FPGA OPTIMIZATIONS

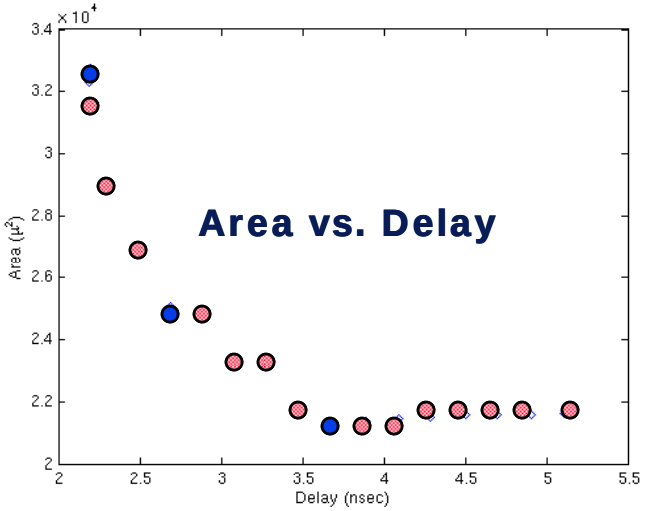
Prof. Don Bouldin, Ph.D.

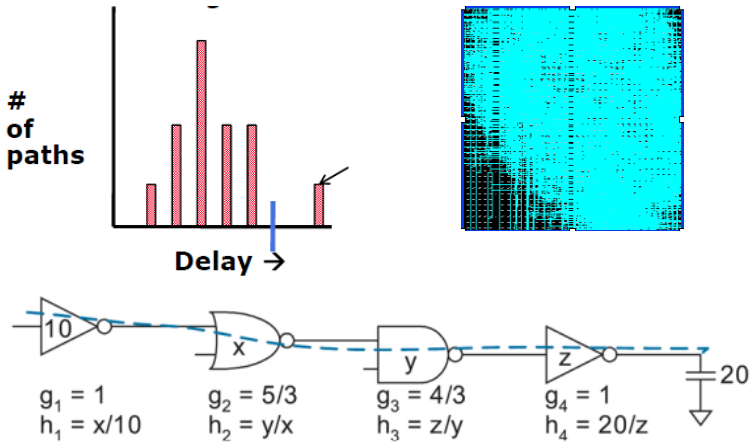
Electrical & Computer Engineering

University of Tennessee

Knoxville, TN 37996-2100

dbouldin@tennessee.edu





SPECIFYING SYNTHESIS-LEVEL OPTIMIZATIONS

- General Tips
- Pipelining
- Retiming
- Preserving Objects
- Optimizing Fanout
- Sharing Resources
- Inserting I/Os and Probes
- Optimizing State Machines
- Working with Gated Clocks

GENERAL TIPS FOR SYNTHESIS

1. Use *CASE* statements for priority-independent logic instead of nested *IF-THEN-ELSE* statements.
2. Use the *syn_encoding* attribute with a value of *safe to* ensure that the synthesized state machine never locks in an illegal state.
3. For FSMs coded in VHDL using enumerated types, use the same encoding style (*syn_enum_encoding* attribute value) on both the state machine enumerated type and the state signal.
4. To enable hierarchical optimization on your design, set the *syn_hier* attribute to *firm*.
5. For accurate results with timing-driven synthesis, explicitly define clock frequencies with a constraint, instead of using a global clock frequency.

SYNTHESIZING FOR AREA

1. Increase the *fanout limit* when you set the implementation options to achieve a smaller area.
2. Check the *Resource Sharing* option when you set implementation options to share hardware resources like adders, multipliers, and counters to minimize area.
3. For designs with large FSMs, use the *gray* or *sequential encoding* styles.
4. Set the encoding style for FSMs to *sequential* instead of *onehot* to minimize area.
5. For small designs (less than 20K gates), you might improve area by using the *syn_hier* attribute with a value of *flatten*. When specified, the software optimizes across hierarchical boundaries and creates smaller designs.

SYNTHESIZING FOR TIMING

1. Use *realistic* design constraints for clocks and paths.
2. Select a balanced fanout constraint. A large constraint creates nets with large fanouts, and a low fanout constraint results in replicated logic.
3. If the critical path goes through arithmetic components, try disabling *Resource Sharing* to get faster times at the expense of increased area.
4. If the P&R and synthesis tools report different critical paths, use a timing constraint with the *route* option. This option adds route delay to its calculations when trying to meet the clock frequency goal.
5. For FSMs, use the *onehot* encoding style, because it is often the fastest implementation. If a large output decoder follows an FSM, *gray* or *sequential* encoding could be faster.
6. For designs with black boxes, characterize the timing models accurately, using the *syn_tpd*, *syn_tco*, and *syn_tso* directives.
7. Make sure that you pass your timing constraints to the P&R tools, so that they can use the constraints to optimize timing.

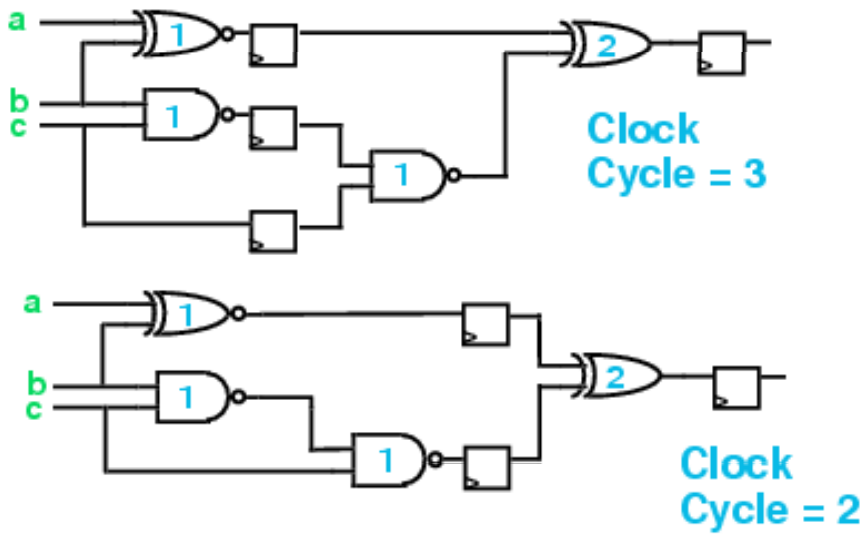
PIPELINING

- Pipelining is the process of splitting logic into stages so that the first stage can begin processing new inputs while the last stage is finishing the previous inputs.
- This ensures better throughput and faster circuit performance. You can enable or disable this feature.

Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

RETIMING

- **Retiming or register balancing automatically moves edge-triggered registers across combinatorial gates or LUTs to improve timing while ensuring identical behavior as seen from the primary inputs and outputs of the design. You can enable or disable this feature.**



PRESERVING OBJECTS

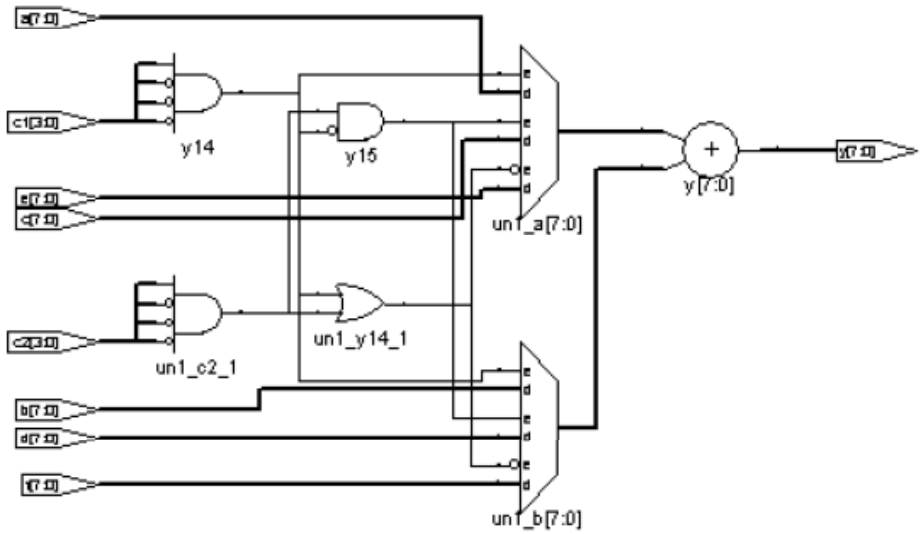
- Synthesis can collapse or remove objects (nets, registers, or other components) during optimization.
- To preserve an object whose area or timing you like, specify the *syn_keep* attribute.
- Otherwise, the synthesis tool will automatically resynthesize any module whose timestamp has been modified.
- Use incremental synthesis to instantiate multiple copies of a previously generated module with the *syn_keep* attribute.

OPTIMIZING FANOUT

- Fanout limits can be set to restrict the timing on critical nets by setting the *syn_maxfan* attribute on a port, net, register or primitive instance.
- Fanouts that exceed the hard limit are buffered or replicated.
- Replication is used on nets driven by registers or combinatorial logic by duplicating the net driver and splitting the net into segments. This increases the number of register bits in the design.
- When replication is not possible, the synthesis tool buffers signals. Buffering is more expensive in terms of intrinsic delay and resource consumption but is automatically used on input ports.

RESOURCE SHARING

- Resources can be shared by using the same arithmetic operators for mutually exclusive statements such as branches of a case statement.
- Conversely, you can improve timing by disabling resource sharing, but at the expense of increased area.
- Specify the *syn_sharing* attribute.



INSERTING I/O AND PROBES

- To preserve any unconnected I/O buffers from being optimized away, specify the *syn_force_pads* attribute in your source code.
- Alternatively, select *Implementation Options* and leave the *Disable I/O Insertion* checkbox empty (disabled) to insert I/O pads automatically for all the inputs, outputs and bidirectionals.

- To specify simulation probes in the source code, add the *syn_probe* attribute to the net.

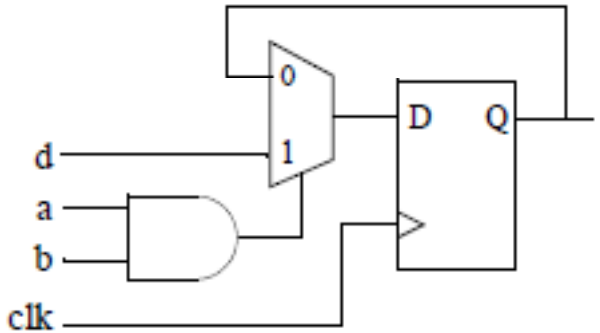
```
reg [7:0] alu_tmp /* synthesis syn_probe=1 */;
```

OPTIMIZING STATE MACHINES

- Use *FSM Compiler* to generate better results since the tool uses advanced optimization techniques like reachability analysis.
- *FSM Compiler* also lets you convert an encoded state machine to another encoding style (to improve speed and area utilization) without changing the source. For example, you can use a *onehot* style to improve results.
- If you do not want to optimize a state machine, add the *syn_state_machine* directive to the registers in the HDL source code.

GATED CLOCKS

- Dedicated FPGA clock trees are routed to every sequential device on the die and are designed to minimize skew to avoid hold-time violations. Thus, the remaining routing resources of the FPGA are available for logic.
- Enable *gated clocks* to conserve power by inserting a multiplexer in a clock net going through an AND, NAND, OR, or NOR gate.



SPECIFYING PHYSICAL-LEVEL OPTIMIZATIONS

- Designing for a Target Architecture (e.g. Xilinx)
- Specifying Macros
- Specifying Global Reset
- Using DSP48 Blocks
- Instantiating CoreGen Cores
- Packing Registers for I/Os
- Specifying RLOCs

DESIGNING FOR A TARGET ARCHITECTURE

- Run successive place-and-route iterations with progressively tighter timing constraints to get the best results possible.
- For critical paths, attach the *xc_fast* attribute to the I/Os.
- The synthesis tool provides Xilinx macro libraries that you can use to instantiate components like I/Os, I/O pads, gates, counters, and flip-flops.
- Use the *IOSTANDARD* parameter to specify the desired I/O standard (e.g. LVCMOS25 or LVPECL).
- To automatically use the GSR (global set/reset), select *Implementation Options -> Device*, and set *Force GSR Usage* to auto.

USING DSP48 BLOCKS

- The synthesis tool automatically tries to use *syn_dspstyle dsp48*.

- Make sure the structure you want to map conforms with these rules:

1. The adder/subtractor does not have more than 96 bits.
2. All registers share the same control signals (enables, clocks, reset).
3. The adder does not have a 48-bit input and a 49-bit output.

INSTANTIATING COREGEN CORES

- Predesigned IP cores (e.g. block rams) save on design effort and improve performance.
- Use the Xilinx CORE generator to create structural EDIF netlists and generate timing and resource usage information for synthesis.
- Define the core as a black box by adding the *syn_black_box* attribute to the module definition line.
- Instantiate the black box in the module or architecture and synthesize.

```
ram64x8 r1(din, addr, we, clk, dout);
```

PACKING IOBS AND SPECIFYING RLOCS

- A register that drives an input or output can be packed into an IOB (input-output block) instead of a CLB (combinational logic block) to minimize the register-to-output delay or input-to-register delay and free up resources for other logic.
- Specify the *syn_useioff* attribute.

- RLOCs are relative location constraints that control placement in critical sections, thus improving performance.
- Specify RLOCs using three attributes, *xc_map*, *xc_rloc*, and *xc_uset*.

OPTIMIZATION STRATEGY

- For optimization, instantiate vendor-supplied macros, BRAMs, IP cores, etc.
- Initially, run synthesis and place/route with minimal constraints to validate the design.
- For each iteration, copy the design files to a new directory, make the desired changes to the constraints and keep the results (which may later be deemed the best).
- Run successive iterations of both the synthesis and place/route tools with progressively tighter constraints to get the best results possible.
- Designate critical paths and rerun the tools.
- Use incremental synthesis and place/route on specific modules to preserve the good ones and to speed tool execution.

FOR ADDITIONAL INFORMATION

- **Synopsys FPGA User Guide**

  http://solvnet.synopsys.com

- **Coding Practices to Accelerate Design Performance**

  http://www.xilinx.com/support/documentation/white_papers/wp231.pdf

- **Design Techniques for FPGA Power Optimization**

  http://www.dsp-fpga.com/articles/id/?4044

- **Power Optimization in FPGA Designs**

  http://www.altera.com/literature/cp/cp-pwropt.pdf