Lecture 11
Logic Synthesis, Part 2

Xuan 'Silvia' Zhang

Washington University in St. Louis

http://classes.engineering.wustl.edu/ese461/

Write Synthesizable Code

- Use meaningful names for signals and variables
- Don't mix level and edge sensitive elements in the same always block
- Avoid mixing positive and negative edge-triggered flip-flops
- Use parentheses to optimize logic structure
- Use continuous assign statements for simple combo logic
- Use nonblocking for sequential and blocking for combo logic
- Don't mix blocking and nonblocking assignments in the same always block (even if Design compiler supports them!!).
- Be careful with multiple assignments to the same variable
- Define if-else or case statements explicitly

Memory Synthesis

- Random logic using flip-flops or latches
  - use large vector or arrays in HDLs
  - inefficient in areas and performance
  - e.g.: a flip-flop takes up to 10 to 20 times area of a 6T SRAM cell
- Register files in datapaths
  - synthesized to a datapath component
  - dependent on software tool and technology
- Memory compilers
  - most area-efficient and high-performance solution
  - foundry, tool, or 3rd party provider

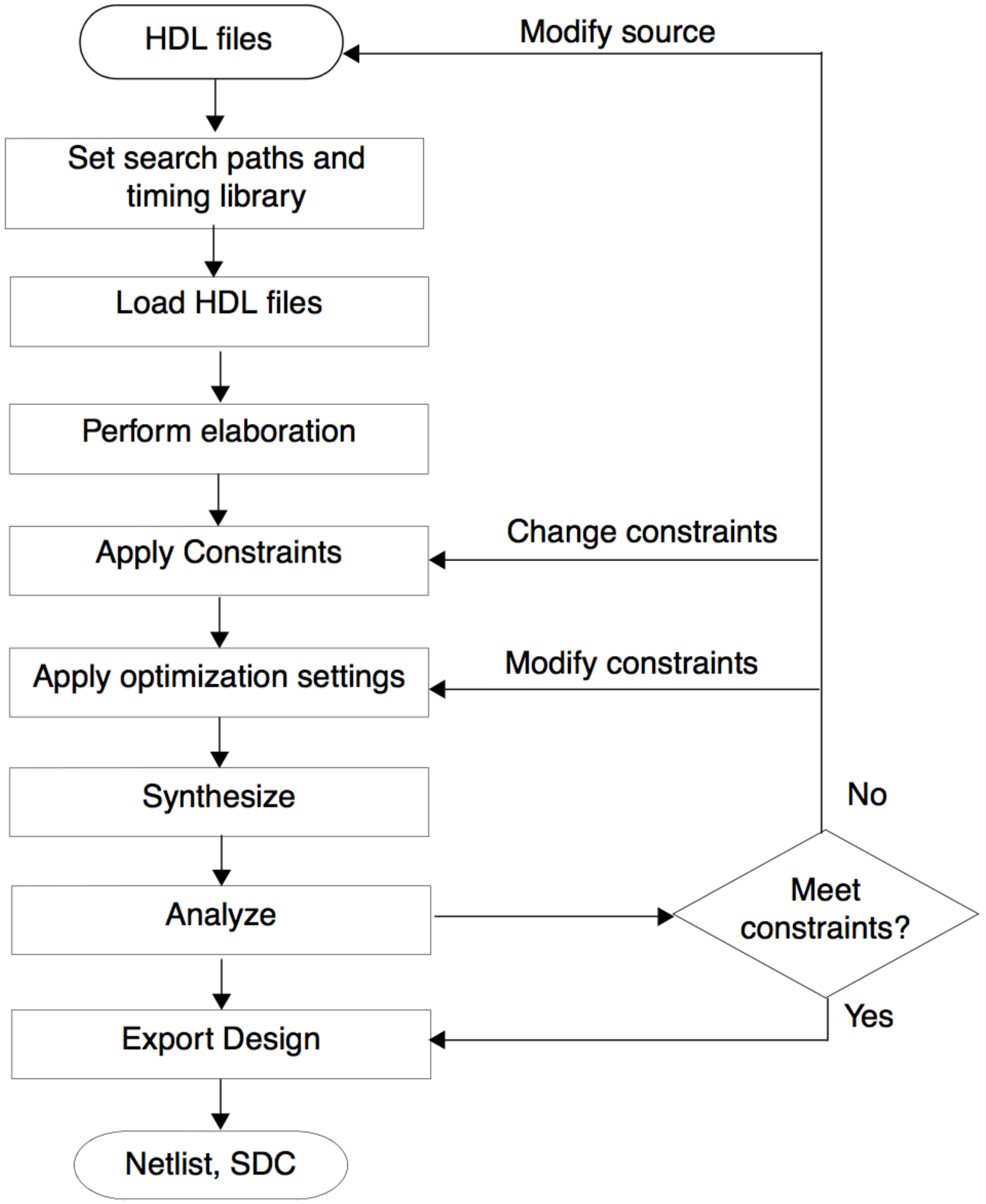
Outline

Write Synthesizable Code

Write Synthesis Script

Design Flow of Synthesis

- Set search paths and timing library
- Load HDL file
- Perform elaboration
- Apply Constraints
- Apply Optimization settings
- Synthesis
- Analysis for constraints
- Export Design
- Netlist and SDC


HDL files
Modify source
Set search paths and timing library
Load HDL files
Perform elaboration
Apply Constraints
Change constraints
Apply optimization settings
Modify constraints
Synthesize
No
Analyze
Meet constraints?
Yes
Export Design
Netlist, SDC

Design Flow of Synthesis

- Set search paths
  - search_path
  - This is the search path for source files and also the the technology library files
- Use set command
  - set search_path <path>
  - where <*path*> is the full path of your target library, script, or HDL file locations.
- analyze
  - Will translates HDL to intermediate format
- read_verilog
  - Will do the job of analyze and elaborate

Design Flow of Synthesis

- Performing Elaboration
  - elaborate
  - Builds data structures
  - Infers registers and latches in the design
  - Performs high-level HDL optimization, such as dead code removal
  - Checks semantics: meaning of sub blocks

Design Flow of Synthesis

- Applying Constraints
- The constraints include
  - Operating conditions
  - Clock waveforms
  - I/O timing
- You can apply constraints in several ways
  - Type them manually in the RTL Compiler shell
  - Include a constraints file
  - Read in SDC constraints
- Two types of constraint
  - Design Rule Check
  - Optimization Constraints

Design Flow of Synthesis

- Applying Optimization Constraints
  - DRC
  - Timing
  - Power
  - Area
- You can perform any of the following optimizations
  - Remove designer-created hierarchies (ungrouping)
  - Create additional hierarchies (grouping)
  - Synthesize a sub-design
  - Create custom cost groups for paths in the design to change the synthesis cost function

Design Flow of Synthesis

- compile _ultra
  - Optimization on full design and complete paths
  - Usually gives best optimization result
  - No iteration required
  - Simpler constraints
  - Simpler data management
  - More processing required
  - More memory required

Design Flow of Synthesis

- Reports
  - Timing: any violation in the timing reports leads to error. Usually solved by operating at lower clock frequencies
  - Area: the rough cell area report before making place and route
  - Power: depends on the operating conditions. Some Technology libraries provide WCCOM option for simulating at worst case conditions
  - Design: overview of the whole simulation in DC compiler

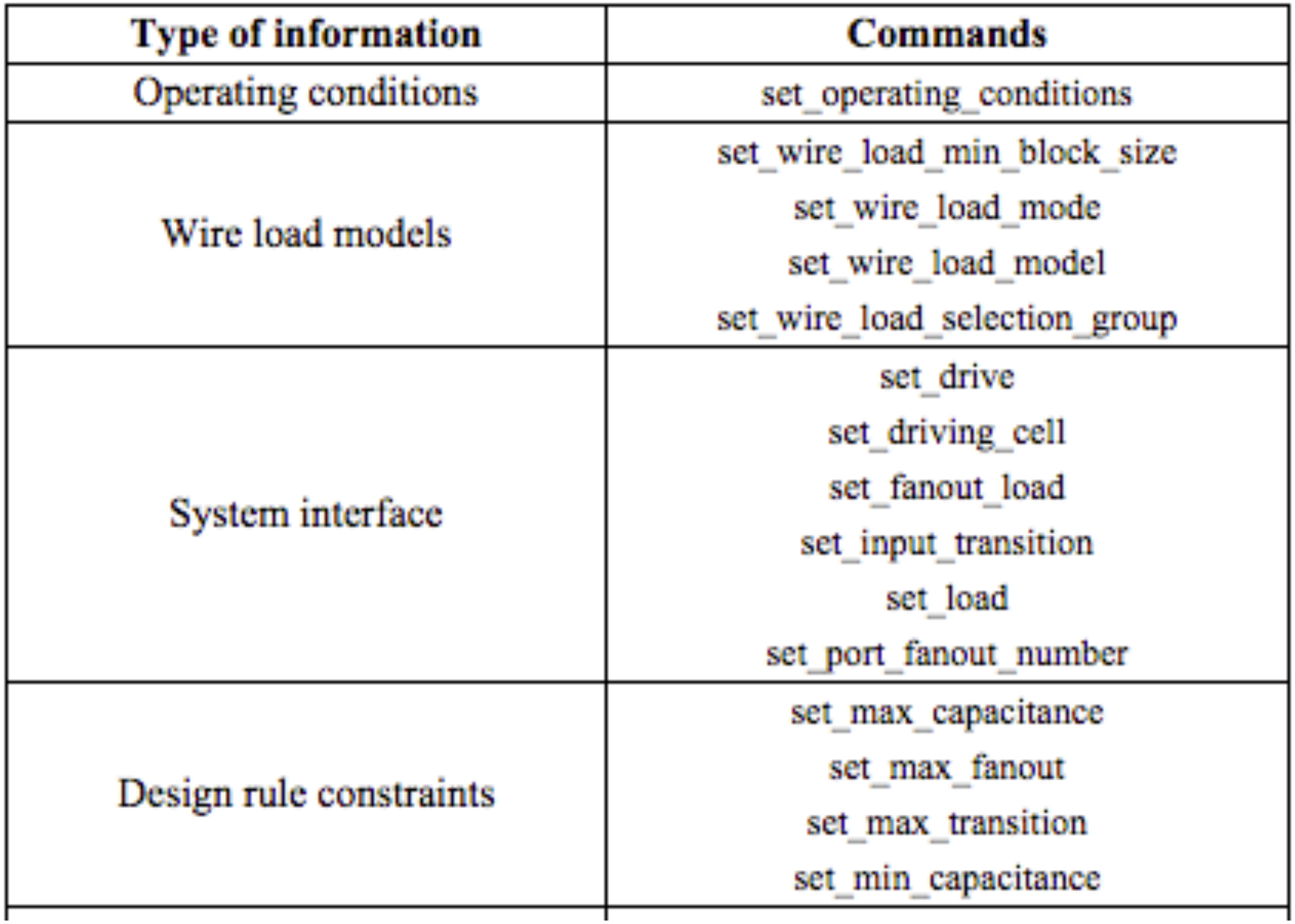
Synopsys Design Constraints (SDC)

- Specify the design intent, including the timing, power, and area constraints for a design
- SDC is Tcl based
- Information in the SDC
  - The SDC version (optional)
  - The SDC units (optional)
  - The Design Constraints
  - Comments (optional)

Synopsys Design Constraints

- SDC version:
  - Variable name: sdc_version
  - e.g.: set sdc_version 1.9
- SDC Units
  - Command name: set_units
  - Specify units for capacitance, resistance, time, voltage, current, and power
  - e.g.: set_units –capacitance 1pF
  - e.g.: set_units –time 1ns

Synopsys Design Constraints

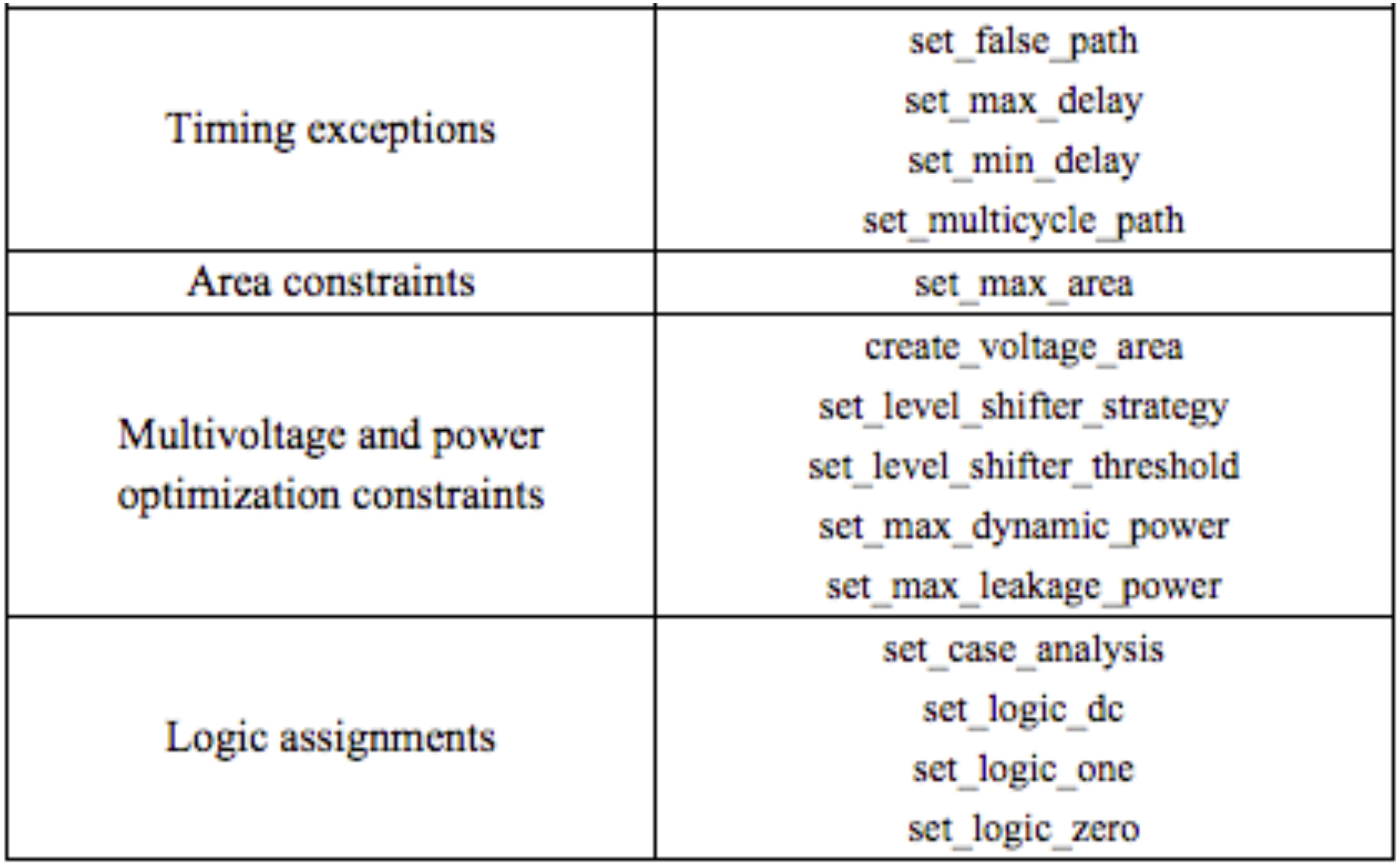


Type of information	Commands
Operating conditions	set_operating_conditions
Wire load models	set_wire_load_min_block_size set_wire_load_mode set_wire_load_model set_wire_load_selection_group
System interface	set_drive set_driving_cell set_fanout_load set_input_transition set_load set_port_fanout_number
Design rule constraints	set_max_capacitance set_max_fanout set_max_transition set_min_capacitance

Synopsys Design Constraints

Timing constraints	create_clock
	create_generated_clock
	group_path
	set_clock_gating_check
	set_clock_groups
	set_clock_latency
	set_clock_sense
	set_clock_transition
	set_clock_uncertainty
	set_data_check
	set_disable_timing
	set_ideal_latency
	set_ideal_network
	set_ideal_transition
	set_input_delay
	set_max_time_borrow
	set_output_delay
	set_propagated_clock
	set_resistance
	set_timing_derate

Synopsys Design Constraints



Timing exceptions	set_false_path
	set_max_delay
	set_min_delay
	set_multicycle_path
Area constraints	set_max_area
Multivoltage and power optimization constraints	create_voltage_area
	set_level_shifter_strategy
	set_level_shifter_threshold
	set_max_dynamic_power
	set_max_leakage_power
Logic assignments	set_case_analysis
	set_logic_dc
	set_logic_one
	set_logic_zero

Synopsys Design Constraints

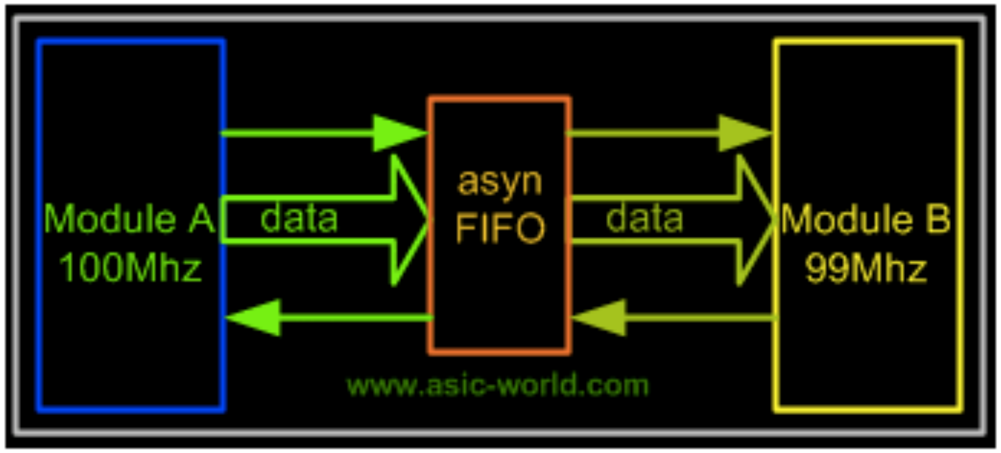
- create_clock
  - Name
  - Period
  - Waveform
  - [get_ports {}]
  - e.g.: create_clock –name “clk” –add –period 500.0 –waveform {0, 250} [get_ports{clk}]

Technology Library files

- db file
  - the actual information about the cells used in the linking
- sdb file
  - information about the symbols used for the cells in the standard cell library
  - used in the process of P&R because we can see the black boxes instead of the gate level logic.
- LEF file
  - related to the P&R tools
  - layout exchange file which has information regarding no of layers of metal used or available for P&R.

Lab #4: Dual-Clock FIFO

- Due 10/19 (Wednesday)
- Cross different clock domains
  - handshake signaling
  - asynchronous first-in-first-out buffer (FIFO)
- FIFO
  - two interfaces
  - two clocks
  - one for write, one for read



Questions?

Comments?

Discussion?