ECE 551

Digital System Design & Synthesis

Synthesis

Administrative Matters

- Midterm is Thursday March 10th in Psychology 113
- Synthesis tutorial March 14th, 15th & 16th
  - ✓ Monday 6:00 – 7:00 room B555
  - ✓ Tuesday 6:00 – 7:00 room B555
  - ✓ Wednesday 6:00 – 7:00 room B555
- HW3 Due Friday March 3rd @ Class
- Team Formation (teams of 3-4)
  - Start arranging your teams

Synthesis Priorities

- First there is functionality
  - Is function coded in Verilog same as synthesized netlist?
    - Boolean correct
    - Sequential correct
    - No uninteded latches or strange clock gating
- Next there are design rules
  - Is the fanout reasonable.
  - Are my transition times fast enough (Hot Electron)
- Finally there are performance criteria
  - Speed
  - Power
  - Area

Synthesis is cost function driven

- Cost is a function of:
  - Timing
  - Area
  - Power

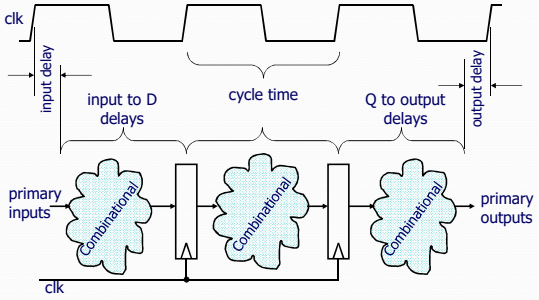
In this priority order

Cost(mapping) = -Slack(mapping) + Area(mapping) + [power(mapping)]

Explore mapping space to minimize this function *(not the exact function)*

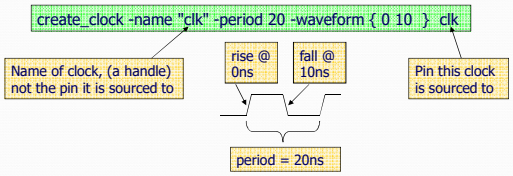
- Mapping refers to the mapping of the logic (elaborated database) to the cell library
- Synopsys is a bit like the typical ECE Student…Smart, but a touch lazy
  - Loose Constraints ➔ Loose Design

What Needs Constraint?



Setting Basic Timing Constraints

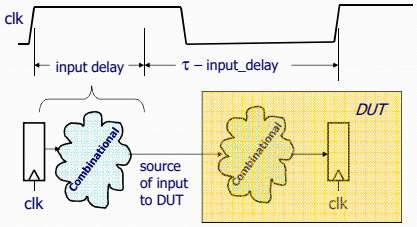
- Establishing clock period, waveform, and pin it is sourced to:



Clock is no ordinary input. Can't let Synopsys buffer it and do other silly such things.

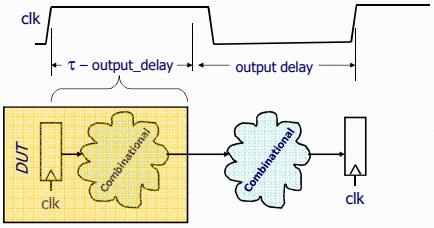
If clock distribution requires buffering then that will be done by a separate tool (CTS) inside the APR environment.

Defining Input Delay



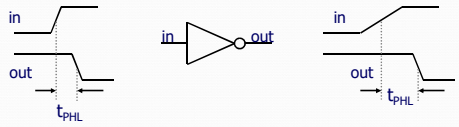
input delay is specified as time after the clock edge (of prior clock cycle) that the input to the DUT is valid.

Defining Output Delay



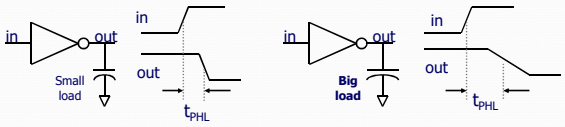
output delay is specified as time prior to next rising edge that the output has to be valid.

What Else Needs Constraint?



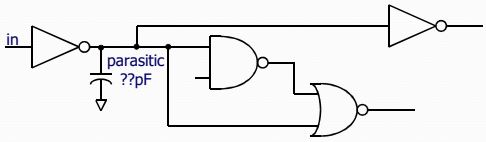
- Propagation delay of a gate is not constant
  - ✓ There is a term that depends on input slope
  - ✓ Synthesis knows the capacitance of a primary input
  - ✓ Synthesis **does not** know the drive strength of a primary input
- We must inform Synopsys about the drive strength of the inputs
  - set_driving_cell -lib_cell <cell> -pin Z -library <lib> <list_of_inputs>
  - set_drive <# of ohms> <signal>

What Else Needs Constraint?



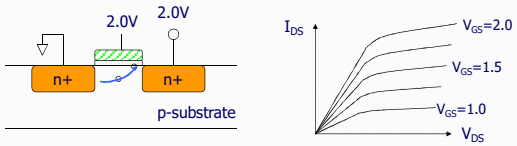
- Capacitive loading affects the propagation delay of a gate
  - ✓ For a given drive strength the slope of the output will elongate with increased capacitive load
  - ✓Synopsys needs to know the capacitive load of primary outputs so it can size gates (or buffer up) to drive load with "crisp" transition times.
- We must inform Synopsys about the load on primary outputs
  - set_load <capacitance> <list of outputs>

Anything More Need Constraints?



- What is the parasitic capacitance of this node?
  - ✓ gate capacitance is known by Synopsys
  - ✓ routing capacitance is the unknown
  - ✓ how to estimate?
    - fanout
    - size of design (gate count)
- set_wire_load_model -name <m_name> -library <lib_name>

Are We Done Yet…Not Quite



- When both V_{DS} and V_{GS} are high a transistor is conducting heavily.
- Electrons "flying" across the channel get a lot of kinetic energy.
- These electrons are also attracted to the surface by the high gate potential.
- They can embed themselves in the gate oxide (causes V_T degradation).
- This is called "Hot Electron Effect"
- Need to limit operating time at high V_{DS} & V_{GS}. Need "snappy" transitions.

set_max_transition_time <time> [current_design]

GUI's are for Children

- GUI's are for learning and some instances of debugging.
- Once you know the tool you will mainly work in shell mode (at least that is my preference):

```
unix_prompt>design_vision –shell dc_shell
```

13

Sample DC Shell Script [1]

```
read_file -format verilog {./source/ss_a2d_dp.v}

############################
# Define clock and set don't mess with it #
############################
create_clock -name "clk" -period 20 -waveform { 0 10 }  { clk  }
set_dont_touch_network [find port clk]

# setup pointer that contains all inputs except clock #
set prim_inputs [remove_from_collection [all_inputs] [find port clk]]

########################
# Set input delay & drive on all inputs #
########################
set_input_delay -clock clk 5 [copy_collection $prim_inputs]
set_driving_cell -lib_cell AO33D0BWP -pin Z -from_pin A1 -library \
                 tcbn40lpbwptc [copy_collection $prim_inputs]
# tell it por_n strongly driven so it won't buffer
set_drive 0.1 por_n
```

14

Sample DC Shell Script [2]

```
############################
# Set output delay & load on all outputs #
############################
set_output_delay -clock clk 5 [all_outputs]
set_load 0.10 [all_outputs]

####################################
# Wire load model allows it to estimate internal parasitics #
####################################
set_wire_load_model -name TSMC32K_Lowk_Conservative \
                    -library tcbn40lpbwptc

################################
# Max transition time is important for Hot-E reasons #
################################
set_max_transition 0.10 [current_design]

########################
# Now actually synthesize for 1st time #
########################
compile -map_effort low
```

15

Sample DC Shell Script [3]

```
check_design
## design ware component caused extra pins

report_area
########################
# Take a look at max & min timings #
########################
report_timing -path full -delay max -nworst 3
report_timing -path full -delay min -nworst 3

## smash the hierarchy (design ware component)
ungroup -all

compile -map_effort medium

check_design

report_area

#### write out final netlist ######
write –format verilog ss_a2d_dp –output ss_a2d_dp.vg
```

Now lets run this line by line and see what it does

16

Some "Nice to Have" Constraints

- set_max_area 20000
  - Sets maximum area to 20,000 area units (square microns)
- set_max_dynamic_power 10
  - Sets maximum dynamic power to 10 mW (default unit for our library is mW)

▪ **Note:** Power & Speed often work against each other.
- ✓ A tight timing constrained design will use more parallel hardware.
- ✓ More nodes toggling, results computed and then discarded

▪ **Note:** Area & Power often work together.
- ✓ In general less area => less toggling nodes
- ✓ less net capacitance switching

17

Timing Reports….How do I read these?

```
dc_shell-xg-t> report_timing -delay max –nworst 1

****************************************
Report : timing
        -path full
        -delay max
        -max_paths 1
Design : ss_a2d_dp
Version: Y-2006.06-SP1
Date   : Thu Oct  2 09:00:42 2008
****************************************

Operating Conditions: NOM   Library: gflxp
Wire Load Model Mode: enclosed

  Startpoint: DAC_reg[0] (rising edge-triggered flip-flop clocked by clk)
  Endpoint: DAC_full (output port clocked by clk)
  Path Group: clk
  Path Type: max
```

report the worst timing path for max delay scenario

Startpoint <signal> <type>
Endpoint <signal> <type>
Very useful to look at

Report continued next page

18

```
Point                                   Incr       Path
--------------------------------------------------------------
clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.00       0.00
DAC_reg[0]/CP (FD2LQM3P)                0.00       0.00 r
DAC_reg[0]/Q (FD2LQM3P)                 0.25       0.25 r
U59/Z (AND3M1P)                         0.10       0.35 r
U55/Z (ND4M2P)                          0.13       0.48 f
U56/Z (NR3M6P)                          0.21       0.69 r
DAC_full (out)                          0.00       0.69 r
data arrival time                                  0.69

clock clk (rise edge)                  20.00      20.00
clock network delay (ideal)             0.00      20.00
output external delay                  -5.00      15.00
data required time                                15.00
--------------------------------------------------------------
data required time                                15.00
data arrival time                                 -0.69
--------------------------------------------------------------
slack (MET)                                       14.31
```

+ slack is a good thing, it means met timing

19

Timing Report…min delay

```
dc_shell-xg-t> report_timing -delay min -nworst 1
Information: Updating design information... (UID-85)

****************************************
Report : timing
        -path full
        -delay min
        -max_paths 1
Design : ss_a2d_dp
Version: Y-2006.06-SP1
Date   : Thu Oct  2 09:17:50 2008
****************************************

Operating Conditions: NOM   Library: gflxp
Wire Load Model Mode: enclosed

  Startpoint: accum_reg_reg[0]
          (rising edge-triggered flip-flop clocked by clk)
  Endpoint: accum_reg_reg[0]
          (rising edge-triggered flip-flop clocked by clk)
```

Use keywork min to get this report

Surprise, Surprise, the min delay is a path between two flops

Report continued next page

20

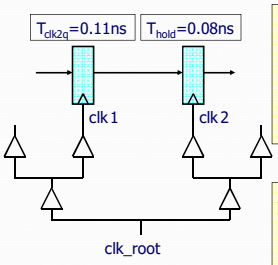
Timing Report…min delay

```
Point                                   Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.00       0.00
accum_reg_reg[0]/CP (FD2LQM1P)          0.00       0.00 r
accum_reg_reg[0]/Q (FD2LQM1P)           0.11       0.11 f
U67/Z (EOFM1P)                          0.05       0.16 r
accum_reg_reg[0]/D (FD2LQM1P)           0.00       0.16 r
data arrival time                                  0.16

clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.00       0.00
accum_reg_reg[0]/CP (FD2LQM1P)          0.00       0.00 r
library hold time                       0.08       0.08
data required time                                 0.08
-----------------------------------------------------------
data required time                                 0.08
data arrival time                                 -0.16
-----------------------------------------------------------
slack (MET)                                        0.09
```

Oh good, we made it

Hold time (worst case is back to back FF's)



Hold time slack for back to back flop scenario with our library

Slack = Clk2q – Hold = +0.03ns

We are "golden" we don't even have to worry about this…right?

Any large design will require a clock tree to distribute the clock.

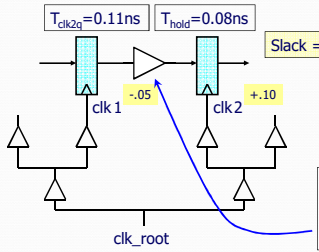
Clock is a large capacitive load. Can't just drive with one honking driver (RC of wires will kill you then)

Now you have clock skew

set_clock_uncertainty 0.15 clk

Inform Synopsys about your skew

Hold Time…how do we fix?



Slack = Clk2q – Hold – Uncertainty = -0.12ns

CTS did the best it could already

Have to add a buffer to slow the signal down

Referred to as a: min-delay buffer

Lets Play…fire up design_vision

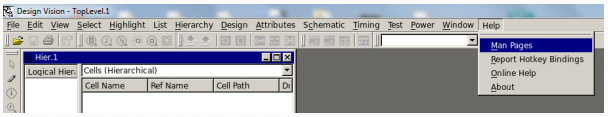
- Code a 64 bit accumulator
- read_file –format verilog accum.v
- constrain the clock to 2ns and compile
- report_area ➔ why are there only 2 combinational cells??
- check_design ➔ why are there all these dangling nets?
- write –format verilog accum –output accum.vg
- ungroup –all –flatten ➔ compile again
- report_area again
- report_timing –delay max
- now constrain clock to 1.5ns and compile again…what did area do
- report_timing –delay min
- set_clock_uncertainty 0.2 clk
- report_timing –delay min ➔ compile agian
- report_timing –delay min ➔ Why are there still violators
- set_fix_hold clk
- compile again

"Holy Mackerel Batmat, This thing is a monster"

- Synopsys is 1 of only 2 significant players in the EDA world
  - Any startup with an interesting EDA tool gets "assimulated"
  - Their EDA tool offering is vast (many good tools)
    - ✓ We are only interested in Design Vision/Design Compiler
  - Design Compiler is their oldest most used tool (around in one form or another since 1986)
    - ✓ 28 years of development makes for a monster of a CAD tool

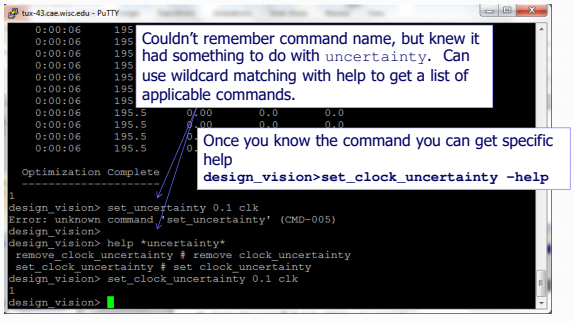
Help!

- If you kick off design_vision in GUI mode it has a help➔Man Pages in the banner menu



- Under this you will find the full list of commands
- Yaaa … real fine Hoffman, this only does me good if I know what command I am looking for.

Help!



Optimization Priorities

- Design rules have priority over timing goals
- Timing goals have priority over area goals
  - Max delay has priority over min-delay
- To prioritize area constraints:
  - use the **ignore_tns** (total negative slack) option when you specify the area constraint:

    set_max_area -ignore_tns 10000

    Specifies that area is to be prioritized above TNS
- To change priorities use **set_cost_priority**
  - Example: set_cost_priority -delay

    prioritizing delay over max transition rules

Constraints Default Cost Vector

Priority (descending order)	Notes
connection classes	
multiple_port_net_cost	
min_capacitance	Design Rule Constraint
max_transition	**Design Rule Constraint**
max_fanout	**Design Rule Constraint**
max_capacitance	**Design Rule Constraint**
cell_degradation	**Design Rule Constraint**
max_delay	**Optimization Constraint**
min_delay	**Optimization Constraint**
power	Optimization Constraint
area	Optimization Constraint
cell count	

Compiling the Design

- Useful **compile** options include:
  - -map_effort low | medium | high (default is medium)
  - -area_effort low | medium | high (default same as map_effort)
  - -incremental_mapping (may improve already-mapped)
  - -verify (compares initial and synthesized designs)
  - -ungroup_all (collapses all levels of design hierarchy)
- New **compile_ultra** command
  - Two pass high effort compile of the design
  - May want to compile normally first to get ballpark figure (higher effort == longer compilation)

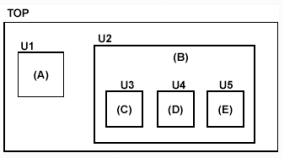
Top-Down Compilation

- Use top-down compile strategy for medium to small designs
- Basic steps are:
  - Read in the entire design
  - Resolve multiple instances of any design references with **uniquify**
  - Apply attributes and constraints to the top level
  - Compile the design using **compile** or **compile_ultra**

Example Top-Down Script

```
# read in the entire design
read_file -library WORK -format verilog {E.v D.v C.v B.v A.v TOP.v}
current_design TOP
# link TOP.v to libraries and modules it references
Link
# set design constraints
...
set_max_area 2000
# resolve multiple references
uniquify
# compile the design
compile –area_effort high
```

What?? Will cover this a bit later



Bottom-Up Compile Strategy

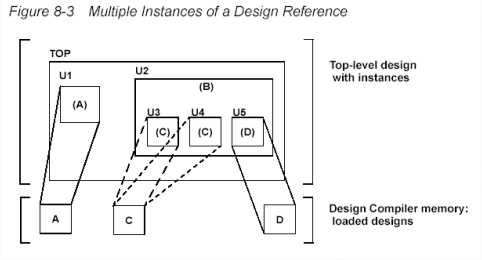
- The bottom-up compile strategy
  - Compile the subdesigns separately and then incorporate them
  - Top-level constraints are applied and the design is checked for violations.
- Advantages:
  - Compiles large designs more quickly (divide-and-conquer)
  - Requires less memory than top-down compile
- Disadvantages
  - Need to develop local constraints as well as global constraints
  - May need to repeat process several times to meet design goals
- Only likely to use if running into serious memory or performance issues

This is a royal pain in the butt

Resolving Multiple References

- In a hierarchical design, subdesigns are often referenced by more than one cell instance

Figure 8-3 Multiple Instances of a Design Reference

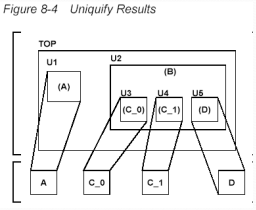


Uniquify Method

- The **uniquify** command creates a uniquely named copy of the design for each instance.

current_design top
uniquify
compile

Figure 8-4 Uniquify Results

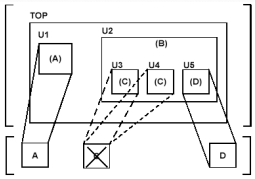


- Each design optimized separately
- What are advantages and disadvantages?

Compile-once-don't-touch Method

- The compile-once-don't-touch method uses the set_dont_touch command to preserve the compiled subdesign

current_design top
characterize –constraints U2/U3
current_design C
compile
current_design top
set_dont_touch {U2/U3 U2/U4}
compile

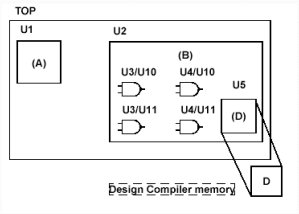


- What are advantages and disadvantages?

Ungroup Method

- The ungroup command makes unique copies of the design and removes levels of the hierarchy

```
current_design B
ungroup {U3 U4}
current_design top
compile
```



- What are advantages and disadvantages?

Flattening Hierarchy

```
module logic1(input a, c, e, output reg x);
always @(a, c, e)
  x = ((~a|~c) & e) | (a&c);
endmodule

module logic2(input a, b, c, d, output reg y);
always @(a, b, c, d)
  y = ((((~a|~c)&b) | ((a|~b)&c))&d) | ((a|~b)&~d);
endmodule

module logic(input a, b, c, d, e, f, output reg z);
wire x, y;
logic1(a, c, e, x);
logic2(a, b, c, d, y);
always @(x, y, f)
  z = (~f&x) | (f&y);
endmodule
```

Unflattened Hierarchy
Area: 36.15
Delay: 0.25

Flattened Hierarchy
Area: 34.15
Delay: 0.25

Checking your Design

- Use the **check_design** command to verify design consistency.
  - Usually run both before and after compiling a design
  - Gives a list of warning and error messages
  - Errors will cause compiles to fail
  - Warnings indicate a problem with the current design
    - Try to fix all of these, since later they can lead to problems
    - Use check_design –summary or check_design -no_warnings to limit the number of warnings given
  - Use **check_timing** to locate potential timing problems
  - Use **report_contraints –all_violators** (check everything)

Analyzing your Design [1]

- There are several commands to analyze your design
  - **report_design**
    - display characteristics of the current design
    - operating conditions, wire load model, output delays, etc.
    - parameters used by the design
  - **report_area**
    - displays area information for the current design
    - number of nets, ports, cells, references
    - area of combinational logic, non-combinational, interconnect, total

Analyzing your Design [2]

- **report_hierarchy**
  - displays the reference hierarchy of the current design
  - tells modules/cells used and the libraries they come from
- **report_timing**
  - reports timing information about the design
  - default shows one worst case delay path
- **report_resources**
  - Lists the resources and datapath blocks used by the current design
- Can send reports to files
  - report_resources > cmult_resources.rpt
- Lots of other report commands available

41

Optimization Strategies

- Area vs. Delay - Often only really optimize for one
  - "Fastest given an area constraint"
  - "Smallest given a speed constraint"
- Design Compiler Reference Manual has several pointers on synthesis settings for these goals

42