Introduction to CAD Tools

- Design entry and synthesis
- Functional simulation
- Converting button pushes to pulses
- Debouncing buttons
- Implementing circuits on prototype board

Jon Turner
Using CAD Tools to Design Circuits

- Computer-Aided Design (CAD) tools have become essential to the design of digital circuits

- Design entry
  - schematic capture – graphic entry of circuit elements
  - hardware description languages and circuit synthesis
    - two major languages – VHDL and Verilog

- Simulation
  - functional simulation – verify logical correctness
  - timing simulation – verify that performance goals are met

- Timing analysis
  - analysis of delays in circuit components and wiring to verify that timing constraints are satisfied
Simplified CAD Tool Flow

- VHDL source → Syntax check → Synthesis → netlist → Implementation
- Timing Analysis → FPGA bitfile → Device Configuration
- Functional Simulation → VHDL testbench → Timing Simulation
Using/Installing CAD Tools

- CAD tools installed in CEC (start⇒Engineering⇒Xilinx)
  - can be accessed on oasis using Remote Desktop
    - available on both Windows and Mac OS
- To install tools on your own computer
  - versions available for Windows and Red Hat Linux
    - for Mac OS, can use VirtualBox + Windows VM
  - check out Xilinx university program
    - http://www.xilinx.com/university/students/
  - select quick link for free webpack and download
    - www.xilinx.com/support/download/index.htm
  - unpack tarfile and run setup.exe
  - accept the defaults if you can
    - do not install in “Program Files” or any other path with spaces
    - select webpack (this is free)
  - install all the selected tools (including the programming drivers)
Starting New Project

- Start Project Navigator by selecting
  - Start ➔ All Programs
    ➔ Xilinx
    ➔ ISE Design Tools
    ➔ Project Navigator

Specify
name and
location

Specify
Spartan3E
XC3S500E
FG320
-4

VHDL
ISim
Creating VHDL Source File

Project ➞ New Source

Select VHDL Module

enter name

File name:
Location:
Entering VHDL Source and Checking

- Select Implementation in mode menu
- VHDL editor
- Syntax check
- Error Messages
Preparing to Simulate

1. Open the project.
2. Click on "New Source".
3. Enter the name for the testbench.
4. Select the VHDL testbench.
5. Add the testbench to the project.
entity testCalculator is end testCalculator;
architecture a1 of testCalculator is begin
  uut: calculator port map(clk, clear, load, add, dIn, result);
  process begin -- clock process
    clk <= '0'; wait for 10 ns; clk <= '1'; wait for 10 ns;
  end process;
  tb : process begin
    clear <= '1'; load <= '1'; add <= '1'; dIn <= x"fffff";
    clear <= '0'; load <= '1'; add <= '0'; dIn <= x"fffff"; wait for 20 ns;
    clear <= '0'; load <= '1'; add <= '1'; dIn <= x"fffff"; wait for 20 ns;
    clear <= '0'; load <= '0'; add <= '1'; dIn <= x"0001"; wait for 20 ns;
    ...
    clear <= '0'; load <= '0'; add <= '1'; dIn <= x"0500"; wait for 20 ns;
    wait for 20 ns;
    assert (false) report "Simulation ended normally." severity failure;
  end process;
end;
Starting Simulation

- select Simulation mode, then behavioral
- select testbench file
- double-click here to start simulation
Running Simulation

- Simulator command window
- Float waveform window
- Zoom controls
- Restart and Run - all buttons
- Use to add signals to waveform window
- Waveform window
- Labeled divider
- Marker
- Waveforms appear here
Using Simulator Effectively

- Use appropriate radix for multi-signal waveforms
  - right-click on signal in waveform window
- Organize signals into logical groups
  - insert labeled “dividers” to highlight groups
- Use markers to better see timing relationships
- Saving waveform window format
  - use “Save As” in waveform window
  - then open the file at start of subsequent simulation runs
- Why bother?
  - save yourself debugging time
  - make it easier for TAs to see that you got it right
Implementing Circuit on Proto Board

- Use knobs and buttons to control calculator inputs
- Use LCD display to show input and result
- Requires two “interface circuits”
  - input module produces 16 data bits, an internal reset signal, 3 debounced button signals and 3 pulse signals
  - output module shows two 16 bit values on LCD display
Top Level Circuit

- Instantiates calculator circuit, input and output modules and connects them
- Uses dBtn(1) for clear signal
  - mechanical buttons vibrate when making contact
    - can lead to several apparent transitions on button press
  - input module includes a debouncer circuit that filters out transitions that last less than about a millisecond
  - ensures “clean transitions” that work as we expect
- Uses pulse(2) and pulse(3) for load and add
  - pulse signals are derived from debounced button signals but are only high for one clock tick
  - using pulse ensures that button press causes single add
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use work.commonDefs.all;

entity top is port(
  clk: in std_logic;
  btn: in buttons;
  knob: in knobSigs;
  lcd: out lcdSigs);
end top;

architecture al of top is
component calculator ... end component;
component binaryInputModule... end component;
component binaryInputModule... end component;

include commonDefs package to define constants and types

top level inputs and outputs for buttons, knob and LCD

component declarations for calculator input and output modules
signal reset, clear, load, add : std_logic;
signal inBits, outBits : word;
signal dBn, pulse : std_logic_vector(3 downto 1);

begin
  -- connect the sub-components
  imod: binaryInMod port map(clk, btn, knob, reset, dBn, pulse, inBits);
  calc: calculator port map(clk, clear, load, add, inBits, outBits);
  omod: binaryOutMod port map(clk, reset, inBits, outBits, lcd);

  -- define internal control signals
  clear <= dBn(1) or reset;
  load <= pulse(2);
  add <= pulse(3);
end ai;
Common Definitions

library IEEE;

package commonDefs is

  constant wordsize: integer := 16; -- number of bits per "word"
  constant nBtn: integer := 4;   -- number of buttons
  constant nKsigs: integer := 3; -- number of knob signals
  constant nSwt: integer := 4;   -- number of switches
  constant nLED: integer := 8;   -- number of LEDs

  subtype word is std_logic_vector(wordSize-1 downto 0);
  subtype buttons is std_logic_vector(nBtn-1 downto 0);
  subtype knobSigs is std_logic_vector(nKsigs-1 downto 0);
  subtype switches is std_logic_vector(nSwt-1 downto 0);
  subtype leds is std_logic_vector(nLED-1 downto 0);

  type lcdSigs is record
    en, rs, rw, sf_CE: std_logic;
    data: std_logic_vector(3 downto 0);
  end record;

  constant operationMode: integer := 0; -- 0 for sim, 1 for proto board
end package commonDefs;
Simulation of Prototype Board Version

- btn(0) generates reset signal
- knob rotation increments input value
- knob press plus rotation adds 16 to input value
- btn(2) generates load signal
- load pulse loads input value
- add pulses trigger additions
Simplified CAD Tool Flow

1. VHDL source
   - Synthesis
     - Syntax check
     - Netlist
   - Implementation
     - Timing Analysis
       - FPGA bitfile
     - Device Configuration
   - Behavioral Simulation
     - VHDL testbench
   - Timing Simulation
User Constraints File

- The ucf file defines mapping of device pins to inputs and outputs of top-level VHDL entity
  ```
  NET "btn<0>" LOC = "V4";
  NET "btn<1>" LOC = "H14";
  ...
  NET "led<0>" LOC = "F12";
  NET "led<1>" LOC = "E12";
  ...
  NET "clk" LOC = "C9";
  NET "swt<0>" LOC = "L13";
  NET "swt<1>" LOC = "L14";
  
  Also is used to define target clock frequency
  ```
  ```
  NET "clk" TNM_NET = "clk";
  TIMESPEC "TS_clk" = PERIOD "clk" 20 ns HIGH 50 %;
  
  - Add as source file to your project
    » you will find ucf file for prototpe board on web site
Set Synthesis Properties

right-click on Synthesize and select Process Properties

recommended settings include Optimize Area, Keep Hierarchy

Advanced Display Level
Synthesizing Circuit

- Double-click on Synthesize in Project Navigator
- Review synthesis report (access thru design summary)
  - provides information about resource usage, performance
- For calculator alone, device configuration part is

<table>
<thead>
<tr>
<th>Component</th>
<th>Quantity</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>9</td>
<td>4656</td>
<td>0%</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>16</td>
<td>9312</td>
<td>0%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>17</td>
<td>9312</td>
<td>0%</td>
</tr>
<tr>
<td>Number of IOs</td>
<td>36</td>
<td>232</td>
<td>15%</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>36</td>
<td>232</td>
<td>15%</td>
</tr>
<tr>
<td>IOB Flip Flops</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of GCLKs</td>
<td>1</td>
<td>24</td>
<td>4%</td>
</tr>
</tbody>
</table>

- Timing summary

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum period</td>
<td>4.152ns</td>
</tr>
<tr>
<td>Maximum Frequency</td>
<td>240.848MHz</td>
</tr>
<tr>
<td>Minimum input arrival time</td>
<td>5.550ns</td>
</tr>
<tr>
<td>Maximum output required time</td>
<td>4.283ns</td>
</tr>
<tr>
<td>Maximum combinational path delay</td>
<td>No path found</td>
</tr>
</tbody>
</table>

Acceptable, since prototype board has 50 MHz clock.
Setting Implementation Properties

right-click on Implement Design and select Process Properties

check Allow Unmatched LOC Constraints
Transferring Circuit to Proto Board

- Implementation step determines how to map design to FPGA and route all connections
  - in Project Navigator, double-click on “Implement Design”

- Next step is to generate bitfile that can be downloaded to prototype board
  - in Project Navigator process window, double-click on “Generate Programming File”
  - creates file with .bit extension (e.g. top.bit)

- Transfer file to PC in lab (using thumb drive)

- Use Impact tool (in Accessories sub-menu) to transfer bit file to prototype board
  - first make sure board is connected to PC’s USB port
Initialize Prototype Board

- First double-click on Boundary Scan
- Then right-click and select Initialize chain
- Right click on FPGA icon and select Assign New Configuration File; when prompted, specify top.bit
Transfer Bit File to Prototype Board

- Right-click on icon and select Program
- Click OK