FPGA Design Flow

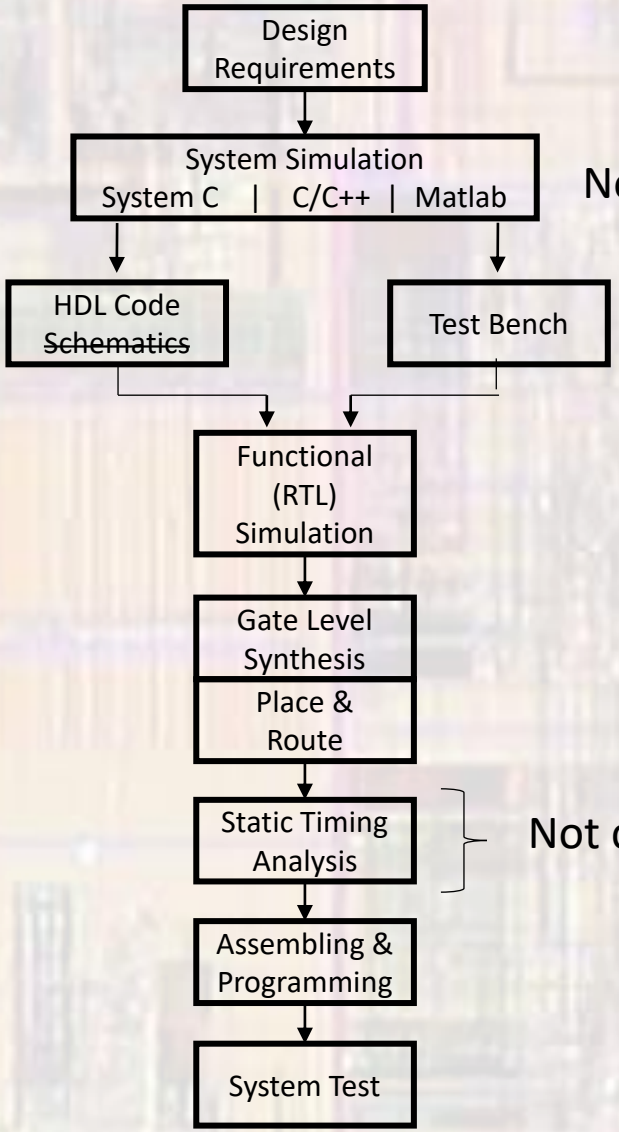
Last updated 1/20/21

FPGA Design Flow

These slides outline the FPGA design flow used in this class

Upon completion: You should be able to describe each step of the design flow and identify the appropriate tools used in each step

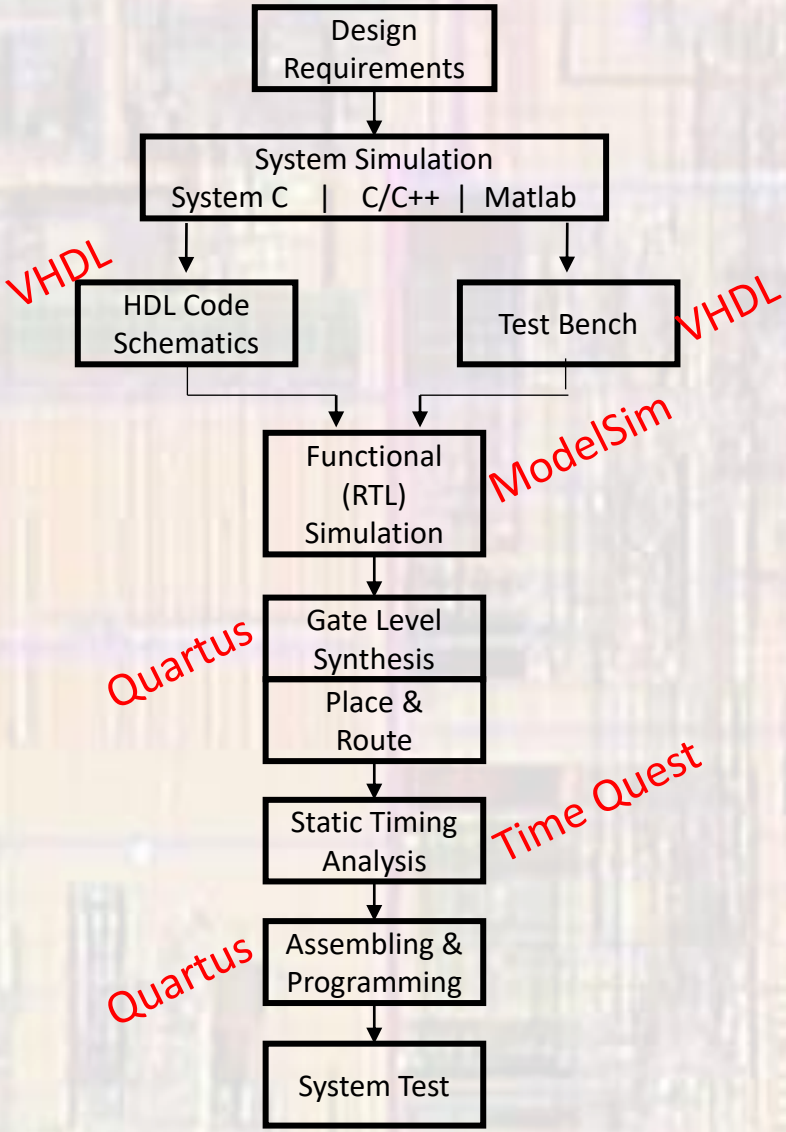
FPGA Design Flow



Not covered in CE1911

Not covered in CE1911

FPGA Design Flow

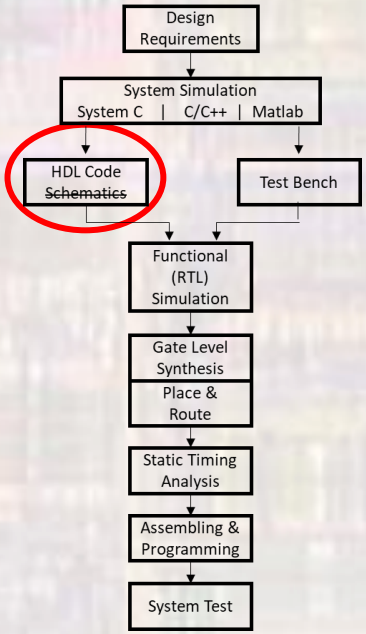


Additional Tools:
Debug – Signal Tap

FPGA Design Flow

- Design Entry
  - Text entry
    - Hardware Description Language
    - VHDL , Verilog, System C, …
    - Hierarchical instantiation of blocks

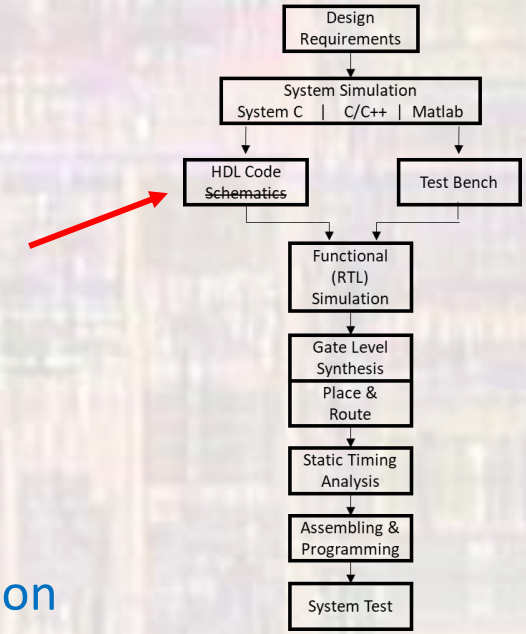
```
count: process(i_clk, i_rstb)
begin
   if(i_rstb = '0') then
      cnt_sig <= (others => '0');
   elsif(rising_edge(i_clk)) then
      if(i_dir = '0') then
         cnt_sig <= cnt_sig + 1;
      else
         cnt_sig <= cnt_sig - 1;
      end if;
   end if;
end process;
```



  - Schematic entry
    - Quartus Block Editor
      - Create bdf schematic files
    - Quartus Symbol Editor
      - Create / modify symbols for the block editor (bsf file)

FPGA Design Flow

- RTL Synthesis
  - Analyze VHDL
    - Processing -> Analyze Current File
    - Finds syntax errors
    - Does not check for synthesizability
  - Analysis and Elaboration
    - Processing -> Start -> Start Analysis and Elaboration
    - Finds syntax errors
    - Check for synthesizability
    - Creates RTL
      - Check for errors – especially unintended latches



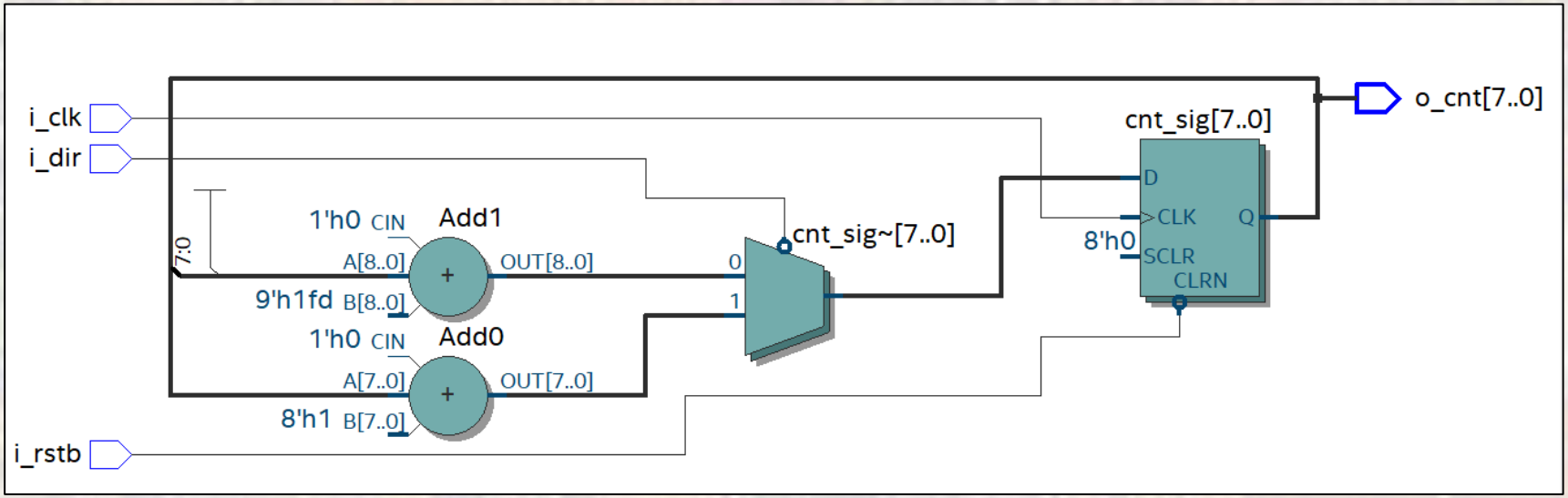
FPGA Design Flow

- RTL Synthesis
  - What is RTL
    - Register Transfer Level
    - Set of design abstractions (primitive elements) and the rules that govern input/output relationships
    - Describes the operation of registers and intermediate logic between registers
    - Abstractions range from NAND/NOR gates through adders/subtractors to memories
    - It is NOT a physical implementation
      - An adder primitive is a mathematical model used to describe the action of addition
      - It is not tied to any circuit implementation

FPGA Design Flow

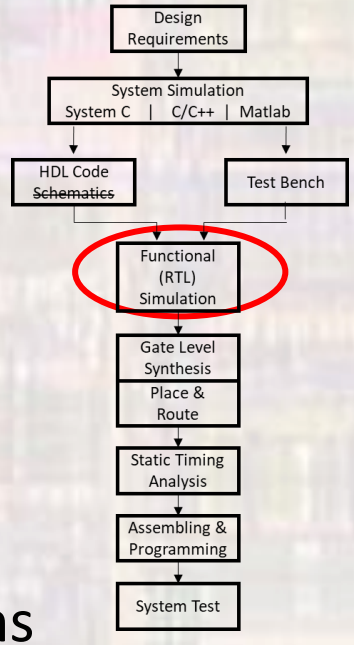
- RTL Synthesis
  - View RTL
    - Tools-> Netlist Viewer -> RTL Viewer
    - Does this make sense?

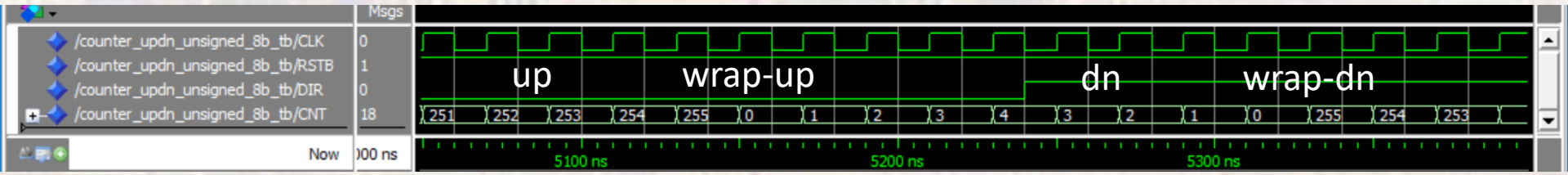
up-down counter



FPGA Design Flow

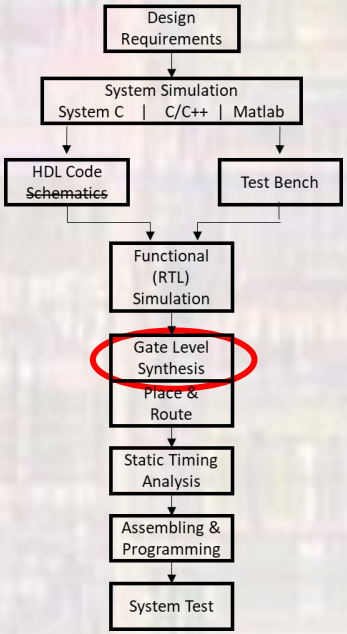
- Functional Simulation
  - ModelSim via Quartus
    - Tools -> Run Simulation Tool-> RTL Simulation
  - ModelSim stand alone – not used by us
  - RTL simulation uses the mathematical abstractions supported by the tool to simulate the actions of the circuit
    - There is NO defined circuit information in these simulations





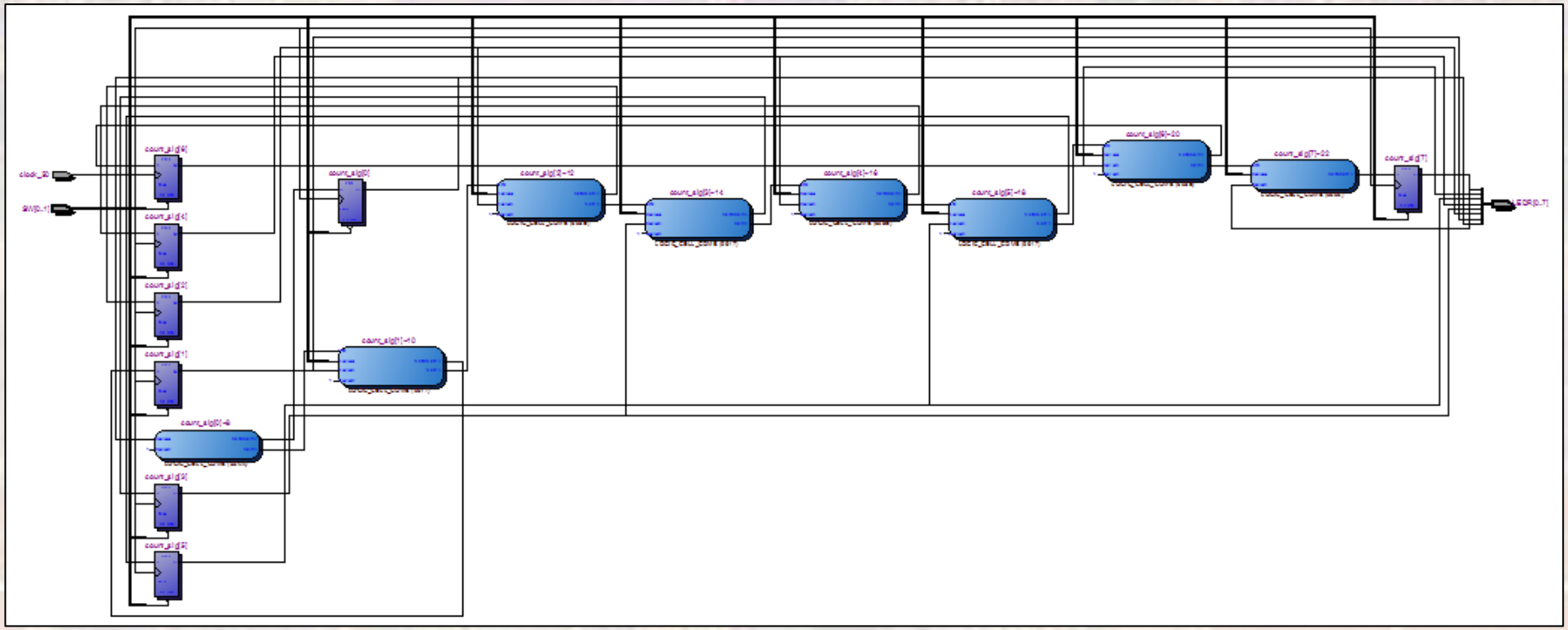
FPGA Design Flow

- Gate Level Implementation
  - Analysis and Synthesis
    - Processing -> Start -> Analysis and Synthesis
    - Maps the RTL to non-specific FPGA blocks
  - Partition and Merge
    - Processing -> Start -> Partition and Merge
    - Allows for incremental synthesis
  - Optional Gate Level Simulation
    - ModelSim via Quartus
    - Tools -> Run Simulation Tool-> Gate Level Simulation
    - New work directory : gate_work



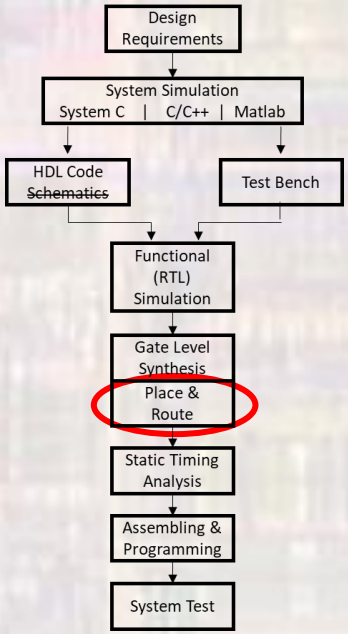
FPGA Design Flow

- Gate Level Implementation
  - Technology Map Viewer – Post Mapping
    - Up/Down Counter
    - Not tied to specific blocks



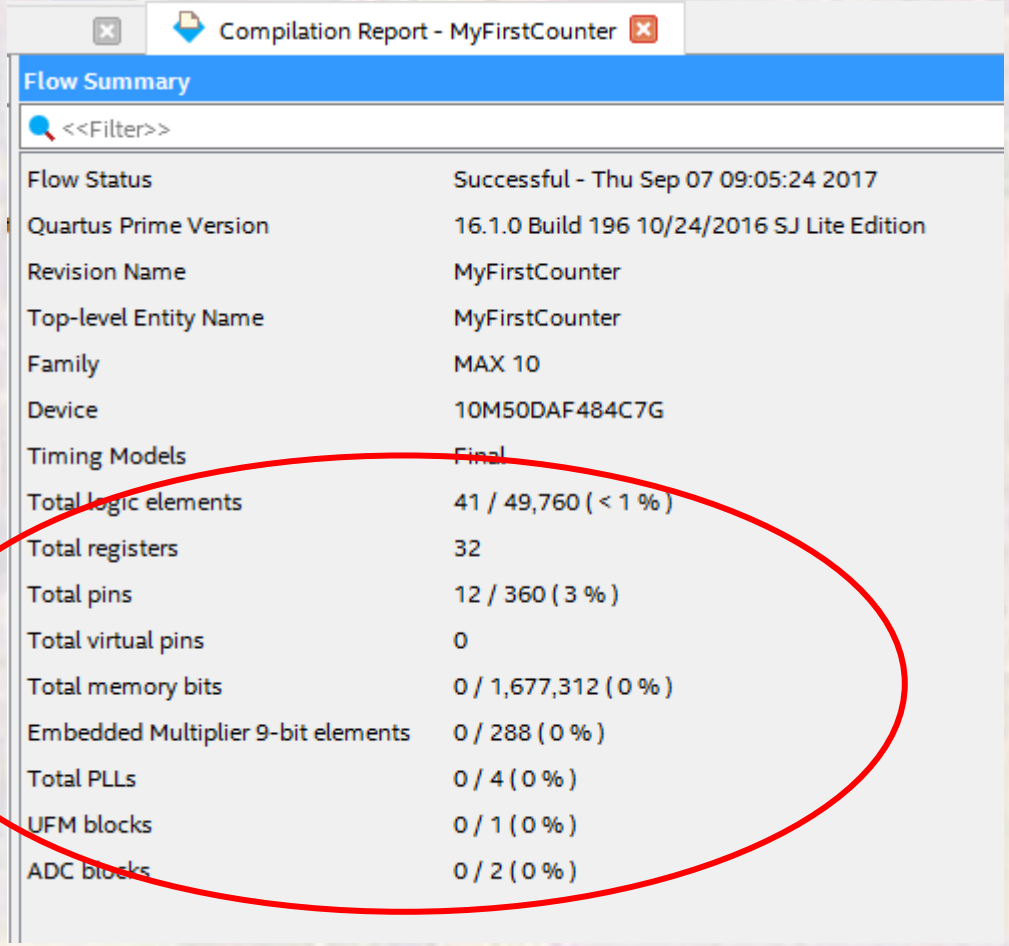
FPGA Design Flow

- FPGA Implementation
  - Timing Constraints
    - Load via TimeQuest
  - Fitter
    - Processing -> Start -> Start Fitter
    - Maps the generalized gate level logic to specific FPGA blocks
    - Accounts for loading and timing constraints
  - Chip Planner
    - Tools -> Chip Planner
    - View the physical implementation
    - Cross Probe via Locate -> Locate in …



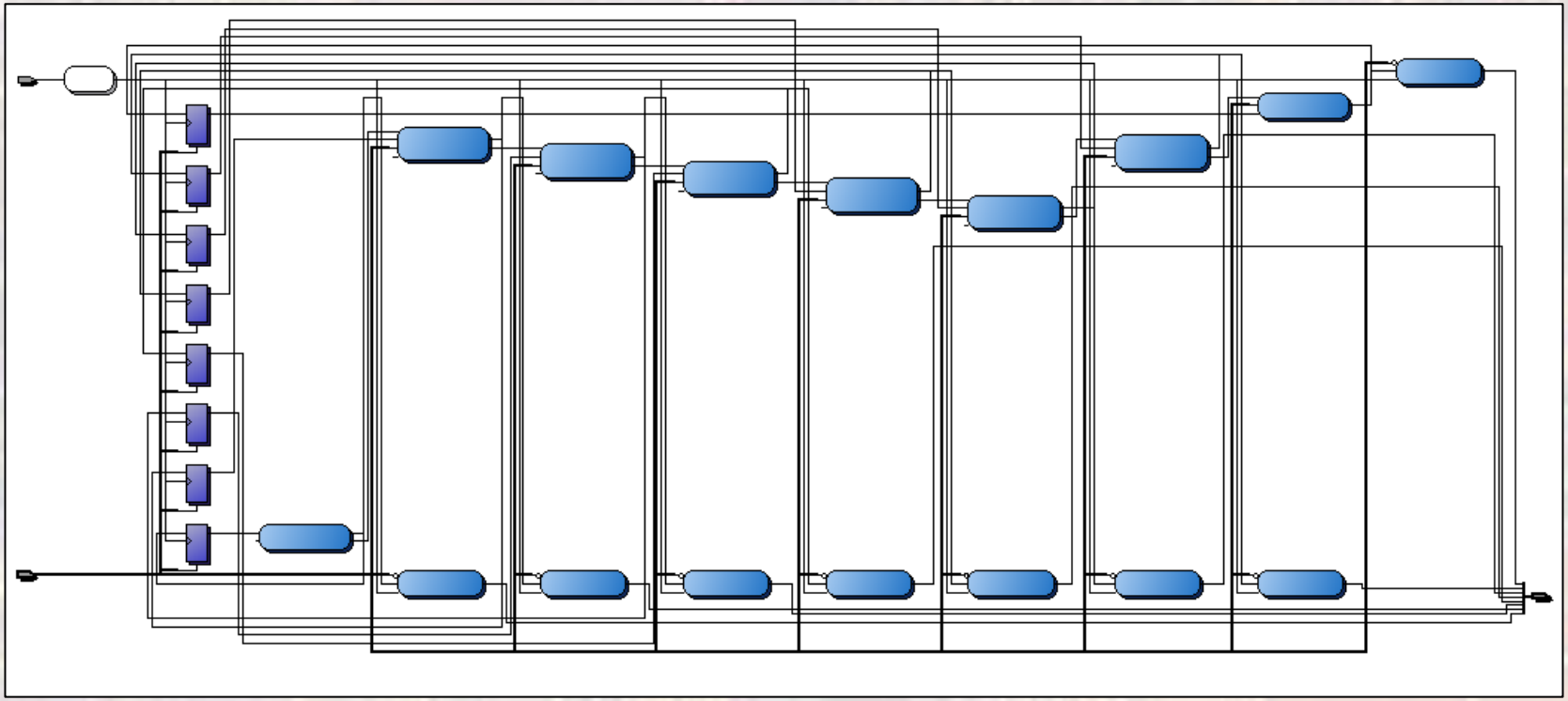
FPGA Design Flow

- FPGA Implementation
  - Fitter – Resource Usage



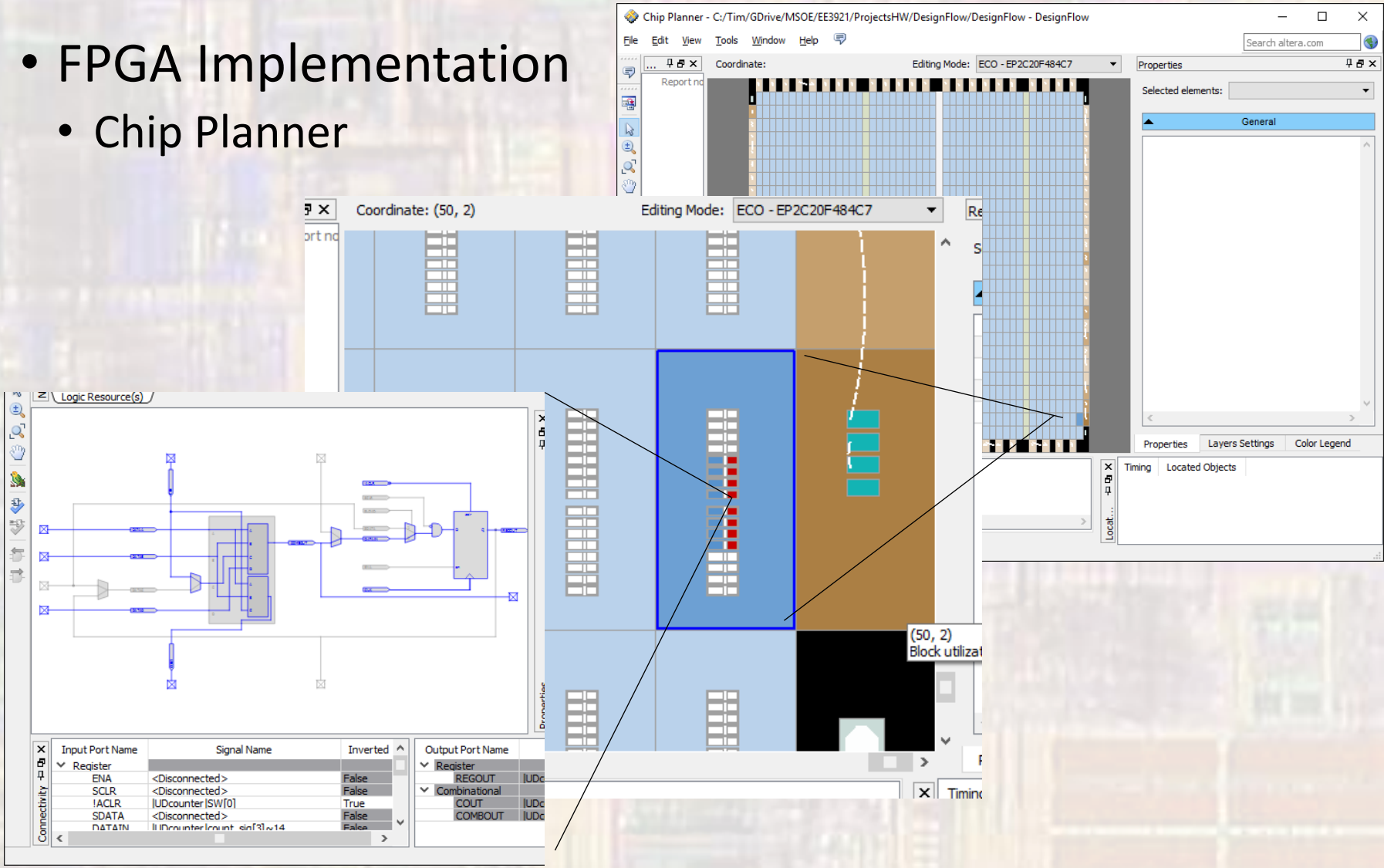
FPGA Design Flow

- FPGA Implementation
  - Technology Map Viewer – Post Fitting
    - Up/Down Counter
    - Tied to specific blocks in the FPGA



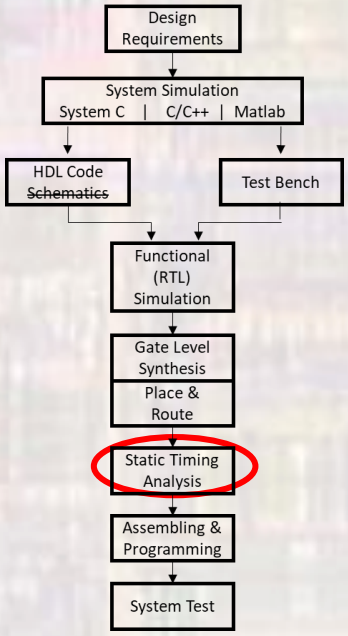
FPGA Design Flow

- FPGA Implementation
  - Chip Planner



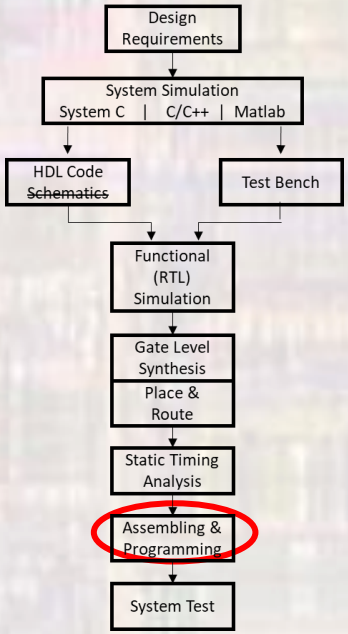
FPGA Design Flow

- FPGA Implementation
  - Static Timing Analysis
    - Processing -> Start -> Start TimeQuest Timing Analyzer
    - Run automatically with the fitter
    - Results are saved in a report file
      - *myDesign*.sta.rpt
    - Runs across process, temperature and voltage variations
      - Fast/typ/slow process
      - High/typ/low temp
      - High/typ/low voltage



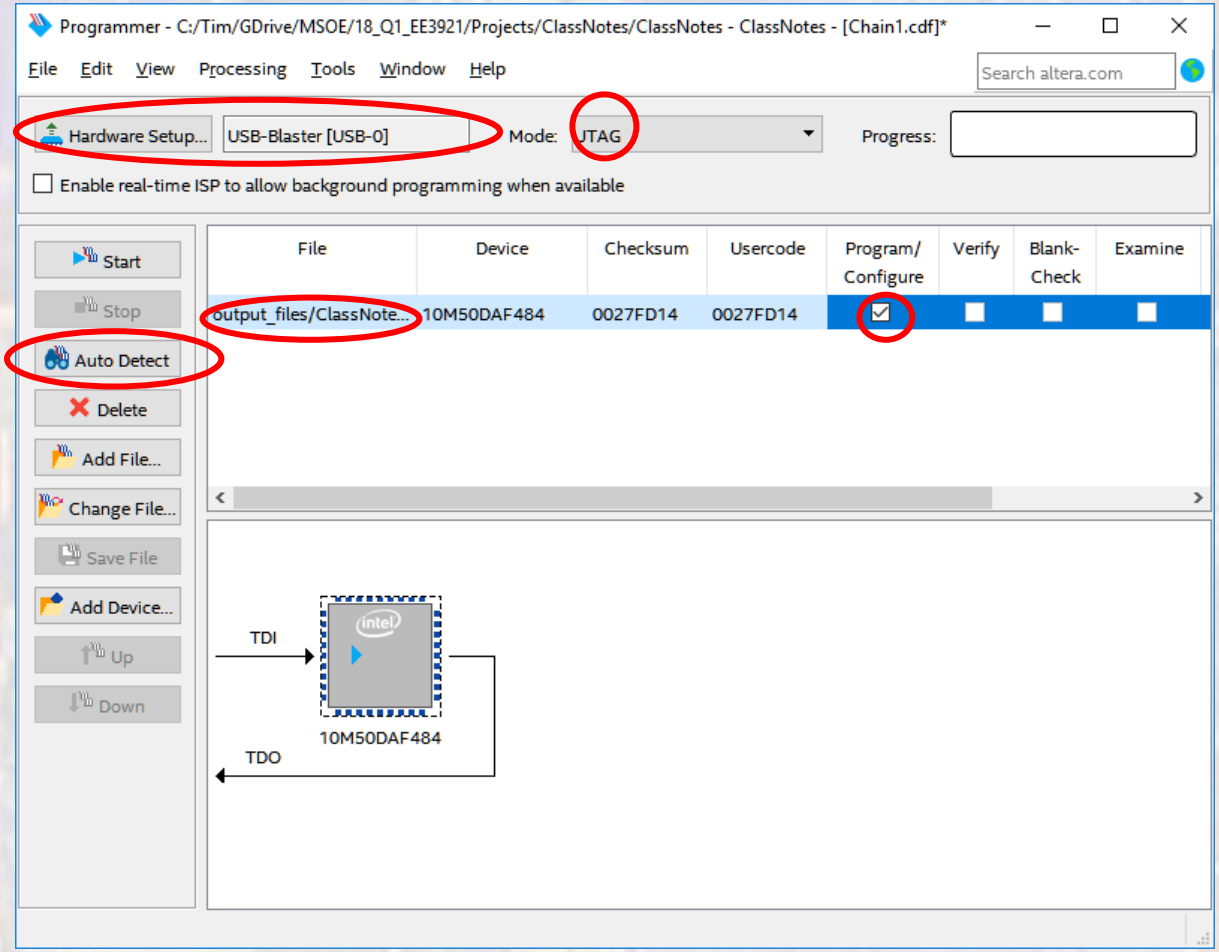
FPGA Design Flow

- FPGA Implementation
  - Assembler
    - Processing -> Start -> Start Assembler
    - Creates the programming file
    - Prepares for additional power analysis
  - Programming
    - Tools -> Programmer



FPGA Design Flow

- FPGA Implementation
  - Programmer



FPGA Design Flow

- FPGA Implementation
  - Programmer