Tutorial 2: Methodology for Design Analysis

Dr. Paul D. Franzon

Outline

1. Design Evaluation thoughts
2. NCSU methodology for analysis of designs (without memory)
Design Evaluation

1. Aim
   1. Design Evaluation as Power (P), Area (A), Critical Path Delay (D)
   2. A weighted combination of two or more of these parameters used as constraint for optimization. AD, PD, PD^2, αP + βD + γA, etc.

2. Considerations
   1. Synthesis provides PAD values without wiring considerations
   2. Critical wiring capacitance values missing: fanouts, wire lengths
      
   \[ P = αC V^2 f \]
   \[ D_{inv} \sim R_s (C_{int} + C_L) \]
   \[ C_L \text{ includes wire capacitance} \]

   • Needs:
     1. Prototyping flow for determining layout aware performance in quick time

1. Other thoughts
   1. Implementability of design not evaluated (ECE761 for understanding this)
      1. Distribution of number of inputs and outputs
      2. Relative sizes of design units
      3. Routability of wires etc etc etc..
NCSU Design Prototyping Flow (Tools)

module counter (clock, in, latch, dec, zero);

// Simple down counter with zero flag
input clock; /* clock */
input [3:0] in;  /* starting count */
input latch;  /* latch `in' when high */
input dec;    /* decrement count when dec high */
output zero;   /* high when count down to zero */
output [3:0] value;  /* current count value */
reg zero; wire [3:0] value_minus1; reg [3:0] mux_out;

always@(posedge clock)
begin
    value <= mux_out;
    value_minus1 <= value - 1'b1; // combinational logic for zero flag
    assign zero = ~|value;
end

endmodule /* counter */

Student Does Manual exploration to determine best RTL for specification
NCSU Design Prototyping Flow (Tools)

module counter (clock, in, latch, dec, zero);
    input [3:0] in;
    input clock, latch, dec;
    output zero;
    wire sub_42_A_0_, sub_42_A_1_, sub_42_A_2_, sub_42_A_3_, n33, n34, n35,
    n36, n37, n38, n39, n40, n41, n42, n43, n44, n45, n46, n47, n48, n49,
    n50, n51, n52, n53, n54, n55, n56, n57, n58;

    DFFPOSX1 value_reg_0_ (.D(n58), .CLK(clock), .Q(sub_42_A_0_));
    DFFPOSX1 value_reg_1_ (.D(n57), .CLK(clock), .Q(sub_42_A_1_));
    DFFPOSX1 value_reg_2_ (.D(n54), .CLK(clock), .Q(sub_42_A_2_));
    DFFPOSX1 value_reg_3_ (.D(n51), .CLK(clock), .Q(sub_42_A_3_));

    INVX1 u3 (.A(n33), .Y(zero));
    OAI21X1 u4 (.A(latch), .B(n43), .C(n38), .Y(n50));
    NAND2X1 u5 (.A(latch), .B(in[2]), .Y(n35));
    AOI22X1 u6 (.A(sub_42_A_2_), .B(n36), .C(n56), .D(n37), .Y(n34));
    INVX1 u7 (.A(n38), .Y(n36));
    OAI21X1 u8 (.A(n39), .B(n40), .C(n41), .Y(n51));
    NAND2X1 u9 (.A(in[3]), .B(latch), .Y(n41));
    OR2X1 u10 (.A(n37), .B(latch), .Y(n40));
    AND2X1 u11 (.A(dec), .B(n42), .Y(n37));
    OAI21X1 u12 (.A(latch), .B(n43), .C(n35), .Y(n52));
    NAND2X1 u13 (.A(in[1]), .B(latch), .Y(n44));
    AOI22X1 u14 (.A(sub_42_A_1_), .B(n45), .C(n38), .Y(n43));
    NOR2X1 u15 (.A(n45), .B(sub_42_A_2_), .Y(n38));
    INVX1 u16 (.A(n46), .Y(n45));
    OAI21X1 u17 (.A(latch), .B(n47), .C(n48), .Y(n53));
    NAND2X1 u18 (.A(in[0]), .B(latch), .Y(n48));
    AOI21X1 u19 (.A(sub_42_A_0_), .B(n49), .C(n46), .Y(n47));
    NOR2X1 u20 (.A(n49), .B(sub_42_A_3_), .Y(n46));
    NAND2X1 u21 (.A(dec), .B(n33), .Y(n49));
    NAND2X1 u22 (.A(n42), .B(n39), .Y(n33));
    INVX1 u23 (.A(n56), .Y(n39));
    NOR3X1 u24 (.A(sub_42_A_1_), .B(sub_42_A_2_), .C(n42), .Y(n42));
    BUF4 u25 (.A(n50), .Y(n54));
    INVX8 u26 (.A(sub_42_A_3_), .Y(n55));
    INVX1 u27 (.A(n55), .Y(n56));
    BUF2 u28 (.A(n52), .Y(n57));
    BUF2 u29 (.A(n53), .Y(n58));

endmodule
NCSU Design Prototyping Flow (Tools)

Power Grid Addition (Encounter)

Tutorial 2
Part A

Verilog Testbench
Verilog RTL
Design Compiler
Pre-Place and Route Netlist
Cadence Encounter (Layout Created)

Post-Place and Route Netlist
Standard Parasitic Exchange Format (SPEF)

© 2007, Dr. Paul D. Franzon, www.ece.ncsu.edu/erl/faculty/paulf.html
NCSU Design Prototyping Flow (Tools)

Standard Cell PLACEMENT (Encounter)

Tutorial 2
Part A
NCSU Design Prototyping Flow (Tools)

ROUTE  Wires + Clock Tree Synthesis =>
Capacitance Determined (Encounter)

Tutorial 2
Part A
NCSU Design Prototyping Flow (Tools)

Create $\alpha$ in $P = \alpha CV^2f$ (Modelsim)
NCSU Design Prototyping Flow (Tools)

Tutorial 2
Part C

Compute P and D (Design Compiler)
ECE 520 Design Prototyping Tool Flow

1. Impose constraints on directory structure
2. Make generic assumptions for running place and route for all designs
   1. Always aiming for square chip
   2. Area for Placing and Routing Design = $x^2 \sim (Logic\ Area / 0.90)$
3. User controls only a few command line inputs to the automation
4. 3 Step process for analysis of any design
   1. Part A (Place and Route) AUTOMATED
   2. Part B (SAIF Creation) MANUAL
   3. Part C (Power & Delay Calculation) AUTOMATED
5. Automation Script: PAD_Flow.pl
   1. Setup of directory structure (./PAD_Flow.pl -op setup)
   2. Clean up of all contents of directory (./PAD_Flow.pl -op clean)
   3. Run analysis (./PAD_Flow.pl -op Power OR ./PAD_Flow.pl -op Analyze)
ECE 520 Design Prototyping Tool Flow (Chronology)


   .PAD_Flow
   .HDL/ .SIMULATION/ .SYNTH/ .PR/
   ./common/ .run_s/ .run_s/ .run_f/ .run_s/ .run_f/ .run_f/

   1. **Green (s):** User directories
   2. **Red (f):** Flow Directories (STAY AWAY!!)

2. ./HDL/run_s/ : All the HDL files for your design
   1. Create sub-directories here for design versions / modularity

3. Copy
   1. modelsim.ini to ./SIMULATION/run_f and ./SIMULATION/run_s
   2. Library_fwd.saif to ./SIMULATION/run_f
   3. Constraints.tcl to ./SYNTH/run_f (Different from Tutorial 1)

4. At this point all that you need to start your design creation process and its evaluation is in places
ECE 520 Design Prototyping Tool Flow (Chronology)

1. Converge to design of choice by using Tutorial 1 Simulation Flow
   1. You are in ./SIMULATION/run_s. Create Test Test.v here.
   2. Work in ./SIMULATION/run_s and compile as
      1. vlog ../HDL/run_s/counter.v ... (MORE FILES IF ANY)
      2. vlog ./Test.v
   3. Make modifications and repeat until design is satisfactory
   4. Copy final testbench over to ../SIMULATION/run_f

2. Synthesize the design of choice by using Tutorial 1 Synthesis Flow
   1. You are in ./SYNTH/run_s. Copy all the tcl files from Tutorial 1 here.
   2. Copy .synopsys_dc.setup file over.
   3. Modify setup.tcl to make RTL_DIR ../HDL/run_s/
   4. SYNTHESIZE
   5. Make RTL modifications and repeat until design is satisfactory
   6. Copy final netlist over to ../SYNTH/run_f/

Results (w.r.t top directory)
   1. Synthesized netlist in ./SYNTH/run_f/
   2. Testbench of choice in ./SIMULATION/run_f/
ECE 520 Design Prototyping Tool Flow (Chronology)

- Pre-Place and Route Netlist
  - Synopsys Design Compiler
  - Cadence Encounter Place and Route TRIALROUTE
    - Core Area = A_initial/0.90
      - AUTOMATED
        - Partitioning
        - Power Planning
        - Placement
        - Clock Tree Synthesis
        - Routing
        - RC Analysis
      - Post-Place and Route Netlist
  - Standard Parasitic Exchange Format (SPEF)
  - Verilog Testbench
  - Mentor ModelSim Functional Simulation with RTL (Check Correctness)
  - Synopsys Design Compiler
  - Pre-Place and Route Netlist
  - Verilog RTL
  - STUDENT SIDE
    - MANUAL EXPLORATION
      - Core Area = A_initial/0.90
        - AUTOMATED
          - Modify Testbench to add commands to collect switching activity
          - Verilog Testbench for SAIF creation
          - Mentor ModelSim Simulation with netlist (SAIF creation)
          - Switching Activity Interchange Format (SAIF)
          - Post-Place and Route Netlist
          - Standard Parasitic Exchange Format (SPEF)
          - Synopsys Design Compiler for power and timing
            - PAD_Flow.pl
            - PAD_Flow.pl
            - AUTOMATED
              - (PAD) final
              - STUDENT SIDE
                - MANUAL EXPLORATION
                  - (PAD) initial
**ECE 520 Class Notes**

**ECE 520 Design Prototyping Tool Flow (Chronology)**

Setup all relevant tools (assuming you have not already done so):

```plaintext
add cadence2005; add modelsim; add synopsys
```

1. **Automated Place and Route command**
   
   ```plaintext
   ./PAD_Flow.pl -mod counter -clkname clock -period 10 -op analyze
   -net ./SYNTH/run_f/counter_final.v
   ```

   **Command Line options**
   
   - `mod`: Name of top level module
   - `clkname`: name of clock in your design
   - `period`: clock period in ns
   - `net`: full path to netlist
   - `op`: Analyze/Power/Clean/Setup : Here “analyze”

   **Result:** Encounter runs (in ./PR/run_f/) to create
   
   ```plaintext
   <modname>_routed.v (here counter_routed.v)
   <modname>.spef (here counter.spef)
   ```
**ECE 520 Design Prototyping Tool Flow (Chronology)**

1. **Verilog Testbench**
2. **Modify Testbench to add commands to collect switching activity**
3. **Verilog Testbench for SAIF creation**
4. **Mentor Modelsim Simulation with netlist (SAIF creation)**
5. **Switching Activity Interchange Format (SAIF)**
6. **Post- Place and Route Netlist**
7. **Synopsys Design Compiler**
8. **Synopsys Functional Simulation with RTL (Chk Correctness)**
9. **Pre- Place and Route Netlist**
10. **PAD\_flow\_pl Place and Route Netlist**
11. **Cadence Encounter Place and Route TRIALROUTE**
12. **Standard Parasitic Exchange Format (SPEF)**
13. **Synopsys Design Compiler for power and timing**
14. **PAD\_flow\_pl AUTOMATED (PAD)\_final**

**STUDENT SIDE**

**MANUAL EXPLORATION**

- Verilog Testbench
- Modify Testbench to add commands to collect switching activity
- Verilog Testbench for SAIF creation
- Mentor Modelsim Simulation with netlist (SAIF creation)
- Switching Activity Interchange Format (SAIF)
- Post- Place and Route Netlist
ECE 520 Design Prototyping Tool Flow (Chronology)

2. Manual Switching Activity File creation
   1. Work in ./SIMULATION/run_f/

Input
   1. Netlist (here counter_routed.v) in ./PR/run_f/
   2. Testbench (here test.v) in ./SIMULATION/run_f/

Testbench modification
   1. $read_lib_saif("./Library_fwd.sai");
   2. $set_toggle_region(test_fixture.DUT);
      Dependent on testbench and DUT instance names
   3. $toggle_start();
   4. $toggle_stop();
   5. $toggle_report("./counter_back.sai", 1.0e-9,"test_fixture.DUT");
      Dependent on testbench and DUT instance names, Name: <modname>_back.sai
ECE 520 Design Prototyping Tool Flow (Chronology)

Compilation
1. vlog ../../PR/run_f/<modname>_routed.v
2. vlog /afs/bp.ncsu.edu/dist/cadence_cdk/OSU018_StdCells/Typical/osu018_stdcells.v
3. vlog ./<testbench_with_toggles>.v

Simulation
• vsim -novopt -c -pli $SYNOPSYS/sparcOS5/power/vpower/libvpower.so test_fixture
• vsim -novopt -c -pli $SYNOPSYS/sparc64/power/vpower/libvpower.so test_fixture
• vsim -novopt -c -pli $SYNOPSYS/amd64/power/vpower/libvpower.so test_fixture
• vsim -novopt -c -pli $SYNOPSYS/linux/power/vpower/libvpower.so test_fixture

Choose one of the above based on system type (linux should work in most cases)

Modify with each design

Result: Modelsim runs (in ./SIMULATION/run_f/) to create
1. <modname>_back.saif (here counter_back.saif)
ECE 520 Design Prototyping Tool Flow (Chronology)

1. **Pre-Place and Route Netlist**
   - **Standard Parasitic Exchange Format (SPEF)**
   - **Synopsys Design Compiler for power and timing**
   - **PAD_Flow.pl**
   - **(PAD)final**

2. **Mentor Modelsim Functional Simulation with RTL (Chk Correctness)**
   - **Verilog Testbench**
   - **Mentor Modelsim Simulation with netlist**
   - **Switching Activity Interchange Format (SAIF)**

3. **Synopsys Design Compiler**
   - **Verilog RTL**
   - **Post-Place and Route Netlist**

4. **Core Area**
   - **AUTOMATED**
   - **Cadence Encounter Place and Route TRIALROUTE**
   - **PAD_Flow.pl**
   - **(PAD)final**

5. **Student Side Manual Exploration**
   - **Verilog Testbench for SAIF creation**
   - **Synopsys Design Compiler**
   - **Post-Place and Route Netlist**
   - **Standard Parasitic Exchange Format (SPEF)**

© 2007, Dr. Paul D. Franzon, www.ece.ncsu.edu/erl/faculty/paulf.html
ECE 520 Design Prototyping Tool Flow (Chronology)

3. Automated Power Computation

`.PAD_Flow.pl -mod counter -clkname clock -period 10 -op power
-saif ./SIMULATION/run_f/counter_back.saif -inst test_fixture/DUT`

Command Line options

1. `-mod, -clkname, -period` already explained
2. `-op`: Here “Power”
3. `-saif`: Full path to the backward saif file
4. `-inst`: Hierarchy to the block of interest from top of testbench. Make sure you use “/”

Result:

1. Power and Delay values from power and timing reports in `.SYNTH/run_f/*.rpt`
2. Area = (Synthesis Area)/0.90
Concluding Remarks

1. Notes:
   1. PAD_Flow.pl is run on the terminal and at the top working directory
      1. Essentially a means of calling all the tools from command prompt
   2. Student does Simulation and Synthesis in run_s folders;
   3. Flow works in run_f folders
   4. There are two different Constraints.tcl files. (Will RENAME ONE)
   5. Please use replace_synthetic -ungroup when you do your Synthesis
   6. Copy .synopsys_dc.setup file over when you synthesize.

2. Changes for each design (most important)
   1. Synthesis results vary based on module names.
   2. Testbench names vary based on user preference.
   3. Input for SPEF creation
      a) Netlist name (-net <>);  b) Mod name (-mod <>)
   4. Simulation:
      a) Name of test fixture  b) Backward SAIF file name
   5. Input Power results
      a) Name of the backward SAIF file (-back <>)  b) instance name (-inst <>)
      c) mod name (-mod <>)