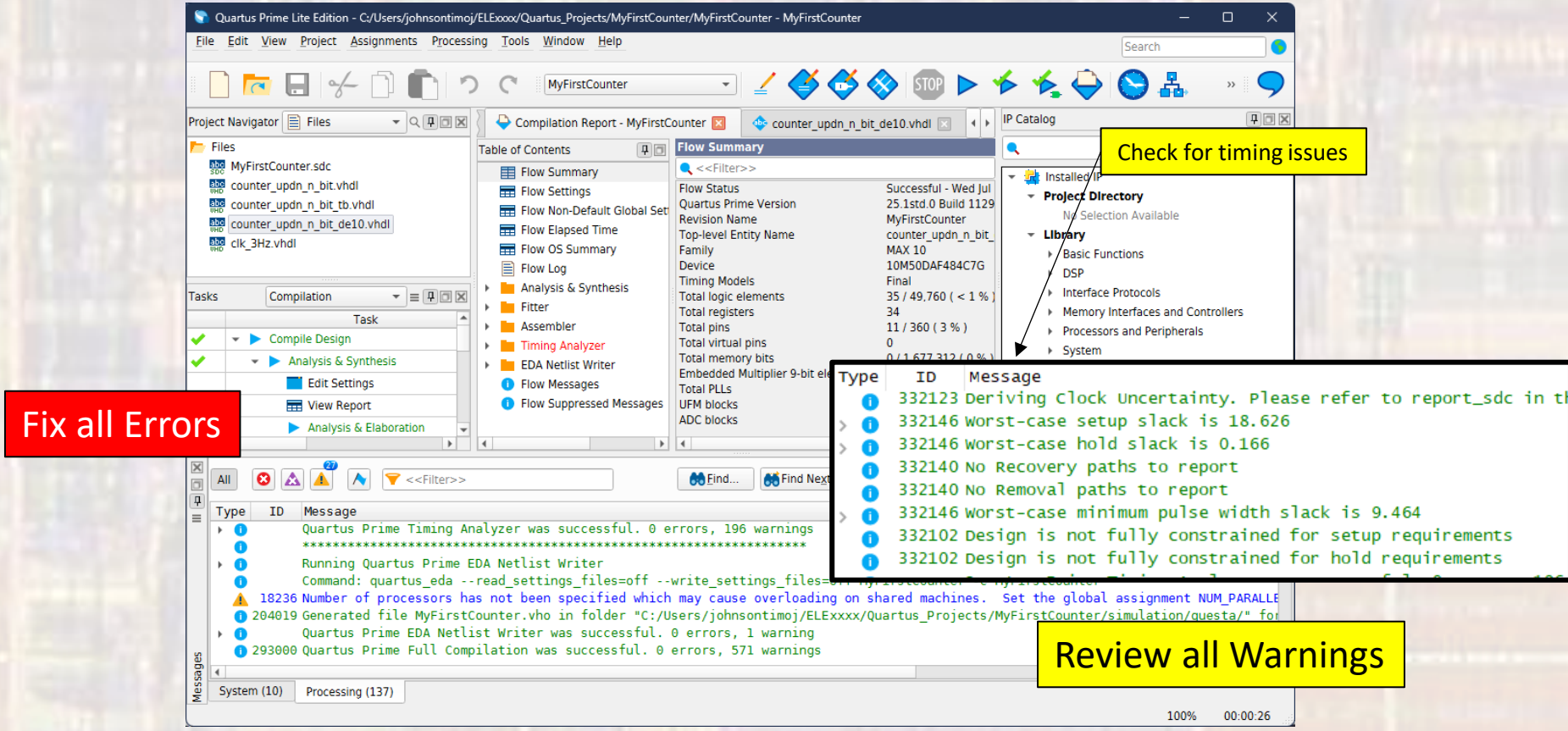
   --
   -- Output logic
   --
   o_clk_3Hz <= clk_sig;

end behavioral;
```


© tj

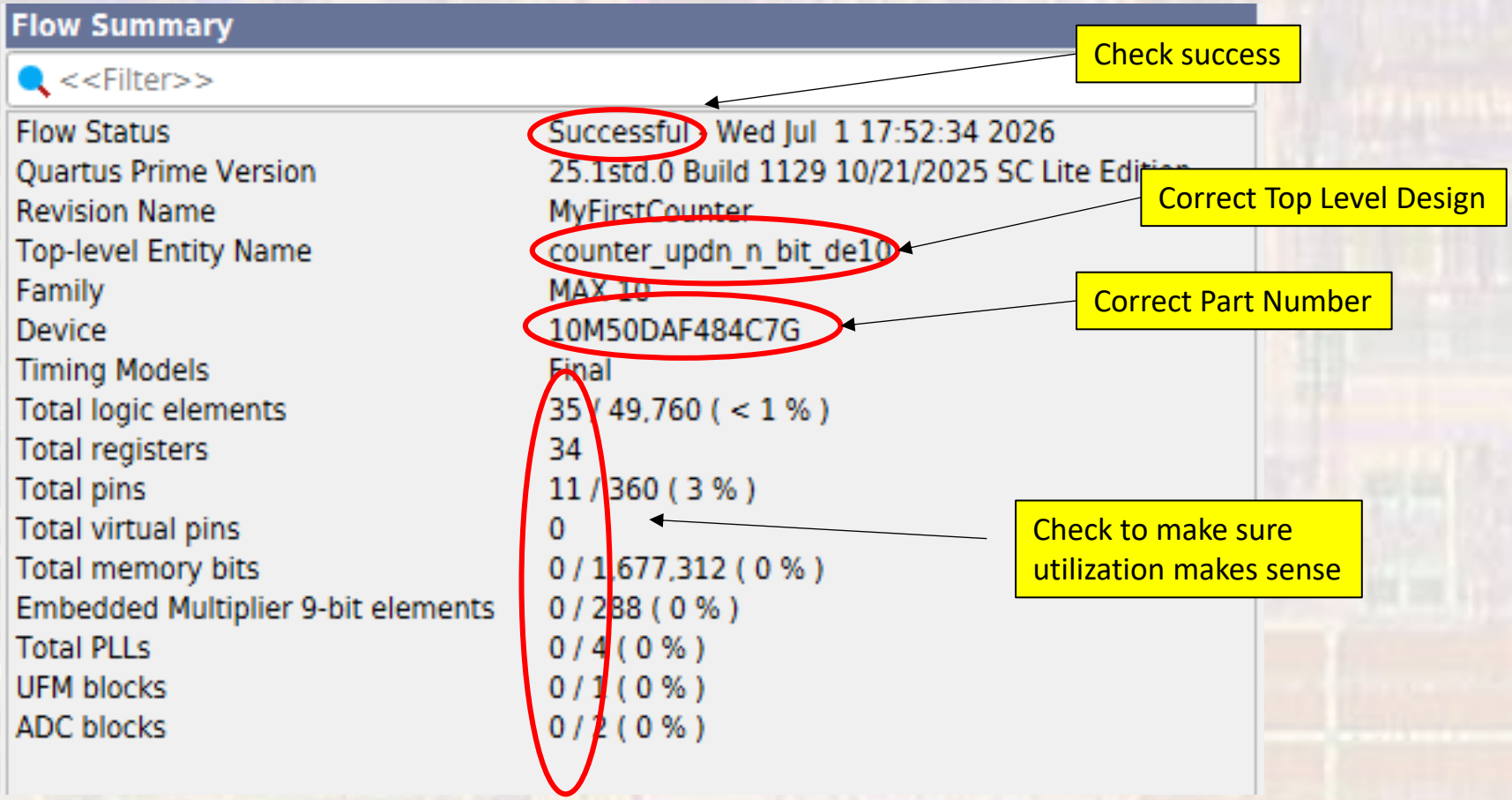
Quartus Counter Example

- Compile
  - Check the design syntax, create RTL, Place and Route
  - Processing → Start Compilation



Quartus Counter Example

- Compile
  - Check the Flow Summary

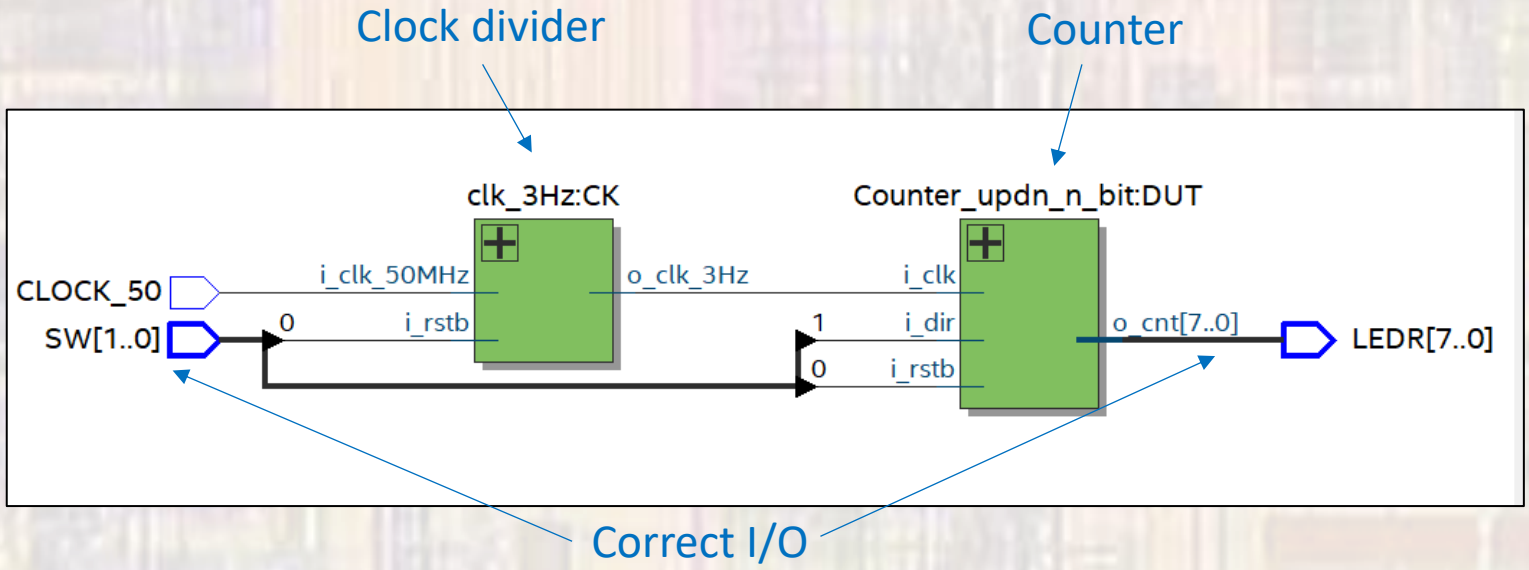


 © tj

Quartus Counter Example

- Verify the RTL
  - Tools → Netlist Viewers → RTL Viewer

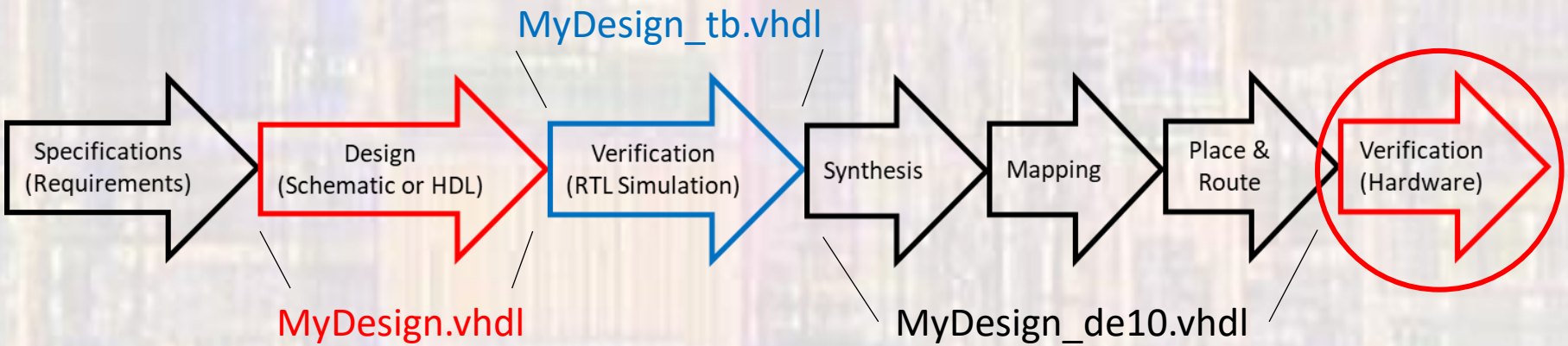
Note: this IS NOT sufficient for a design report – you need to show RTL for all blocks you created



 © tj

Quartus Counter Example

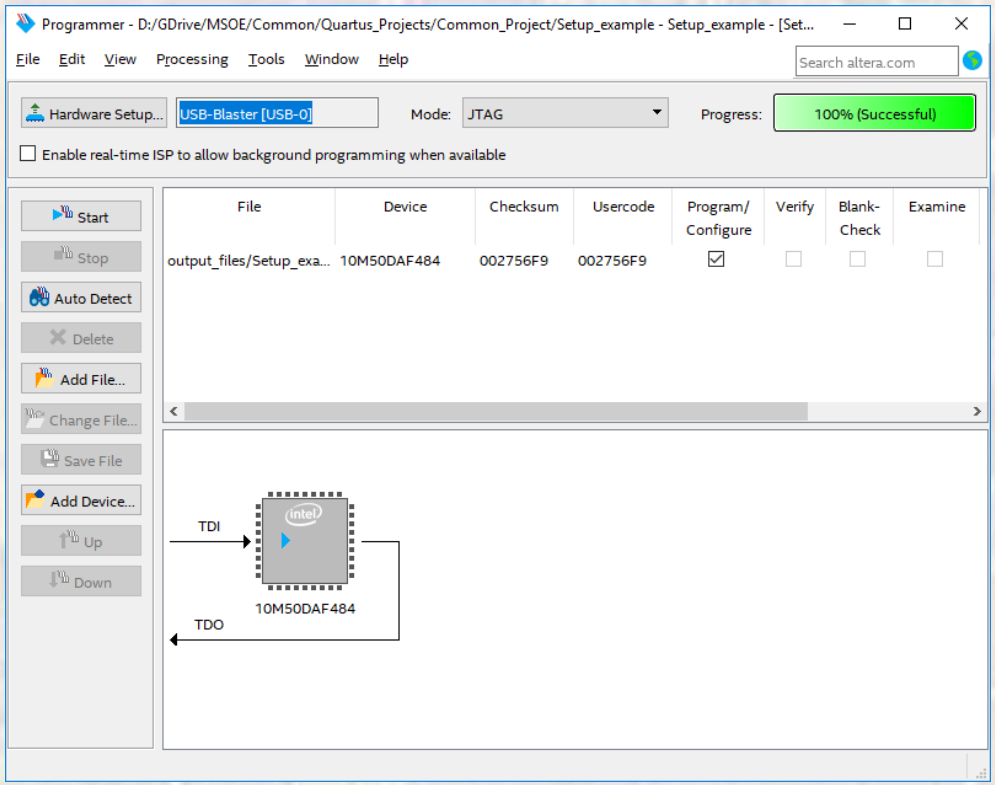
- Flow



 © tj

Quartus Counter Example

- Program the board
  - Connect the DE10 board to computer
  - Tools → Programmer
    - Point to the compiled output file xxxx.sof
  - Start



 © tj

Quartus Counter Example

- Verify Operation
  - video



© tj