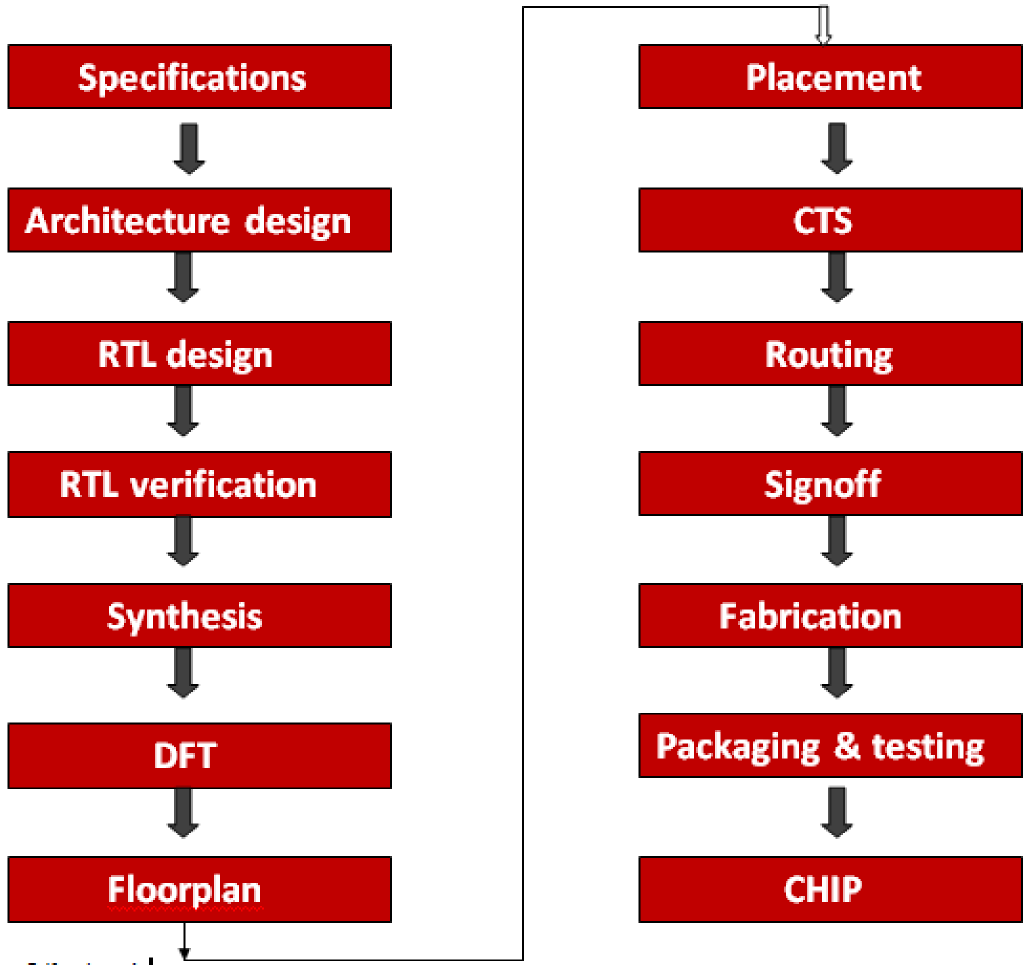
Department of Electrical and Computer Engineering

University of Thessaly

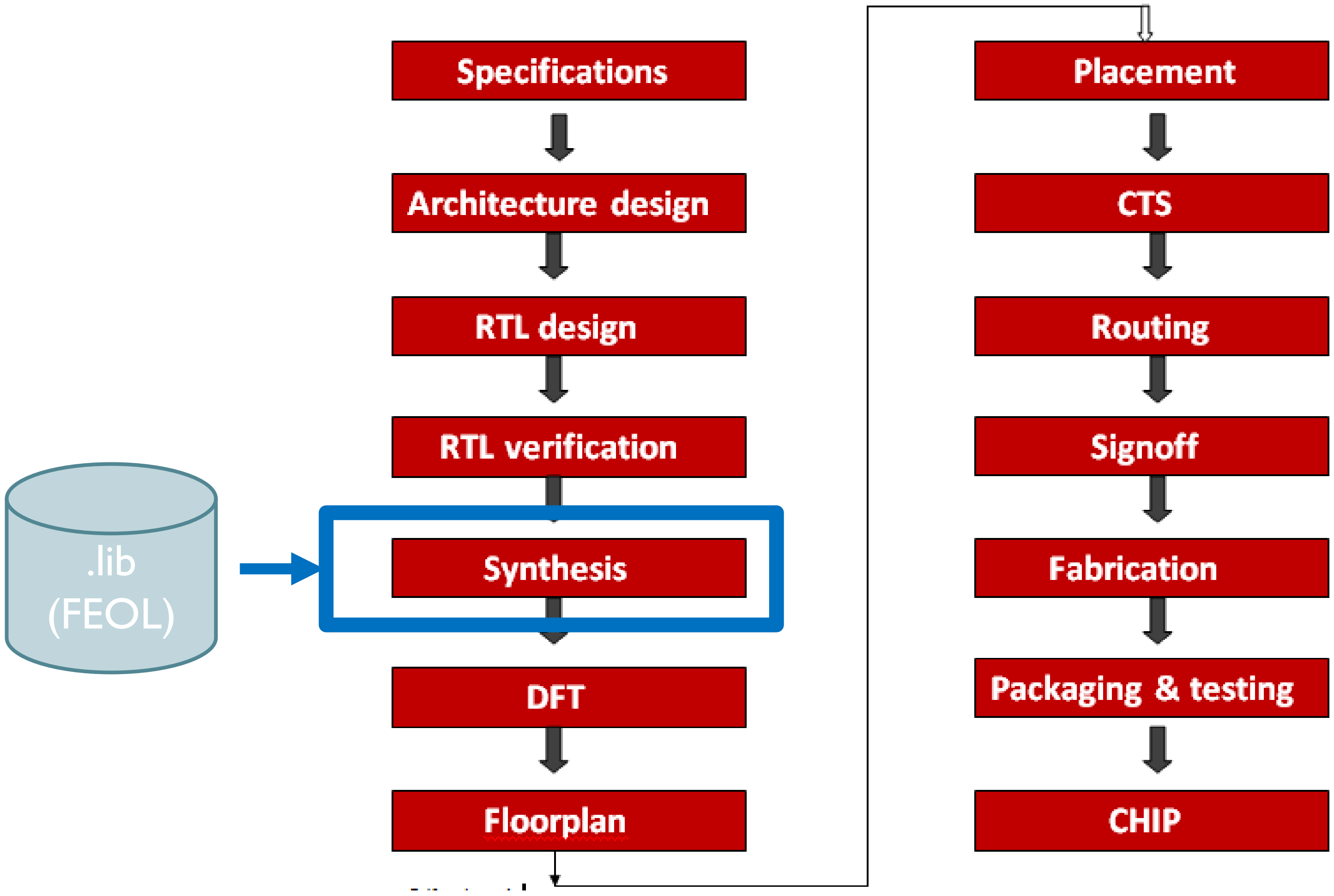
CE327 - Digital VLSI Systems
Semester Project Presentation
Fall Semester - 2022

I. Lilitsis, I. Gkolfos, S. Simoglou, C. Sotiriou

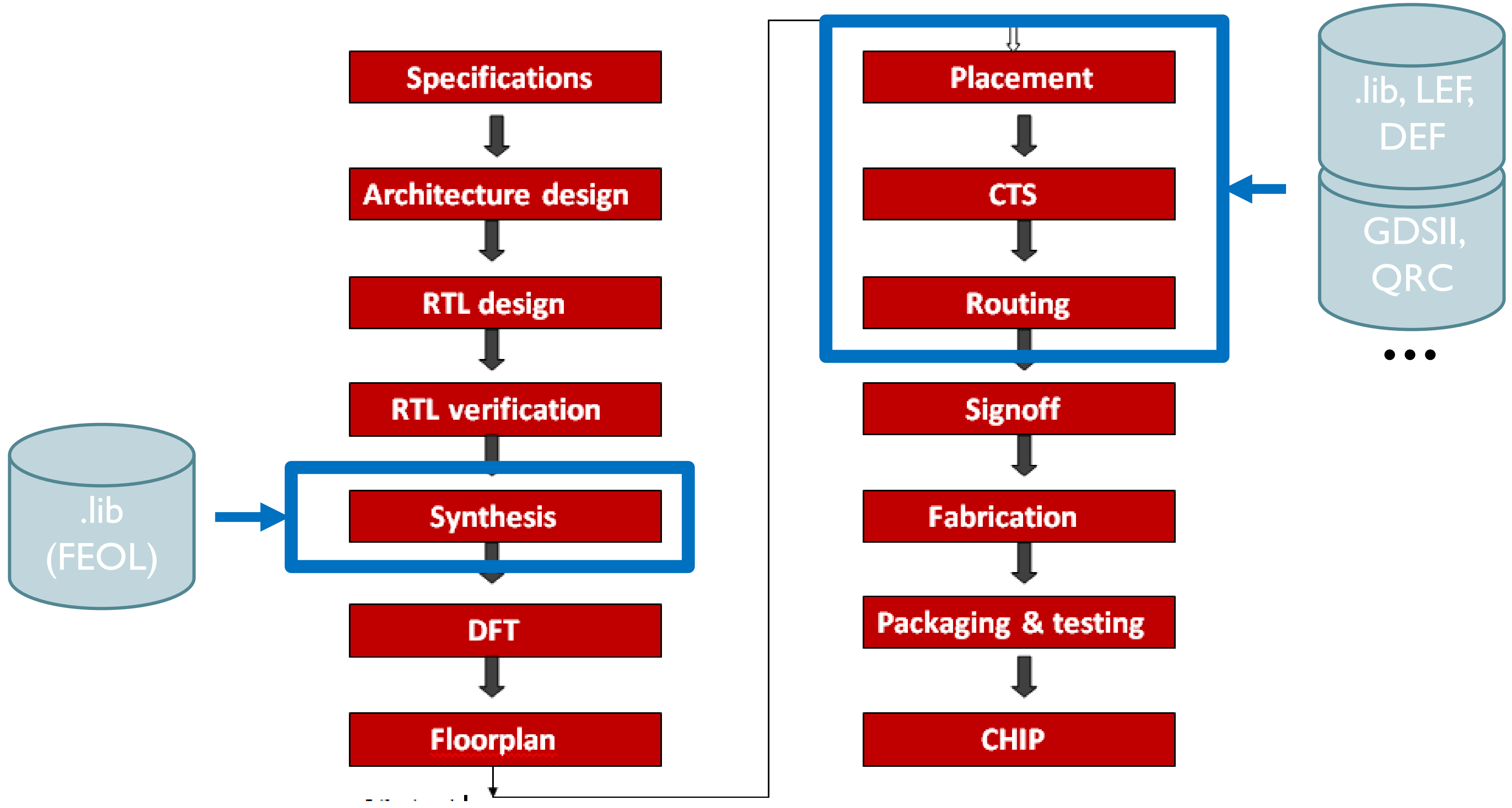
ASIC Design Flow



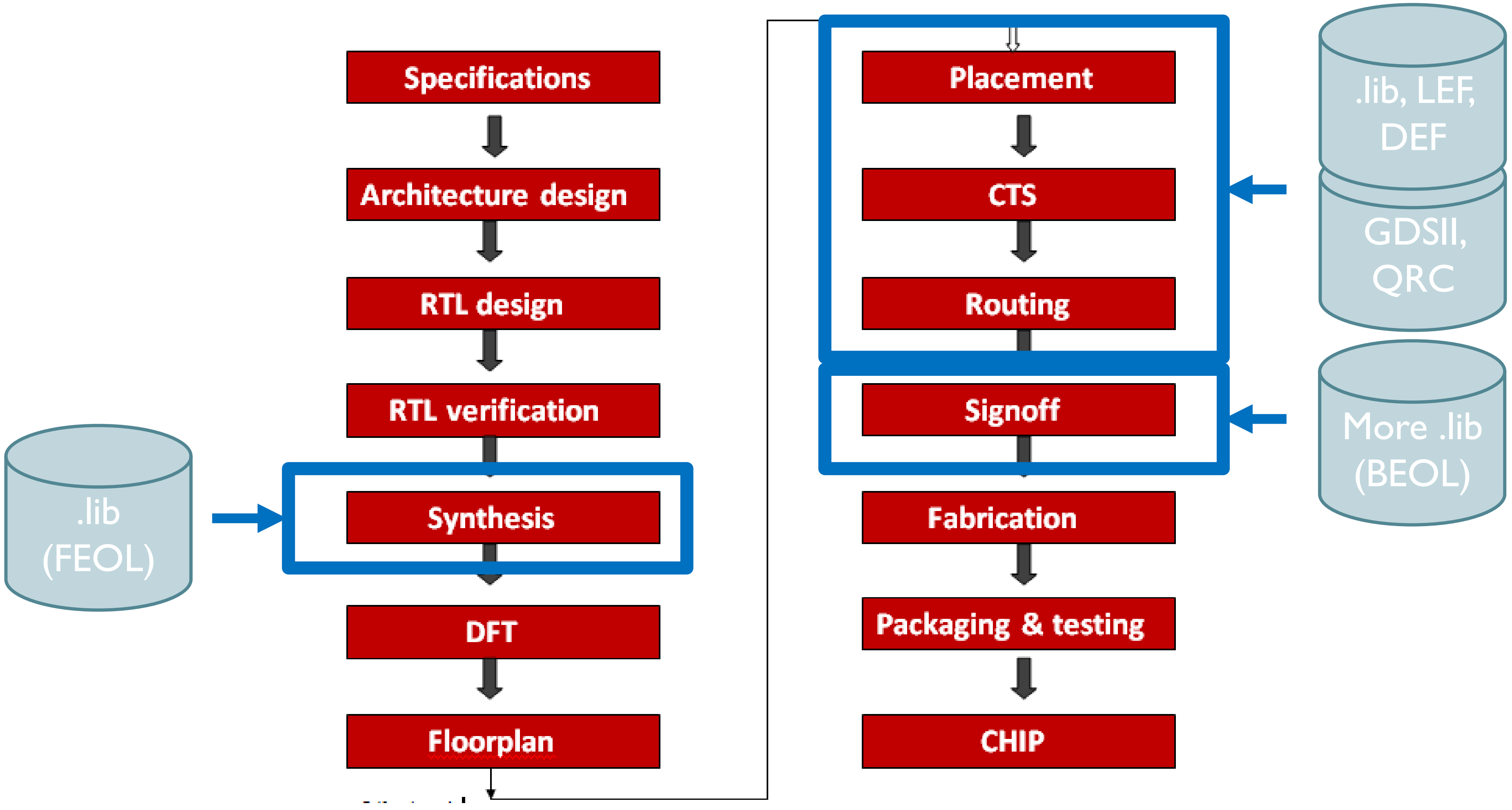
ASIC Design Flow



ASIC Design Flow

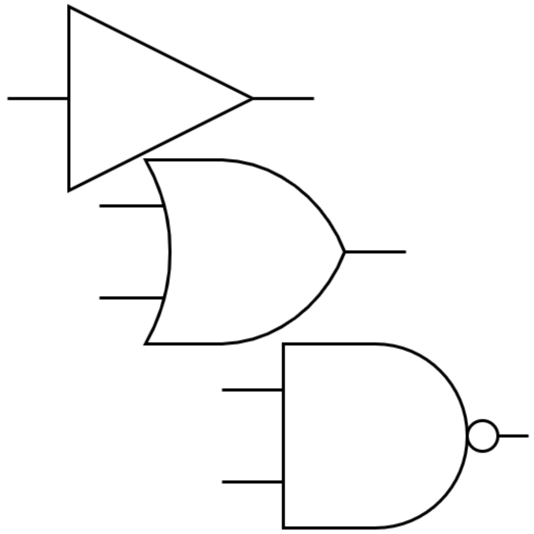


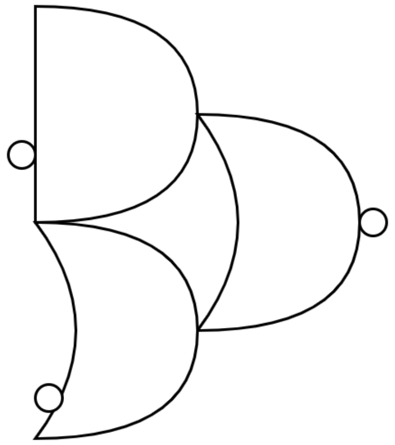
ASIC Design Flow



Digital VLSI Systems Components

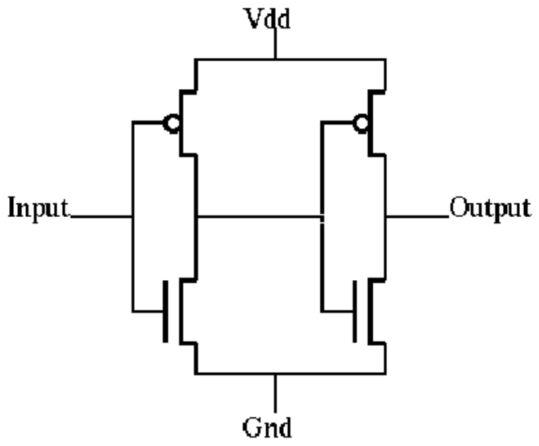
- Combinational Elements:
  - Implement stateless Boolean functions
    - Output is always produced upon any input change

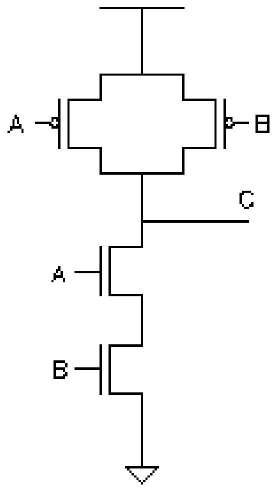


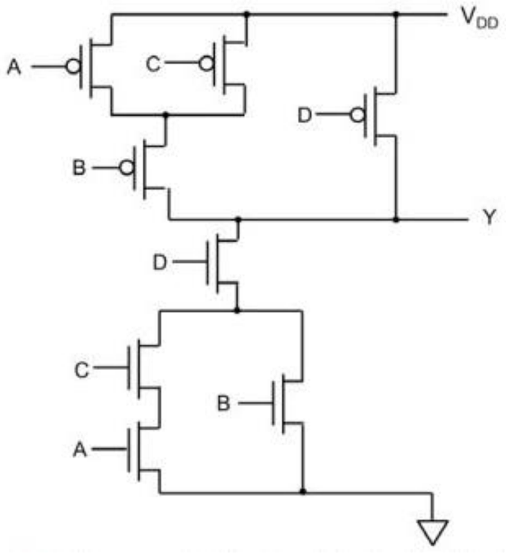


Digital VLSI Systems Components

- Combinational Elements:
  - Implement stateless Boolean functions
    - Output is always produced upon any input change

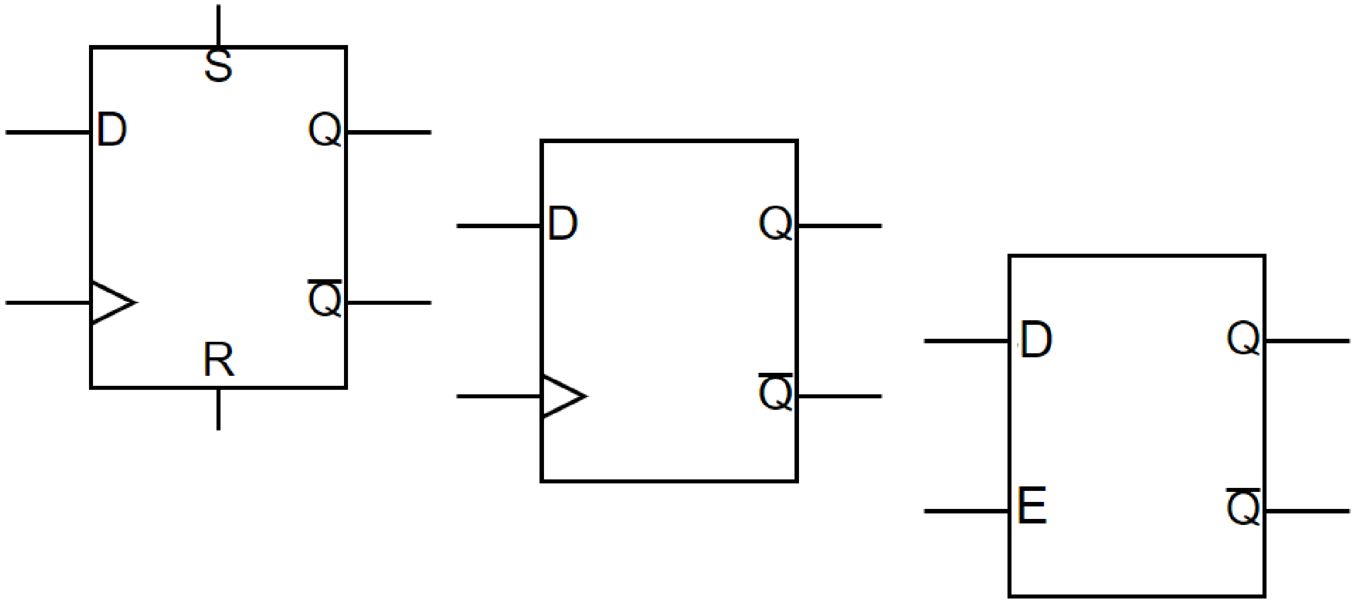






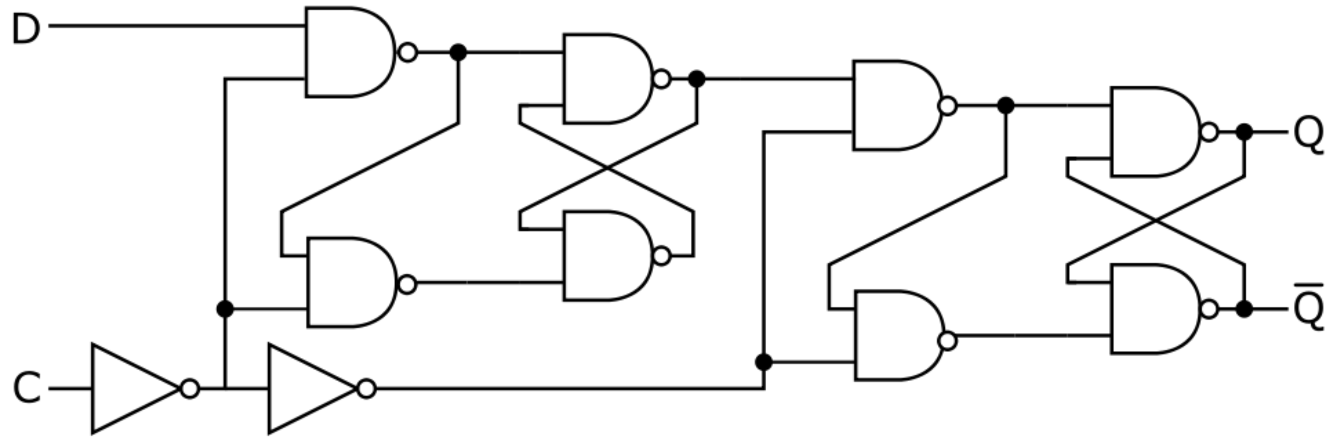
Digital VLSI Systems Components

- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon specific triggering events, i.e. clock edges



Digital VLSI Systems Components

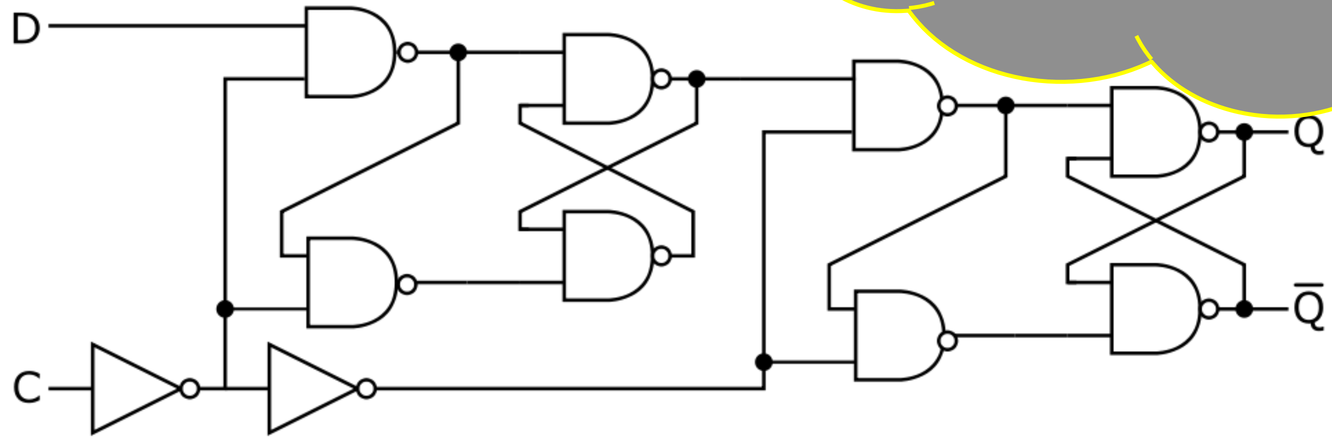
- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon specific triggering events, i.e. clock edges



Digital VLSI Systems Components

- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon spe

Let's count transistors:

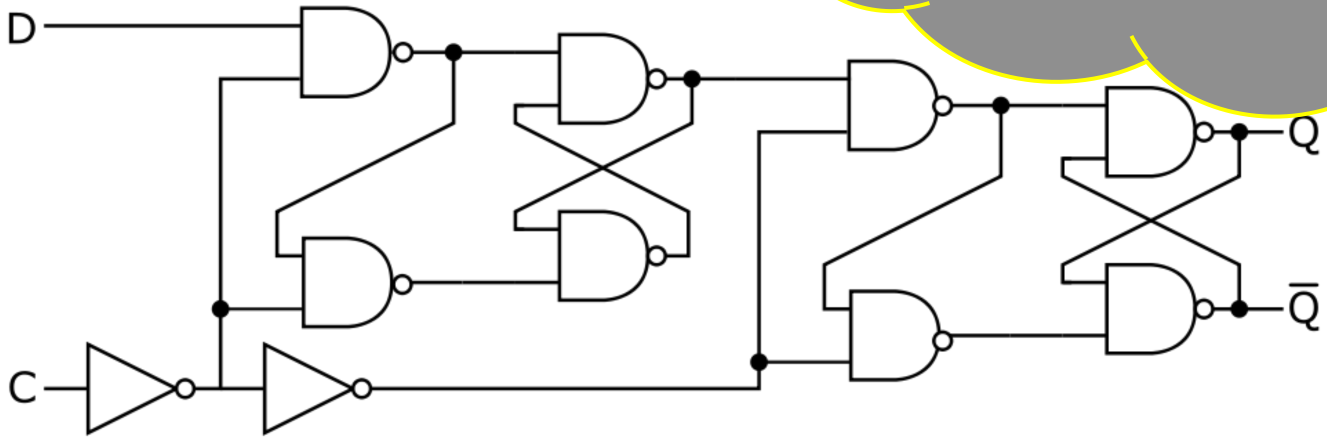


Digital VLSI Systems Components

- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon spe

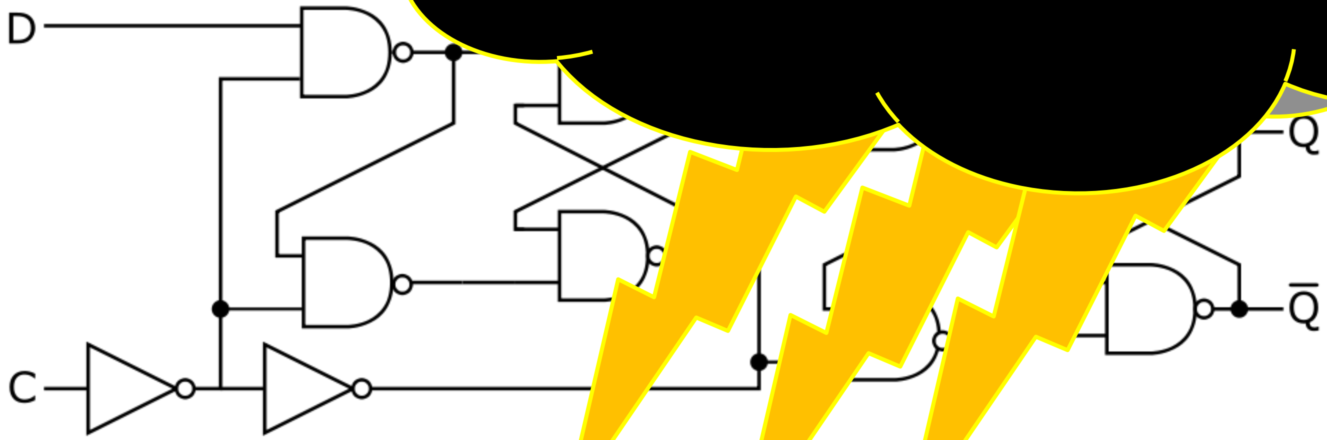
Let’s count transistors:

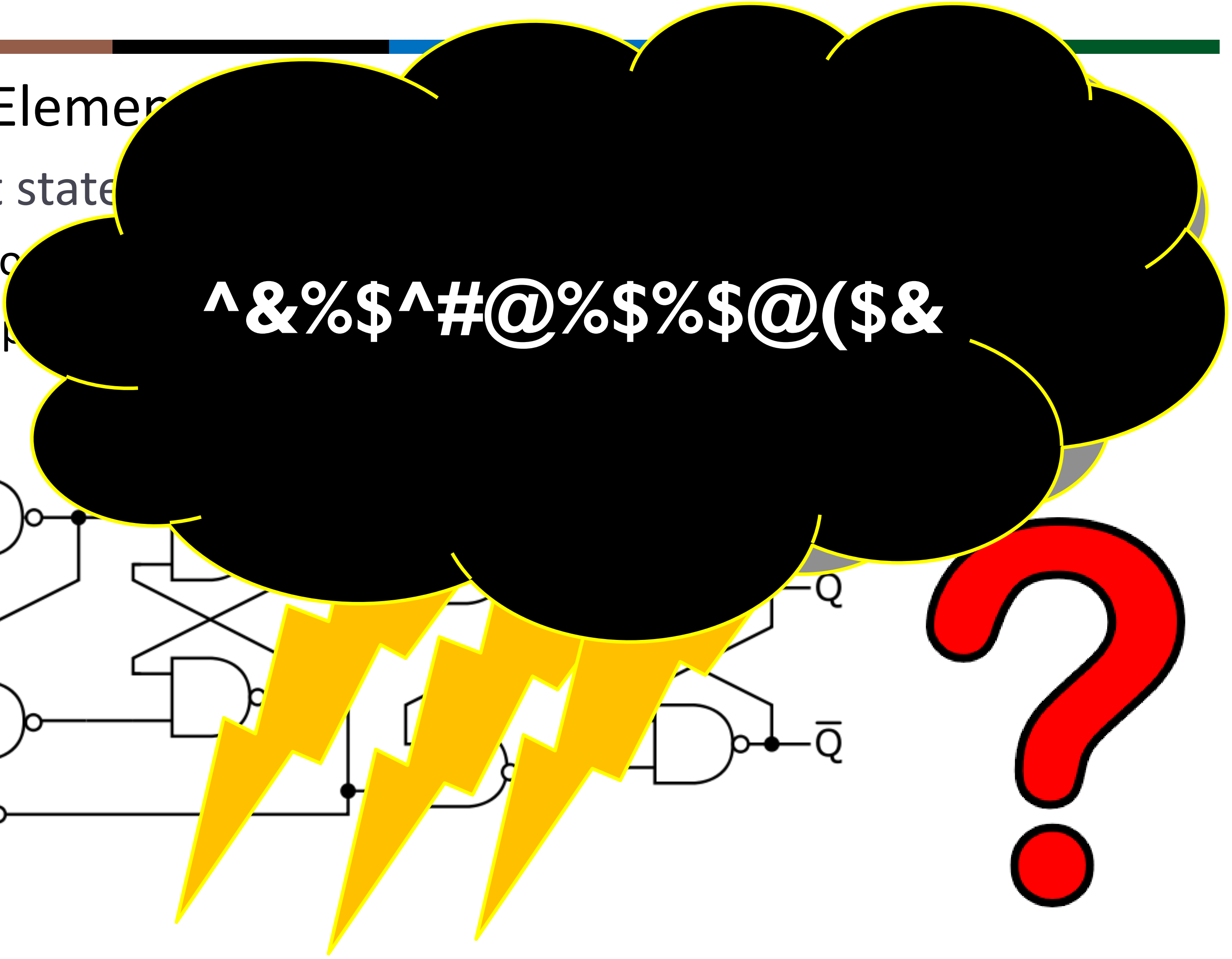
36



Digital VLSI Systems Components

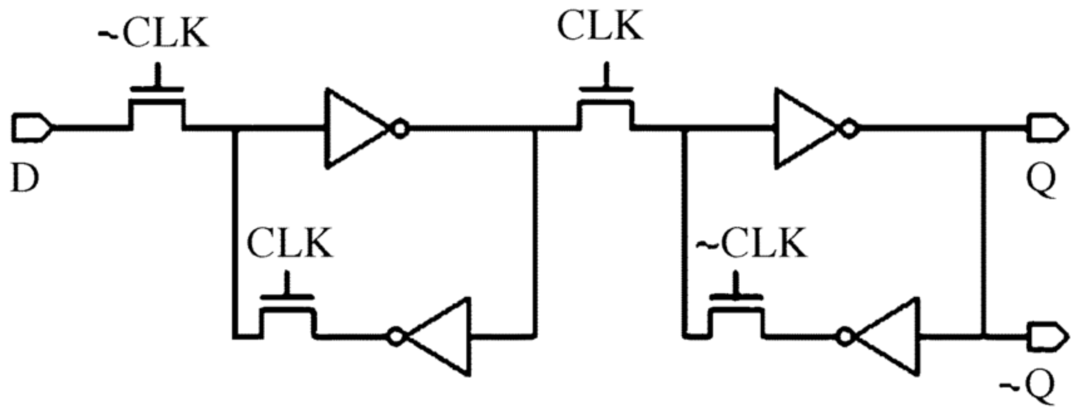
- Sequential Elemen
  - Implement state
    - Input is sto
    - Output is p





Digital VLSI Systems Components

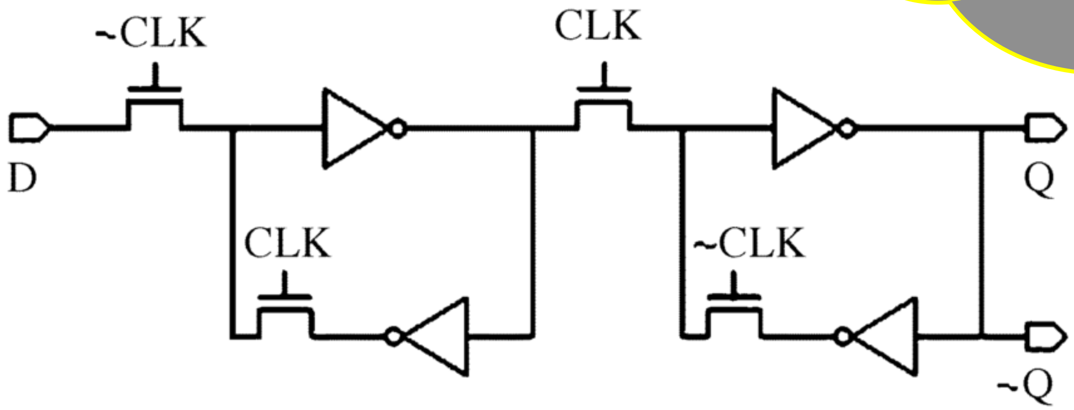
- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon specific triggering events, i.e. clock edges



Digital VLSI Systems Components

- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon spe

Let’s count transistors:

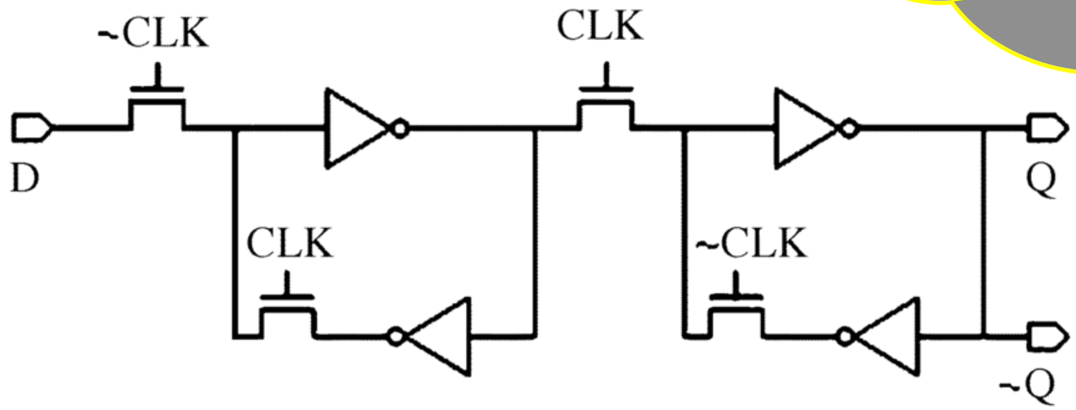


Digital VLSI Systems Components

- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon spe

Let's count transistors:

12

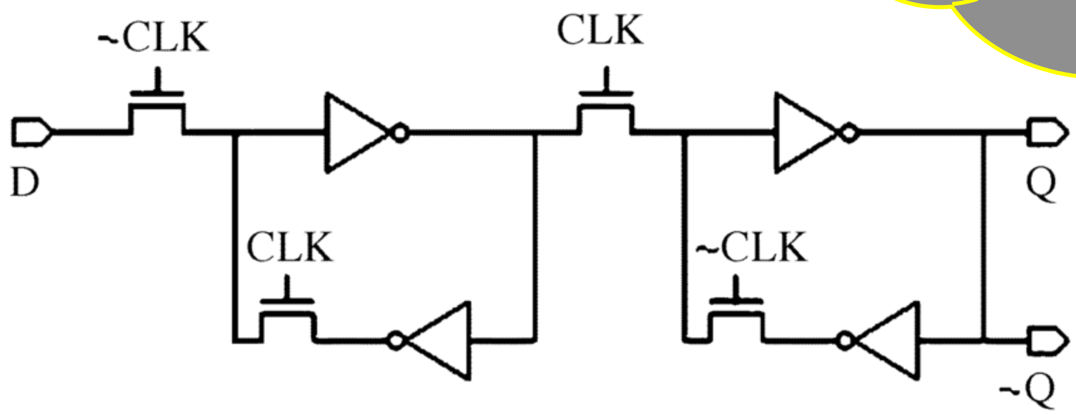


Digital VLSI Systems Components

- Sequential Elements:
  - Implement state
    - Input is stored
    - Output is produced upon spe

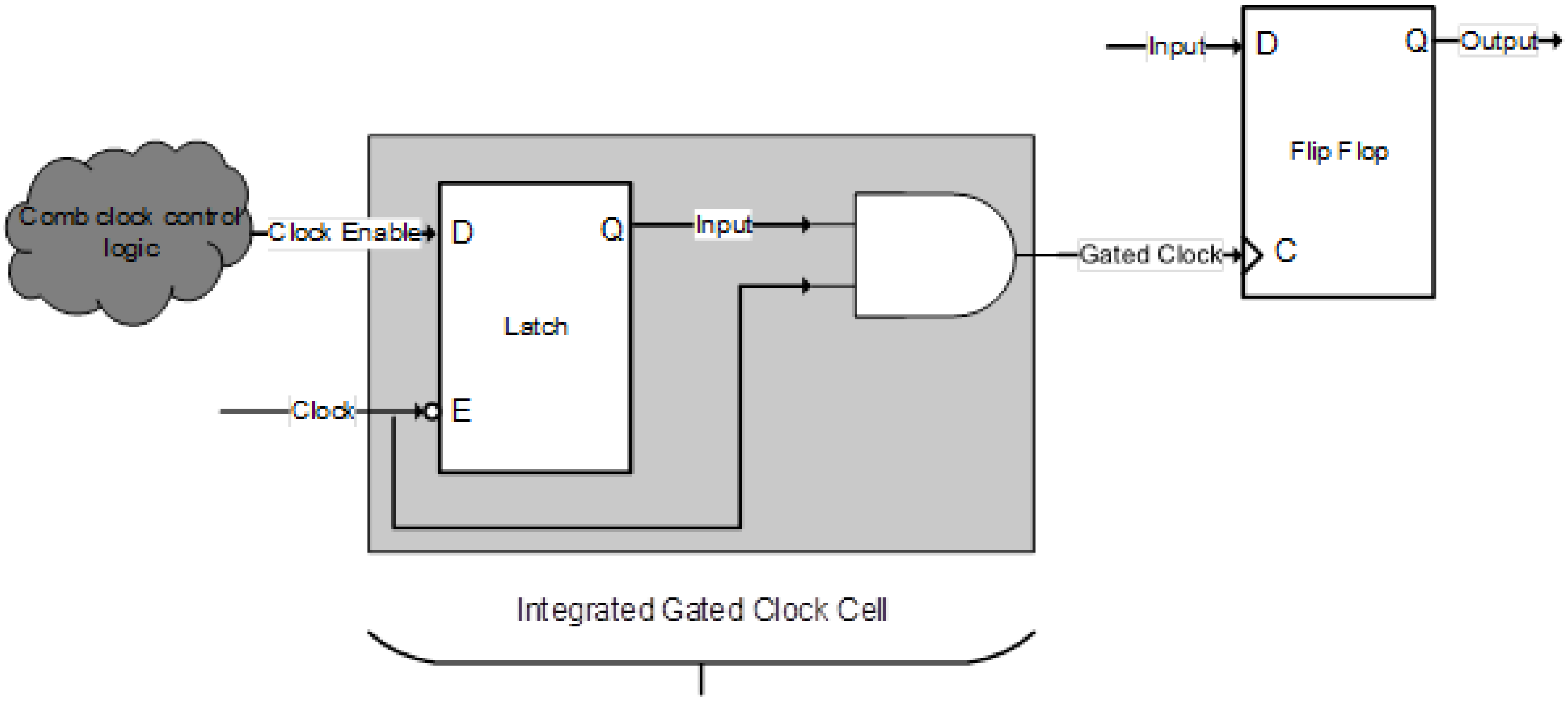
Let’s count transistors:

12



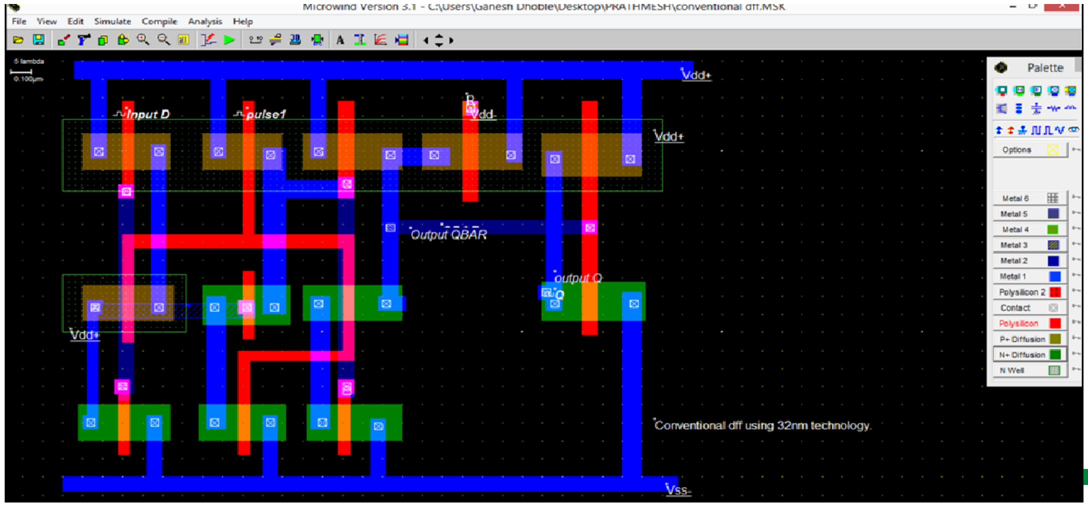
Digital VLSI Systems Components

- Complex:
  - Implement both state and Boolean operations
    - Clock Gates



Part I

- Design DFF, DFFRS standard cells
  - Create stick diagram
  - Create layout
    - Magic or
    - Microwind or
    - Cadence IC (Virtuoso) Design Suite or
    - Synopsys Custom Compiler


Microwind Version 3.1 - C:\Users\Ganesh Dhoble\Desktop\PRATHMESH\conventional dff.MSK
File View Edit Simulate Compile Analysis Help
Vdd+
Input D
pulse1
Vdd
Output QBAR
output Q
Vss-
Conventional dff using 32nm technology.
Palette
Options
Metal 6
Metal 5
Metal 4
Metal 3
Metal 2
Metal 1
Polysilicon 2
Contact
Polysilicon
P+ Diffusion
N+ Diffusion
N Well

Part I

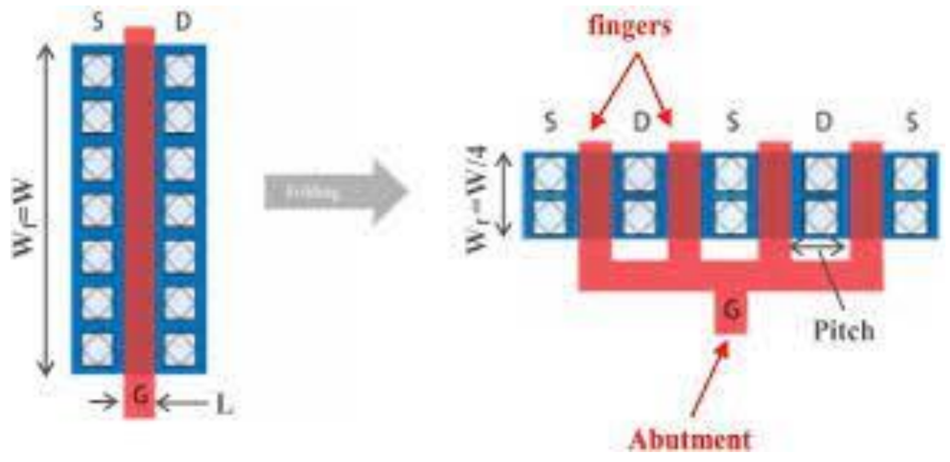
- Design XOR/XNOR standard cells
  - Create stick diagram
  - Create layout
    - Magic or
    - Microwind or
    - Cadence IC (Virtuoso) Design Suite or
    - Synopsys Custom Compiler

Part I

- Layout Extraction
  - Create SPICE netlist
- Verify functionality of all aforementioned standard cells
  - SPICE simulation
    - How?

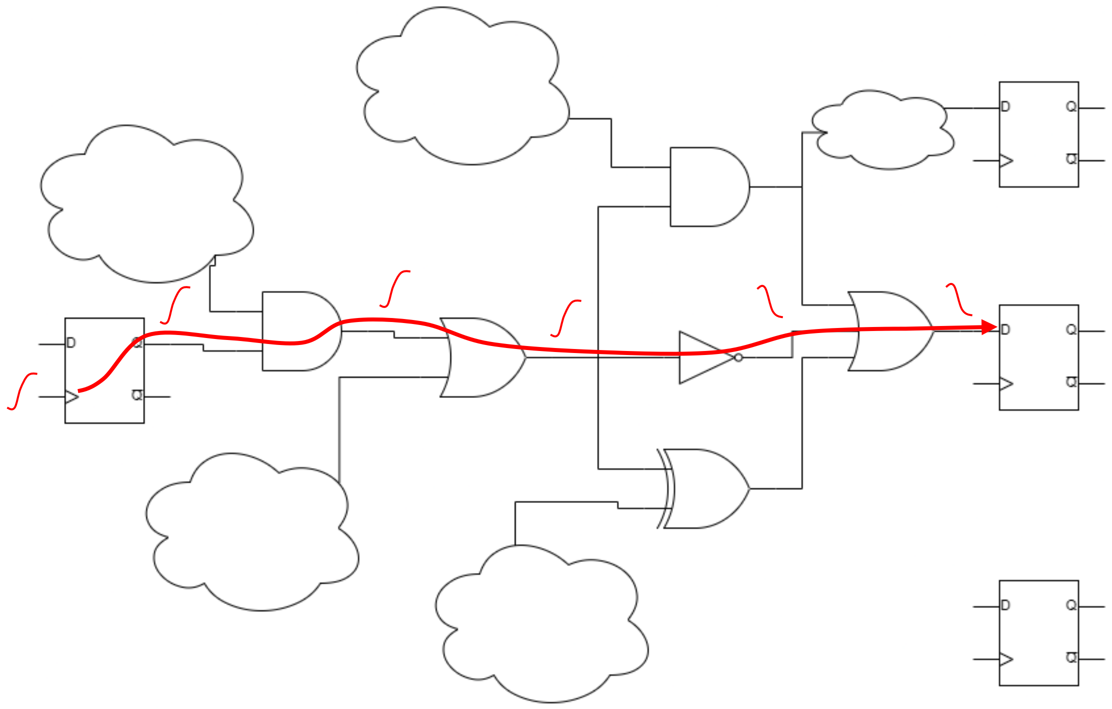
Bonus I

- Design, extract and verify X1, X4 and X12 cells
  - What is “drive”?
  - Differences?



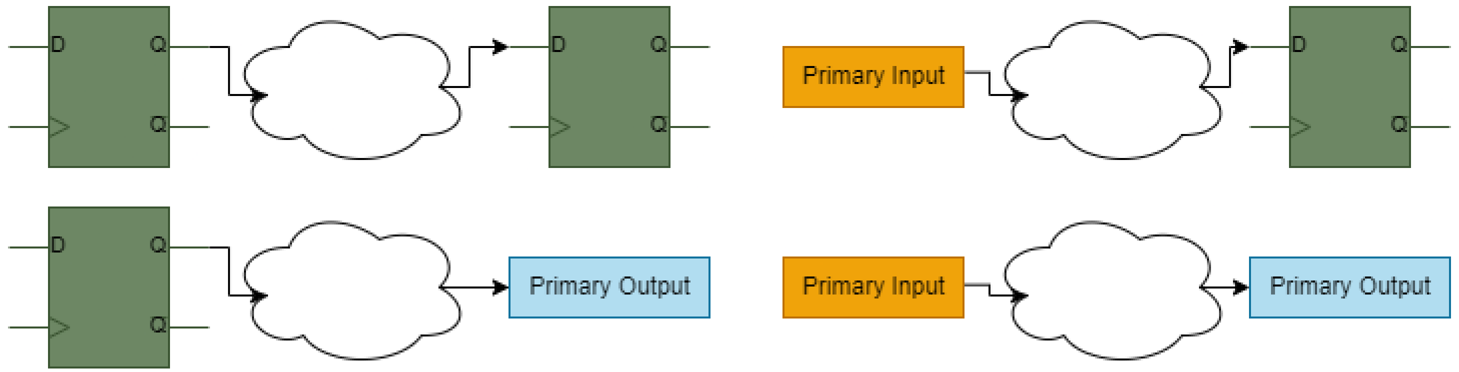
Timing Paths

- Timing paths determine the Performance of our design
  - Clock frequency



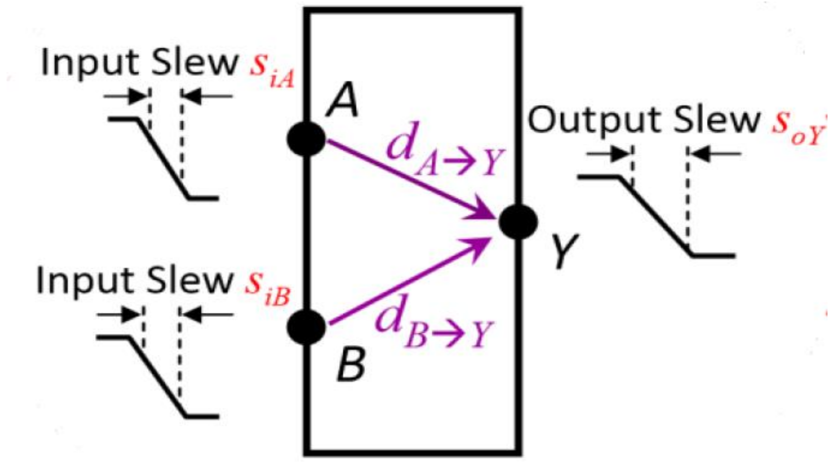
Timing Paths

- Four types
  - From Sequential Element to Sequential Element
  - From Sequential Element to Primary Output
  - From Primary Output to Sequential Element
  - From Primary Input to Primary Output



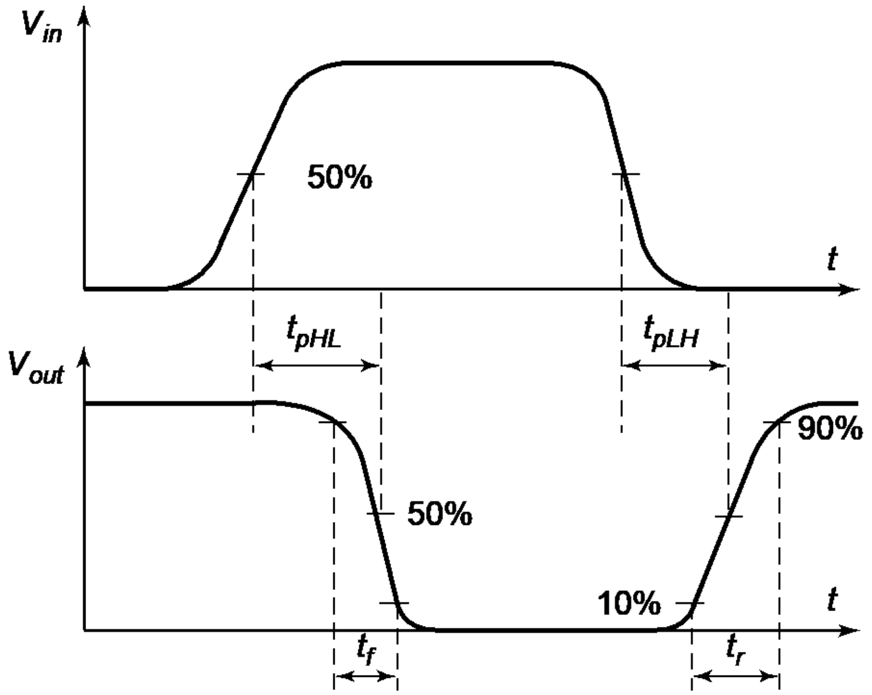
Timing Arcs

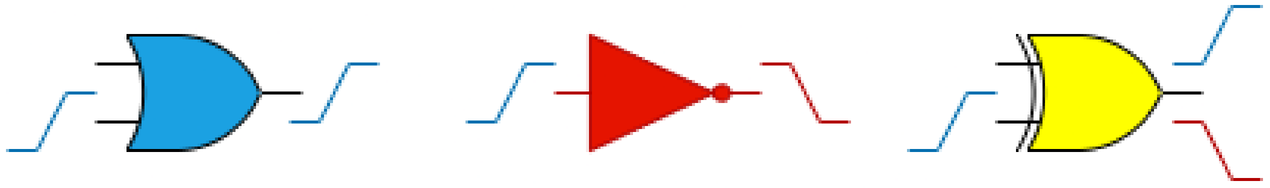
- Describe timing information from component's input to component's output
  - Slew
  - Delay
  - Unateness



Timing Arcs

- Slew
- Delay
- Unateness





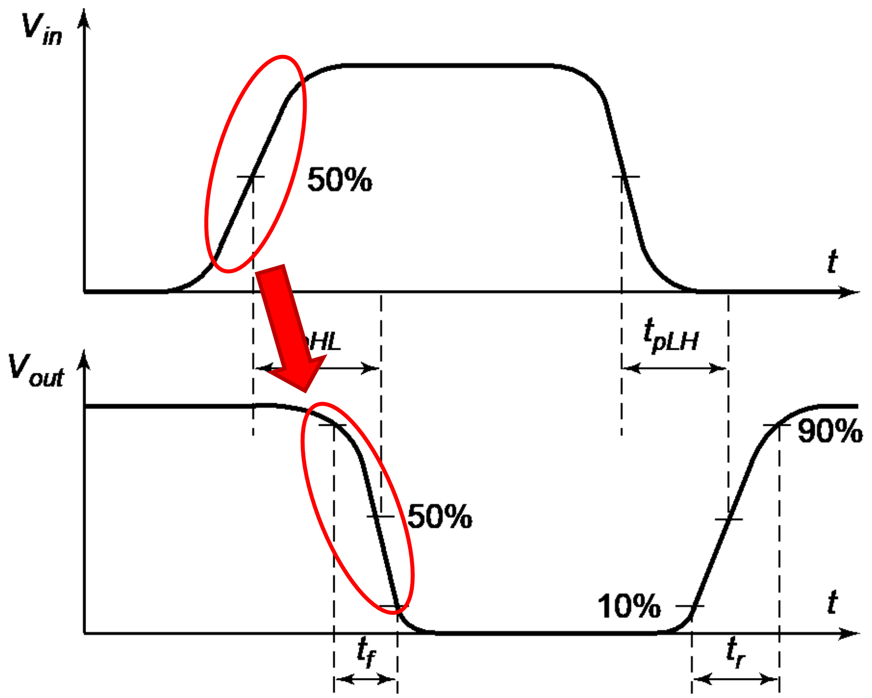
Positive Unate

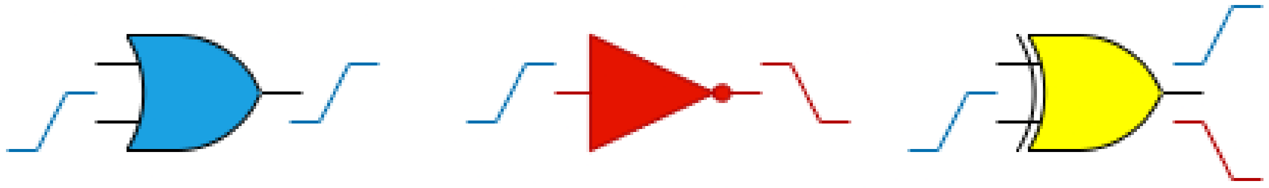
Negative Unate

Binate

Timing Arcs

- Slew
- Delay
- Unateness





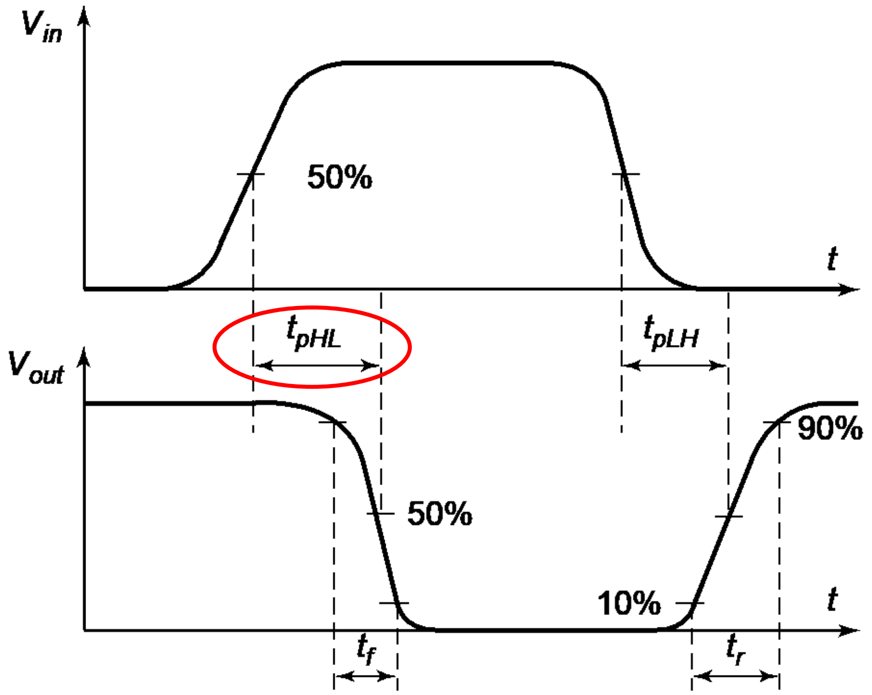
Positive Unate

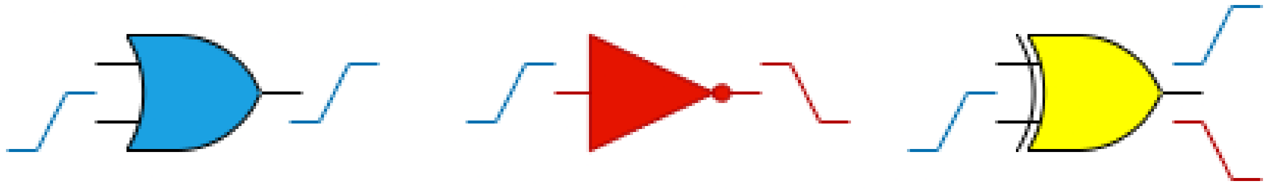
Negative Unate

Binate

Timing Arcs

- Slew
- Delay
- Unateness





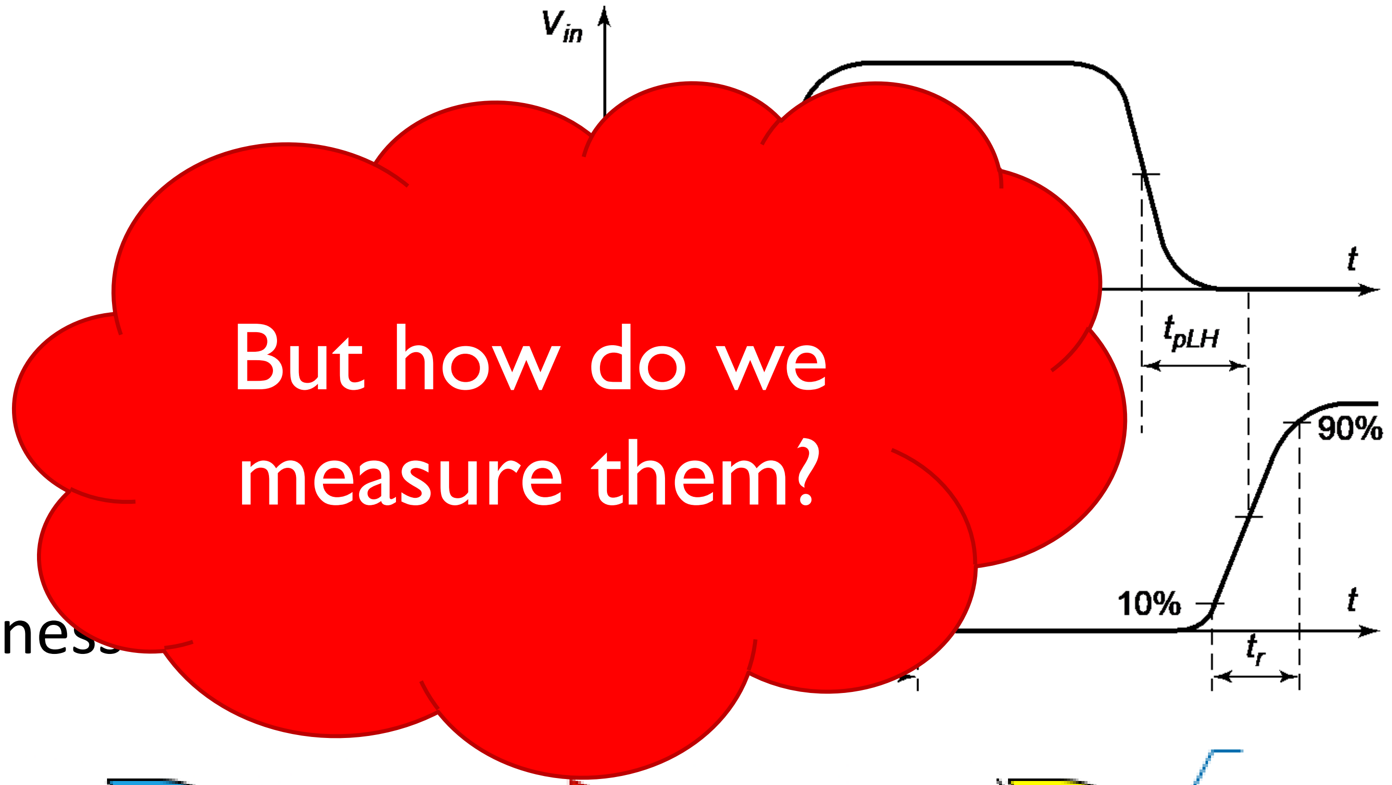
Positive Unate

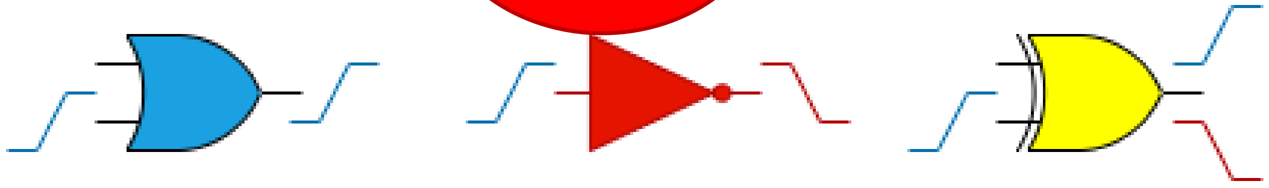
Negative Unate

Binate

Timing Arcs

- Slew
- Delay
- Unateness





Positive Unate

Negative Unate

Binate

SPICE

- SPICE simulates the design at the device level and measures the required timing information
- Multiple SPICE tools exist
  - ng-spice
  - Synopsys HSPICE
  - Cadence Spectre

SPICE vs STA

Spice

- Pros
  - Accuracy
  - Models all electrical phenomena
- Cons
  - Time consuming
    - Nowadays digital designs consist of million gates
  - Requires input vectors

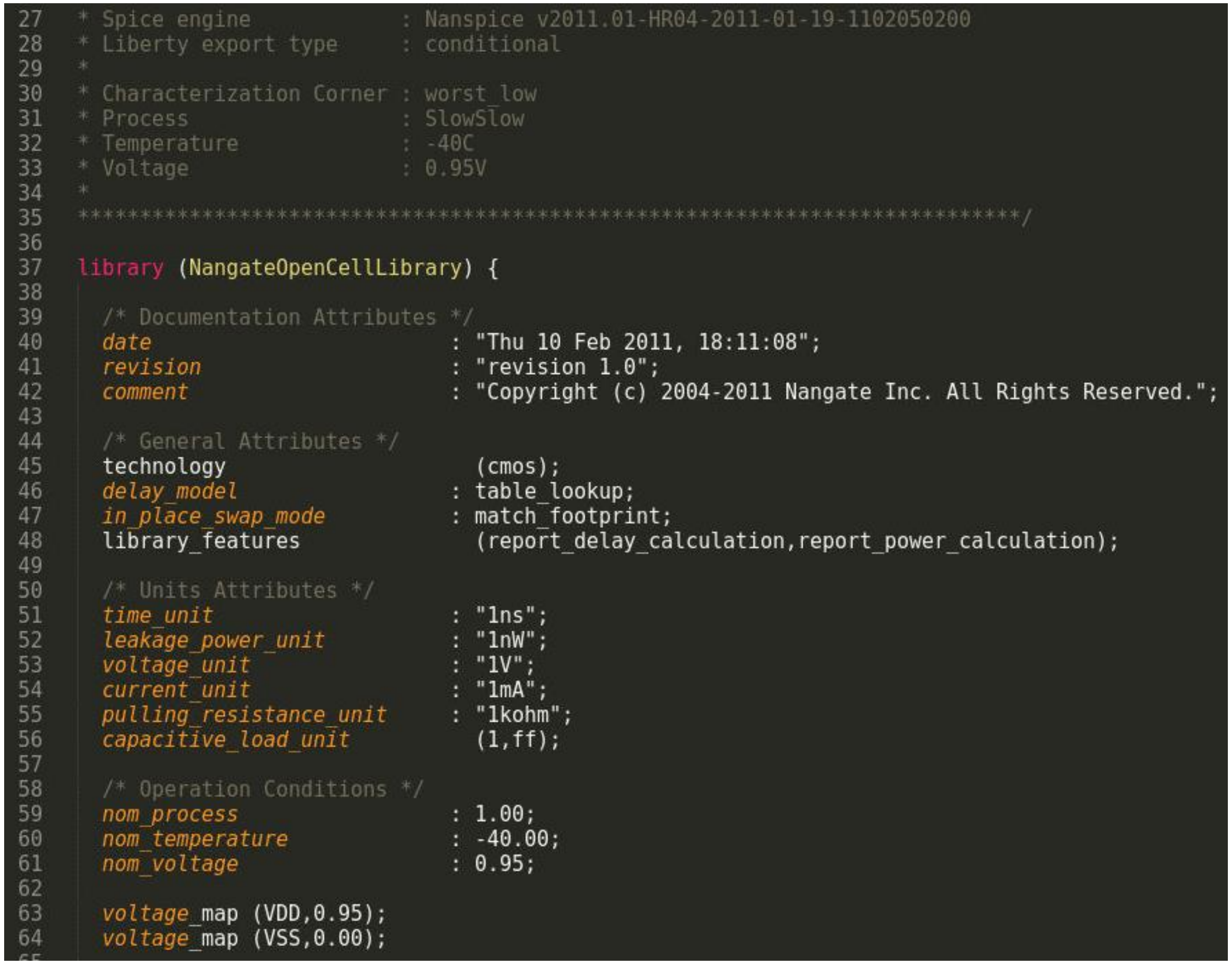
Static Timing Analysis

- Pros
  - Orders of magnitude faster than SPICE
  - Vectorless
- Cons
  - Inflicts Pessimism
  - Circuits are DAG
    - Incompatible with most analog circuits

From SPICE to STA

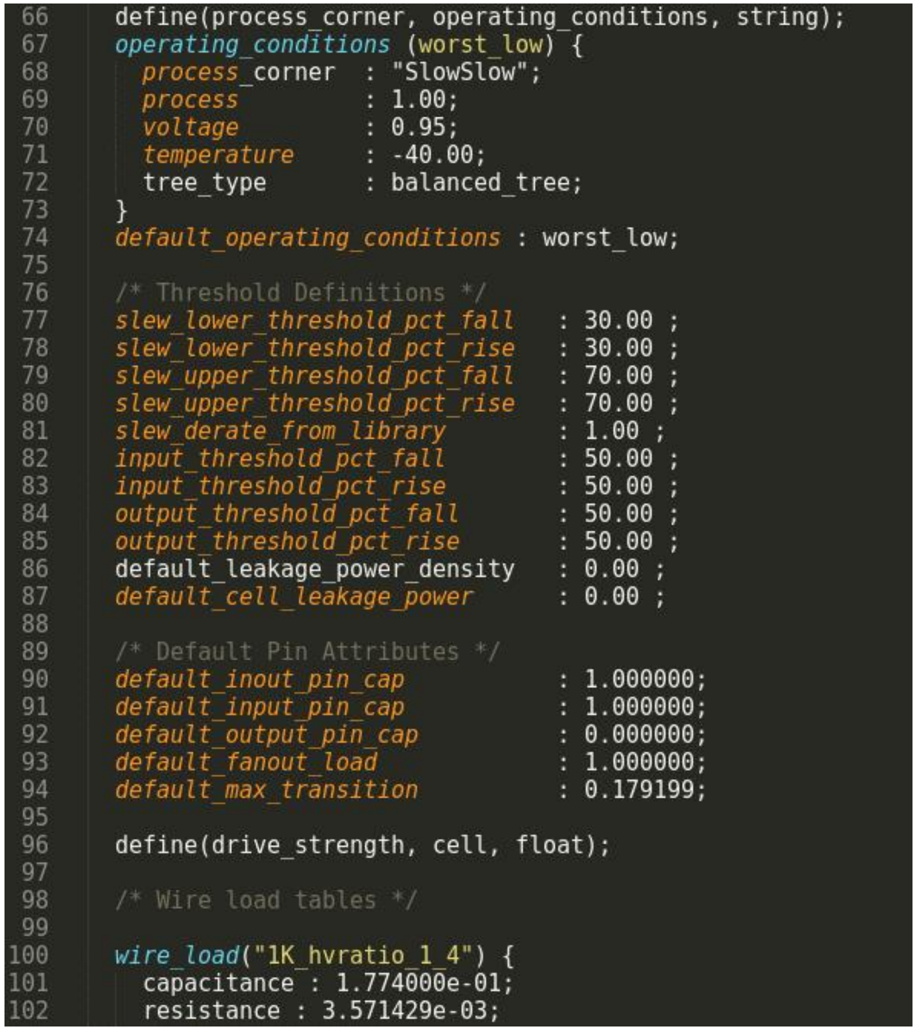
- We use SPICE to **characterize** standard cells
  - Build standard cells layout
  - Perform SPICE Simulation for multiple vectors (cases)
  - Store measurement data in file
    - Liberty File (.lib)
    - A library of standard cells is created
- Then STA is applicable on any design consisted of these standard cells
  - Use .lib file to estimate all cells timing information
  - Perform all timing checks
    - Worst case analysis (setup, recovery)
    - Best case analysis (hold, removal)
    - Dual mode analysis
    - Multi-Mode Multi-Corner (MMMC)
  - Verify no timing violations occur

Liberty File Example



```
* Spice engine             : Nanspice v2011.01-HR04-2011-01-19-1102050200
* Liberty export type      : conditional
*
* Characterization Corner : worst_low
* Process                  : SlowSlow
* Temperature              : -40C
* Voltage                  : 0.95V
*
****************************************************************************/

library (NangateOpenCellLibrary) {

  /* Documentation Attributes */
  date                          : "Thu 10 Feb 2011, 18:11:08";
  revision                      : "revision 1.0";
  comment                       : "Copyright (c) 2004-2011 Nangate Inc. All Rights Reserved.";

  /* General Attributes */
  technology                      (cmos);
  delay_model                   : table_lookup;
  in_place_swap_mode            : match_footprint;
  library_features                (report_delay_calculation,report_power_calculation);

  /* Units Attributes */
  time_unit                     : "1ns";
  leakage_power_unit            : "1nW";
  voltage_unit                  : "1V";
  current_unit                  : "1mA";
  pulling_resistance_unit       : "1kohm";
  capacitive_load_unit            (1,ff);

  /* Operation Conditions */
  nom_process                   : 1.00;
  nom_temperature               : -40.00;
  nom_voltage                   : 0.95;

  voltage_map (VDD,0.95);
  voltage_map (VSS,0.00);
```

Liberty File Example

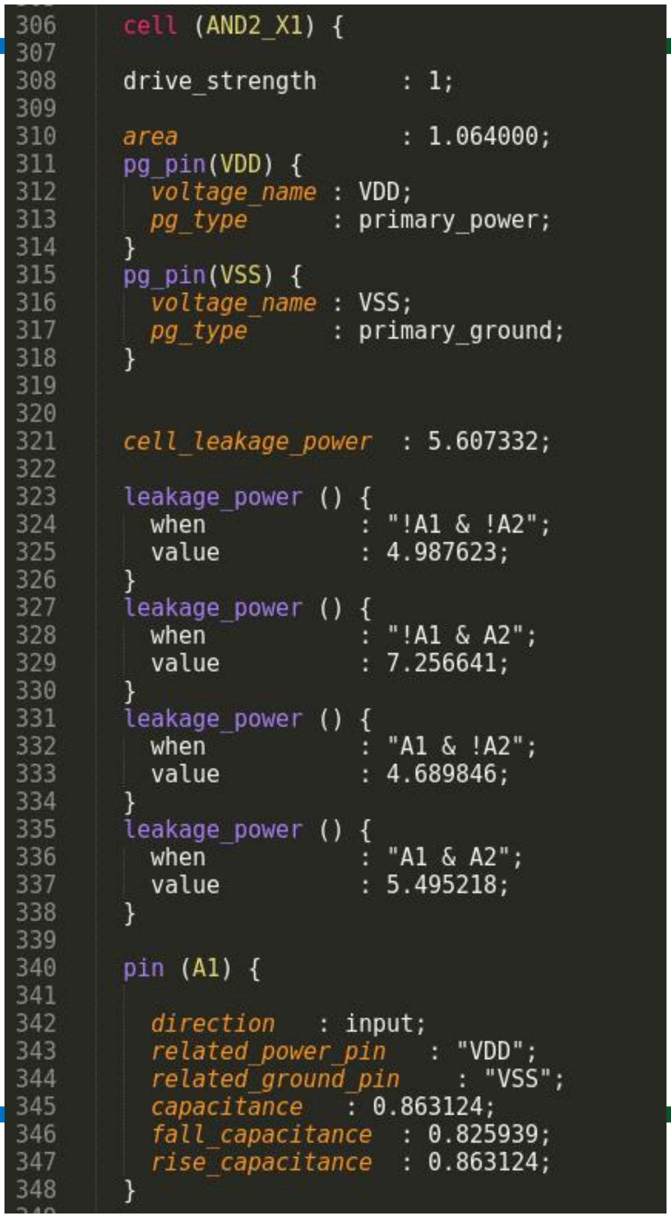


```
define(process_corner, operating_conditions, string);
operating_conditions (worst_low) {
  process_corner  : "SlowSlow";
  process         : 1.00;
  voltage         : 0.95;
  temperature     : -40.00;
  tree_type       : balanced_tree;
}
default_operating_conditions : worst_low;

/* Threshold Definitions */
slew_lower_threshold_pct_fall   : 30.00 ;
slew_lower_threshold_pct_rise   : 30.00 ;
slew_upper_threshold_pct_fall   : 70.00 ;
slew_upper_threshold_pct_rise   : 70.00 ;
slew_derate_from_library        : 1.00 ;
input_threshold_pct_fall        : 50.00 ;
input_threshold_pct_rise        : 50.00 ;
output_threshold_pct_fall       : 50.00 ;
output_threshold_pct_rise       : 50.00 ;
default_leakage_power_density   : 0.00 ;
default_cell_leakage_power      : 0.00 ;

/* Default Pin Attributes */
default_inout_pin_cap           : 1.000000;
default_input_pin_cap           : 1.000000;
default_output_pin_cap          : 0.000000;
default_fanout_load             : 1.000000;
default_max_transition          : 0.179199;

define(drive_strength, cell, float);

/* Wire load tables */

wire_load("1K_hvratio_1_4") {
  capacitance : 1.774000e-01;
  resistance : 3.571429e-03;
```

Liberty File Example

- Standard Cell

```
Library{
  ...
  Cell{
     ...
       Pin {
          ...
            Timing Arc{
                ...
            }
       }
  }
}
```

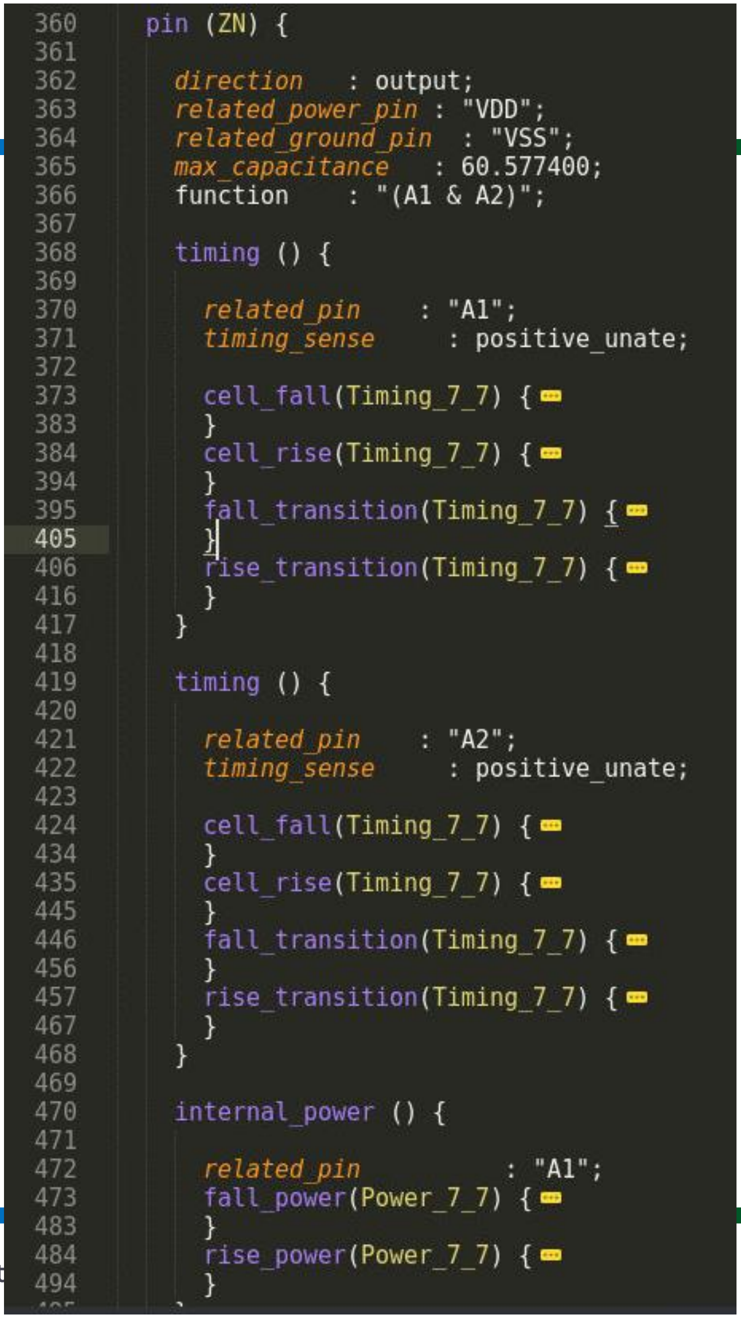


```
cell (AND2_X1) {

drive_strength      : 1;

area                : 1.064000;
pg_pin(VDD) {
  voltage_name : VDD;
  pg_type      : primary_power;
}
pg_pin(VSS) {
  voltage_name : VSS;
  pg_type      : primary_ground;
}


cell_leakage_power  : 5.607332;

leakage_power () {
  when              : "!A1 & !A2";
  value             : 4.987623;
}
leakage_power () {
  when              : "!A1 & A2";
  value             : 7.256641;
}
leakage_power () {
  when              : "A1 & !A2";
  value             : 4.689846;
}
leakage_power () {
  when              : "A1 & A2";
  value             : 5.495218;
}

pin (A1) {

  direction    : input;
  related_power_pin    : "VDD";
  related_ground_pin    : "VSS";
  capacitance    : 0.863124;
  fall_capacitance  : 0.825939;
  rise_capacitance  : 0.863124;
}
```

Liberty File Example

- Standard Cell
  - Output Pin

```
Library{
  …
  Cell{
    …
    Pin {
      …
        Timing Arc{
          …
        }
      }
   }
}
```

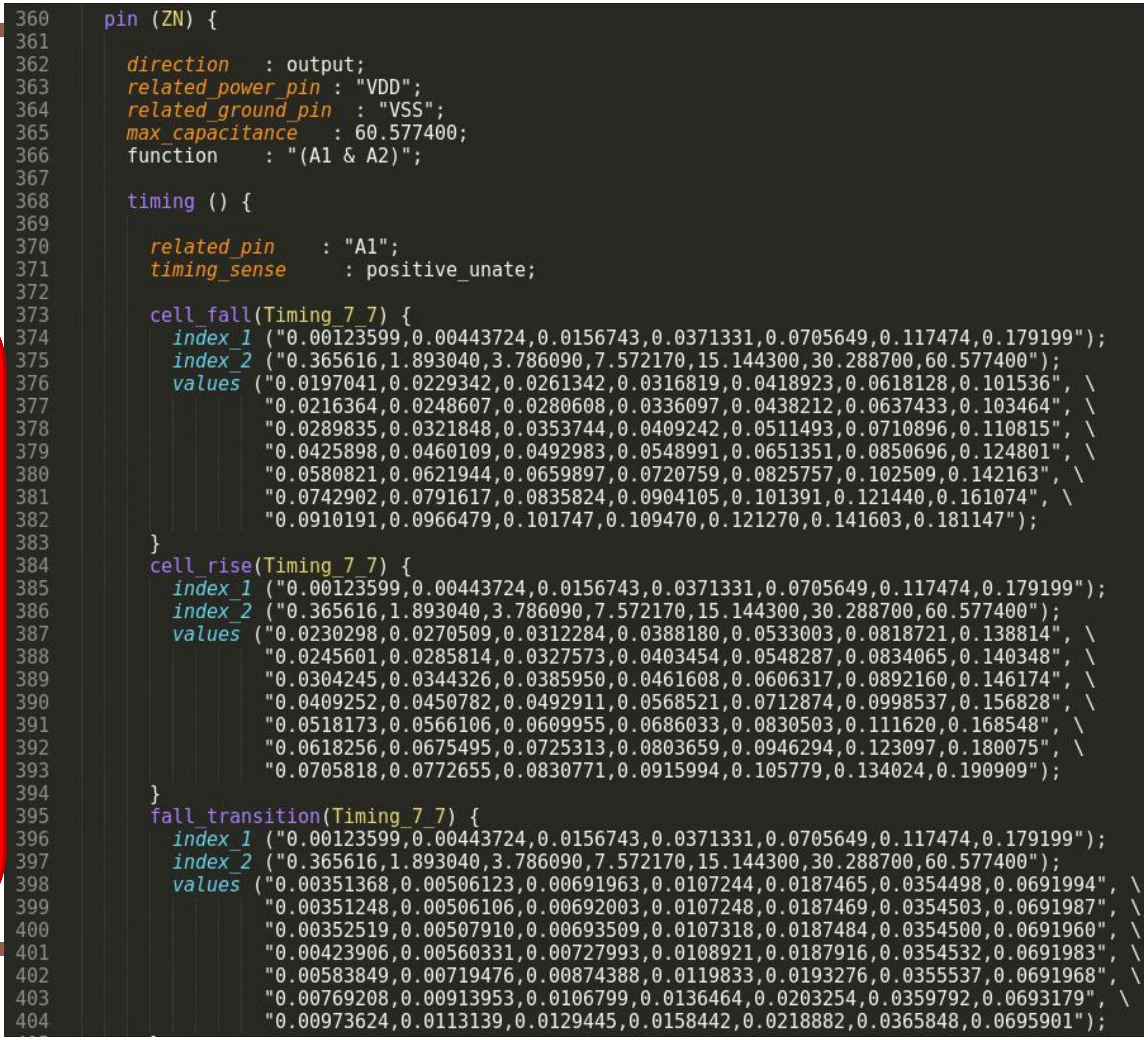


```
pin (ZN) {

  direction      : output;
  related_power_pin : "VDD";
  related_ground_pin   : "VSS";
  max_capacitance     : 60.577400;
  function     : "(A1 & A2)";

  timing () {

    related_pin      : "A1";
    timing_sense       : positive_unate;

    cell_fall(Timing_7_7) {
    }
    cell_rise(Timing_7_7) {
    }
    fall_transition(Timing_7_7) {
    }
    rise_transition(Timing_7_7) {
    }
  }

  timing () {

    related_pin      : "A2";
    timing_sense       : positive_unate;

    cell_fall(Timing_7_7) {
    }
    cell_rise(Timing_7_7) {
    }
    fall_transition(Timing_7_7) {
    }
    rise_transition(Timing_7_7) {
    }
  }

  internal_power () {

    related_pin              : "A1";
    fall_power(Power_7_7) {
    }
    rise_power(Power_7_7) {
    }
```

Liberty File Example

- Standard Cell
  - Output Pin

```
Library{
  …
  Cell{
    …
    Pin {
      …
      Timing Arc{
        …
      }
    }
  }
}
```

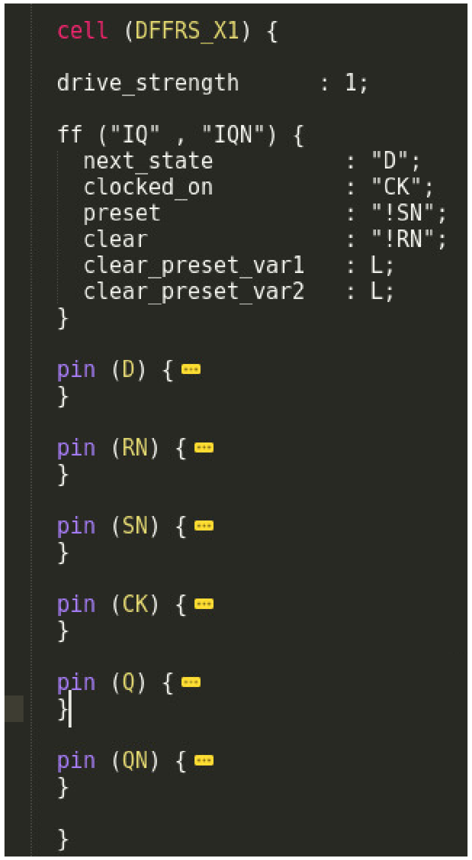


```
pin (ZN) {

  direction    : output;
  related_power_pin : "VDD";
  related_ground_pin  : "VSS";
  max_capacitance     : 60.577400;
  function     : "(A1 & A2)";

  timing () {

    related_pin     : "A1";
    timing_sense      : positive_unate;

    cell_fall(Timing_7_7) {
      index_1 ("0.00123599,0.00443724,0.0156743,0.0371331,0.0705649,0.117474,0.179199");
      index_2 ("0.365616,1.893040,3.786090,7.572170,15.144300,30.288700,60.577400");
      values ("0.0197041,0.0229342,0.0261342,0.0316819,0.0418923,0.0618128,0.101536", \
              "0.0216364,0.0248607,0.0280608,0.0336097,0.0438212,0.0637433,0.103464", \
              "0.0289835,0.0321848,0.0353744,0.0409242,0.0511493,0.0710896,0.110815", \
              "0.0425898,0.0460109,0.0492983,0.0548991,0.0651351,0.0850696,0.124801", \
              "0.0580821,0.0621944,0.0659897,0.0720759,0.0825757,0.102509,0.142163", \
              "0.0742902,0.0791617,0.0835824,0.0904105,0.101391,0.121440,0.161074", \
              "0.0910191,0.0966479,0.101747,0.109470,0.121270,0.141603,0.181147");
    }
    cell_rise(Timing_7_7) {
      index_1 ("0.00123599,0.00443724,0.0156743,0.0371331,0.0705649,0.117474,0.179199");
      index_2 ("0.365616,1.893040,3.786090,7.572170,15.144300,30.288700,60.577400");
      values ("0.0230298,0.0270509,0.0312284,0.0388180,0.0533003,0.0818721,0.138814", \
              "0.0245601,0.0285814,0.0327573,0.0403454,0.0548287,0.0834065,0.140348", \
              "0.0304245,0.0344326,0.0385950,0.0461608,0.0606317,0.0892160,0.146174", \
              "0.0409252,0.0450782,0.0492911,0.0568521,0.0712874,0.0998537,0.156828", \
              "0.0518173,0.0566106,0.0609955,0.0686033,0.0830503,0.111620,0.168548", \
              "0.0618256,0.0675495,0.0725313,0.0803659,0.0946294,0.123097,0.180075", \
              "0.0705818,0.0772655,0.0830771,0.0915994,0.105779,0.134024,0.190909");
    }
    fall_transition(Timing_7_7) {
      index_1 ("0.00123599,0.00443724,0.0156743,0.0371331,0.0705649,0.117474,0.179199");
      index_2 ("0.365616,1.893040,3.786090,7.572170,15.144300,30.288700,60.577400");
      values ("0.00351368,0.00506123,0.00691963,0.0107244,0.0187465,0.0354498,0.0691994", \
              "0.00351248,0.00506106,0.00692003,0.0107248,0.0187469,0.0354503,0.0691987", \
              "0.00352519,0.00507910,0.00693509,0.0107318,0.0187484,0.0354500,0.0691960", \
              "0.00423906,0.00560331,0.00727993,0.0108921,0.0187916,0.0354532,0.0691983", \
              "0.00583849,0.00719476,0.00874388,0.0119833,0.0193276,0.0355537,0.0691968", \
              "0.00769208,0.00913953,0.0106799,0.0136464,0.0203254,0.0359792,0.0693179", \
              "0.00973624,0.0113139,0.0129445,0.0158442,0.0218882,0.0365848,0.0695901");
```

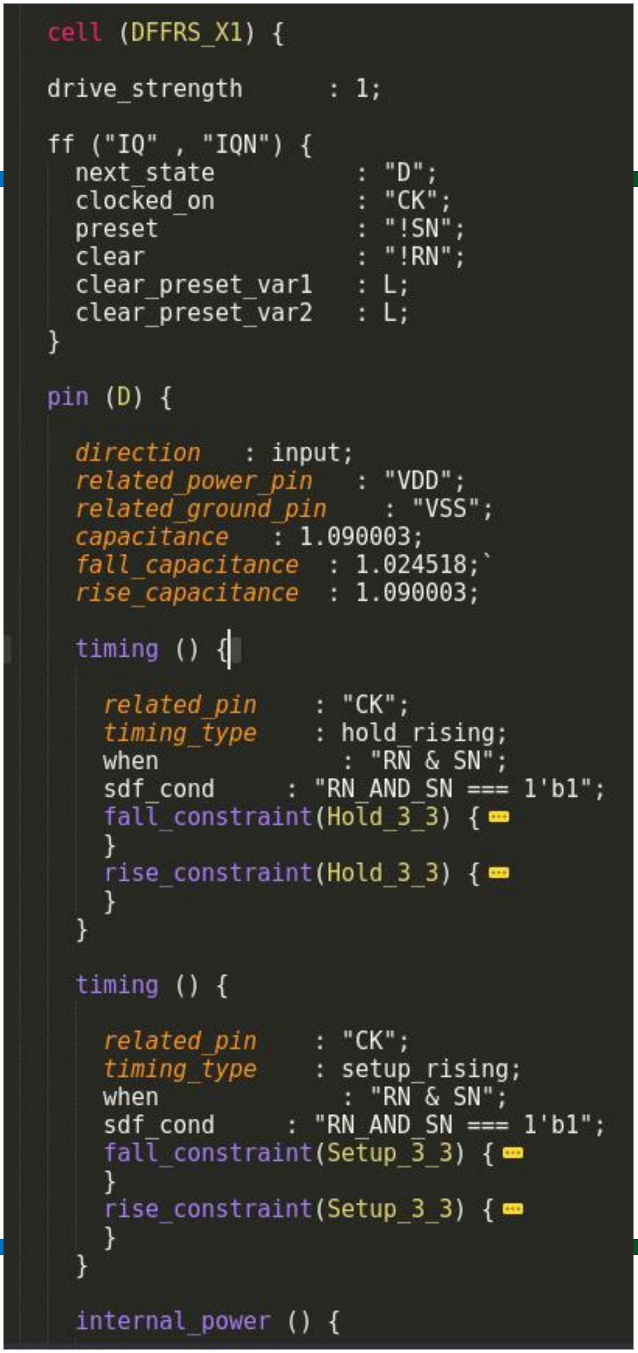
Liberty File Example

- Standard Cell
  - Sequential cell
    - Constraint Arcs

```
Library{
  ...
  Cell{
    ...
      Pin {
        ...
          Timing Arc{
            ...
          }
      }
  }
}
```



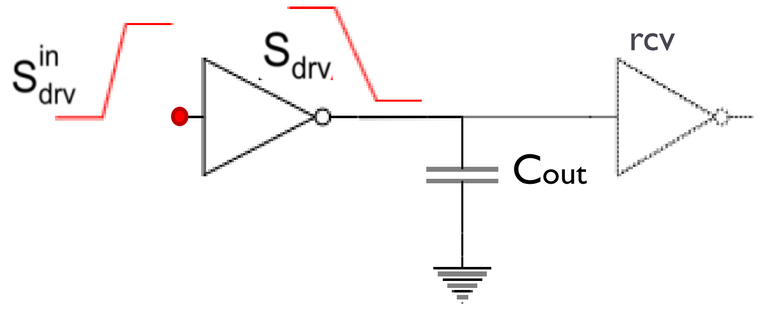
```
cell (DFFRS_X1) {

drive_strength      : 1;

ff ("IQ" , "IQN") {
  next_state            : "D";
  clocked_on            : "CK";
  preset                : "!SN";
  clear                 : "!RN";
  clear_preset_var1     : L;
  clear_preset_var2     : L;
}

pin (D) {
}

pin (RN) {
}

pin (SN) {
}

pin (CK) {
}

pin (Q) {
}

pin (QN) {
}

}
```

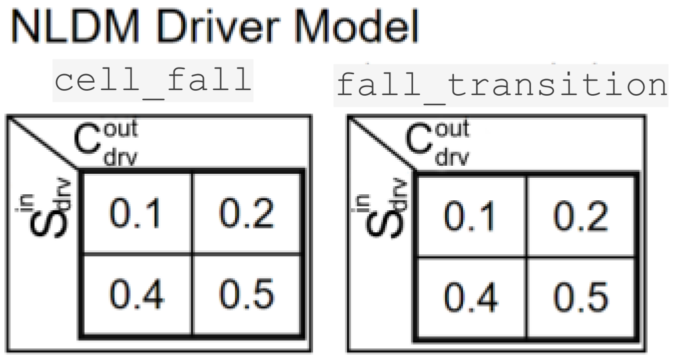


```
cell (DFFRS_X1) {

drive_strength      : 1;

ff ("IQ" , "IQN") {
  next_state            : "D";
  clocked_on            : "CK";
  preset                : "!SN";
  clear                 : "!RN";
  clear_preset_var1     : L;
  clear_preset_var2     : L;
}

pin (D) {

  direction    : input;
  related_power_pin     : "VDD";
  related_ground_pin     : "VSS";
  capacitance    : 1.090003;
  fall_capacitance   : 1.024518;`
  rise_capacitance   : 1.090003;

  timing () {

    related_pin      : "CK";
    timing_type      : hold_rising;
    when               : "RN & SN";
    sdf_cond       : "RN_AND_SN === 1'b1";
    fall_constraint(Hold_3_3) {
    }
    rise_constraint(Hold_3_3) {
    }
  }

  timing () {

    related_pin      : "CK";
    timing_type      : setup_rising;
    when               : "RN & SN";
    sdf_cond       : "RN_AND_SN === 1'b1";
    fall_constraint(Setup_3_3) {
    }
    rise_constraint(Setup_3_3) {
    }
  }

  internal_power () {
```

Timing Calculations

- Output slew and delay calculation using .lib data
- Calculation is performed based on
  - Input slew
  - Output load




NLDM Driver Model

`cell_fall`

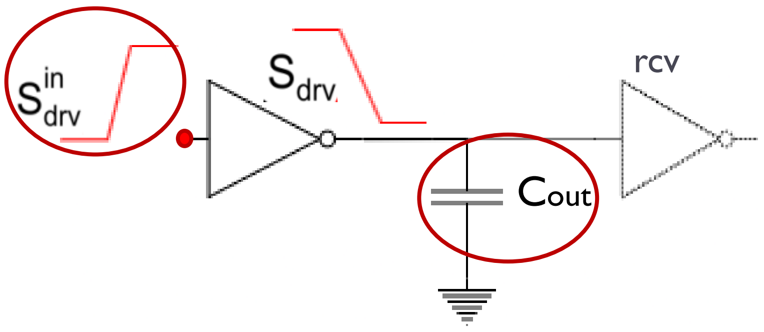
S_{drv}^{in} \ C_{drv}^{out}		
	0.1	0.2
	0.4	0.5

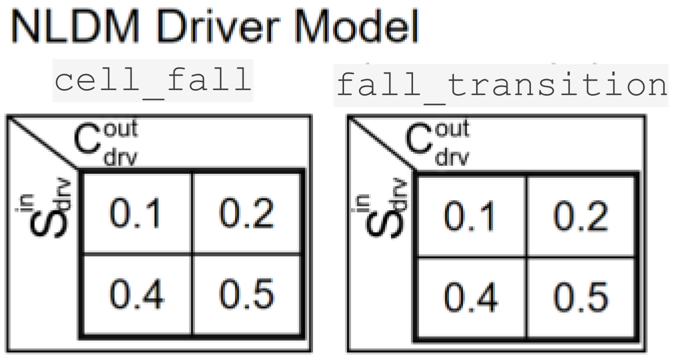
`fall_transition`

S_{drv}^{in} \ C_{drv}^{out}		
	0.1	0.2
	0.4	0.5

Timing Calculations

- Output slew and delay calculation using .lib data
- Calculation is performed based on
  - Input slew
  - Output load





NLDM Driver Model

cell_fall

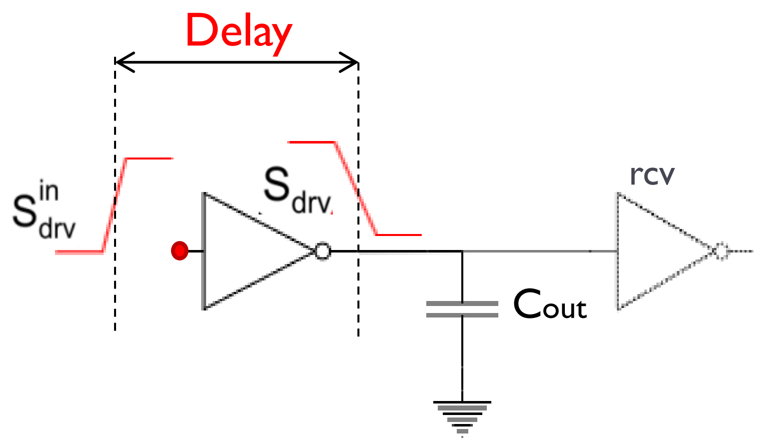
S^{in}_{drv} \ C^{out}_{drv}		
	0.1	0.2
	0.4	0.5

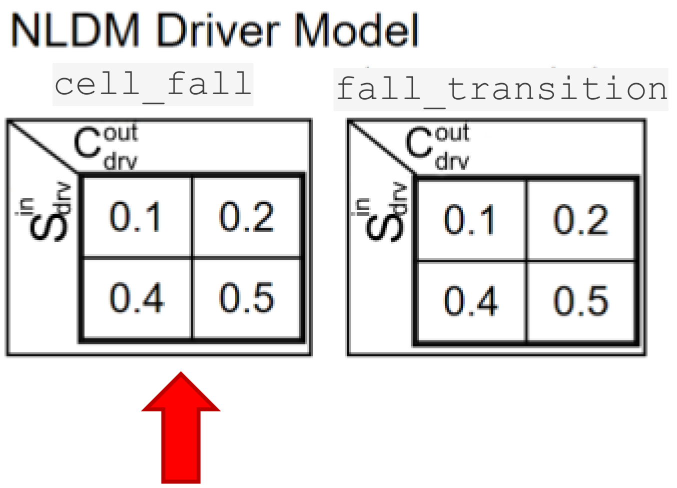
fall_transition

S^{in}_{drv} \ C^{out}_{drv}		
	0.1	0.2
	0.4	0.5

Timing Calculations

- Output slew and delay calculation using .lib data
- Calculation is performed based on
  - Input slew
  - Output load




NLDM Driver Model

`cell_fall`

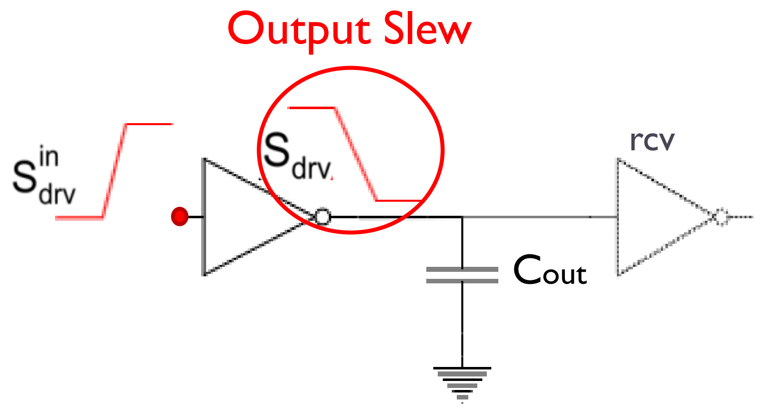
S_{drv}^{in} \ C_{drv}^{out}		
	0.1	0.2
	0.4	0.5

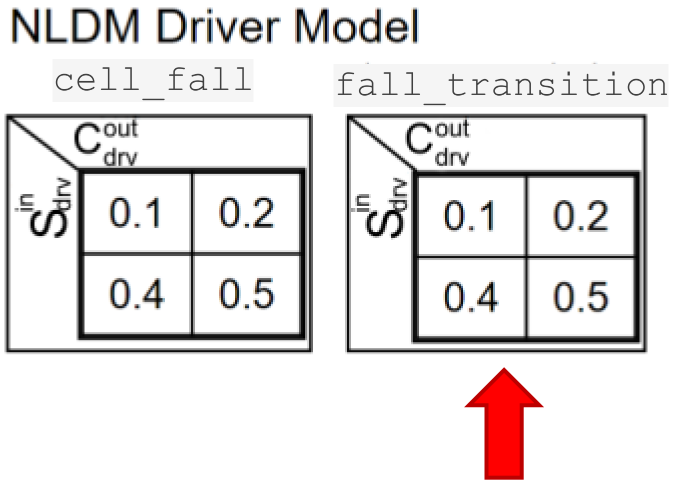
`fall_transition`

S_{drv}^{in} \ C_{drv}^{out}		
	0.1	0.2
	0.4	0.5

Timing Calculations

- Output slew and delay calculation using .lib data
- Calculation is performed based on
  - Input slew
  - Output load




NLDM Driver Model

cell_fall

S_{drv}^{in} \ C_{drv}^{out}		
	0.1	0.2
	0.4	0.5

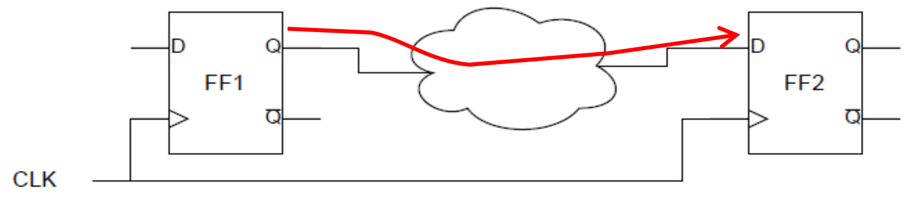
fall_transition

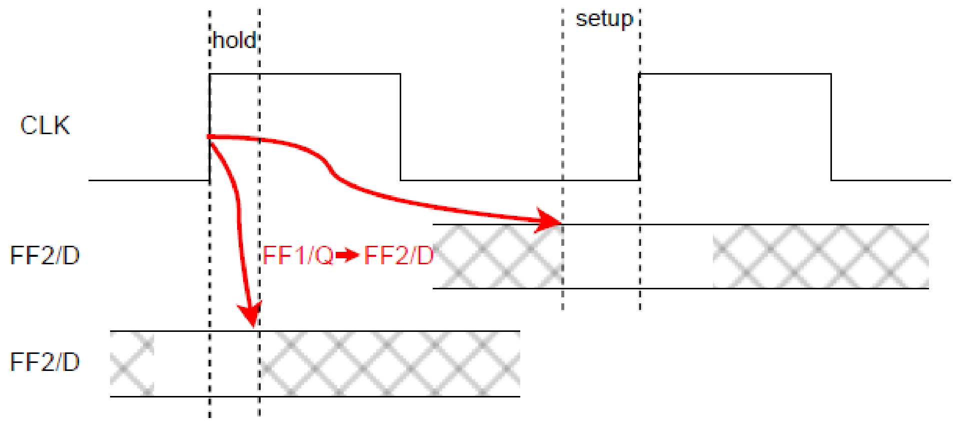
S_{drv}^{in} \ C_{drv}^{out}		
	0.1	0.2
	0.4	0.5

Timing Checks

- Timing constraints are imposed to Sequential std. cells
  - To ensure correct functionality
- Such constraints are Setup, Hold, Recovery and Removal time
  - STA tools perform timing checks to ensure no violations will happen

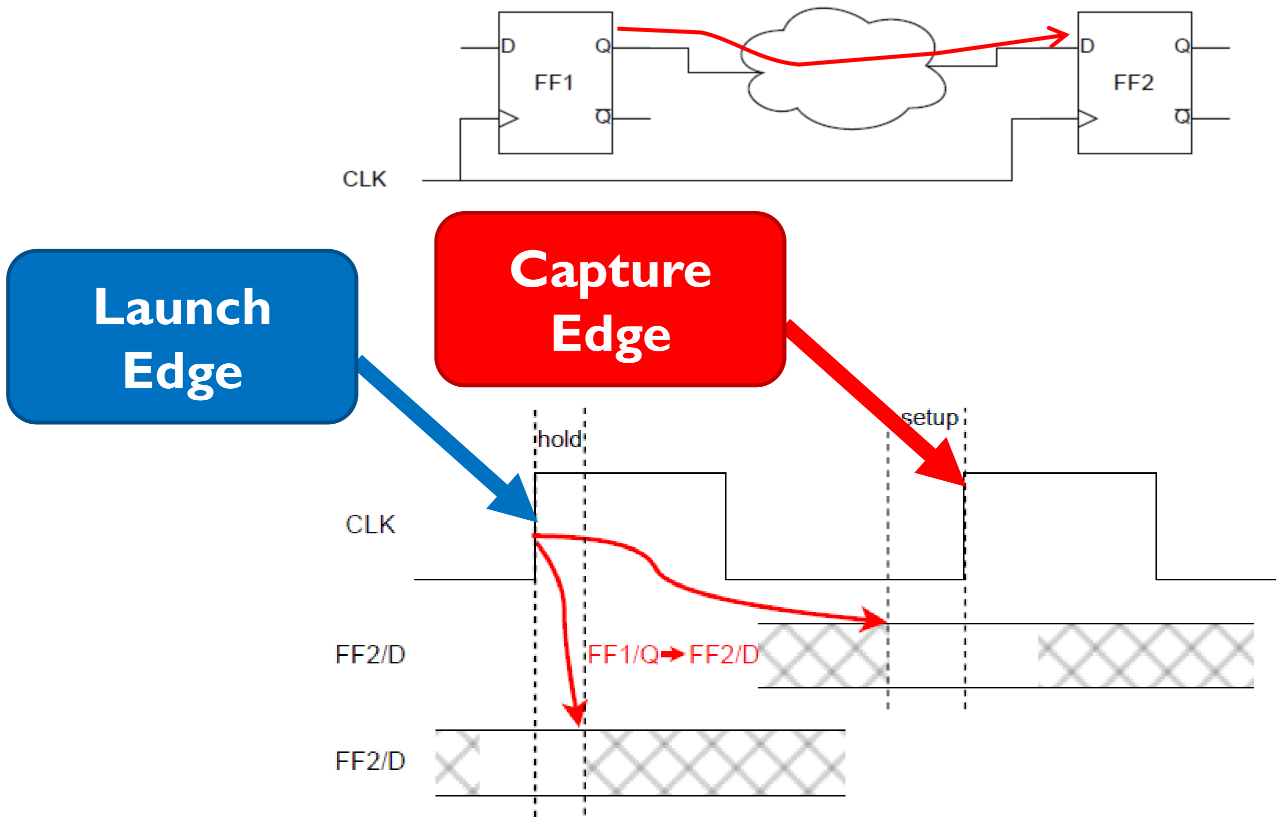
Data Launch and Capture





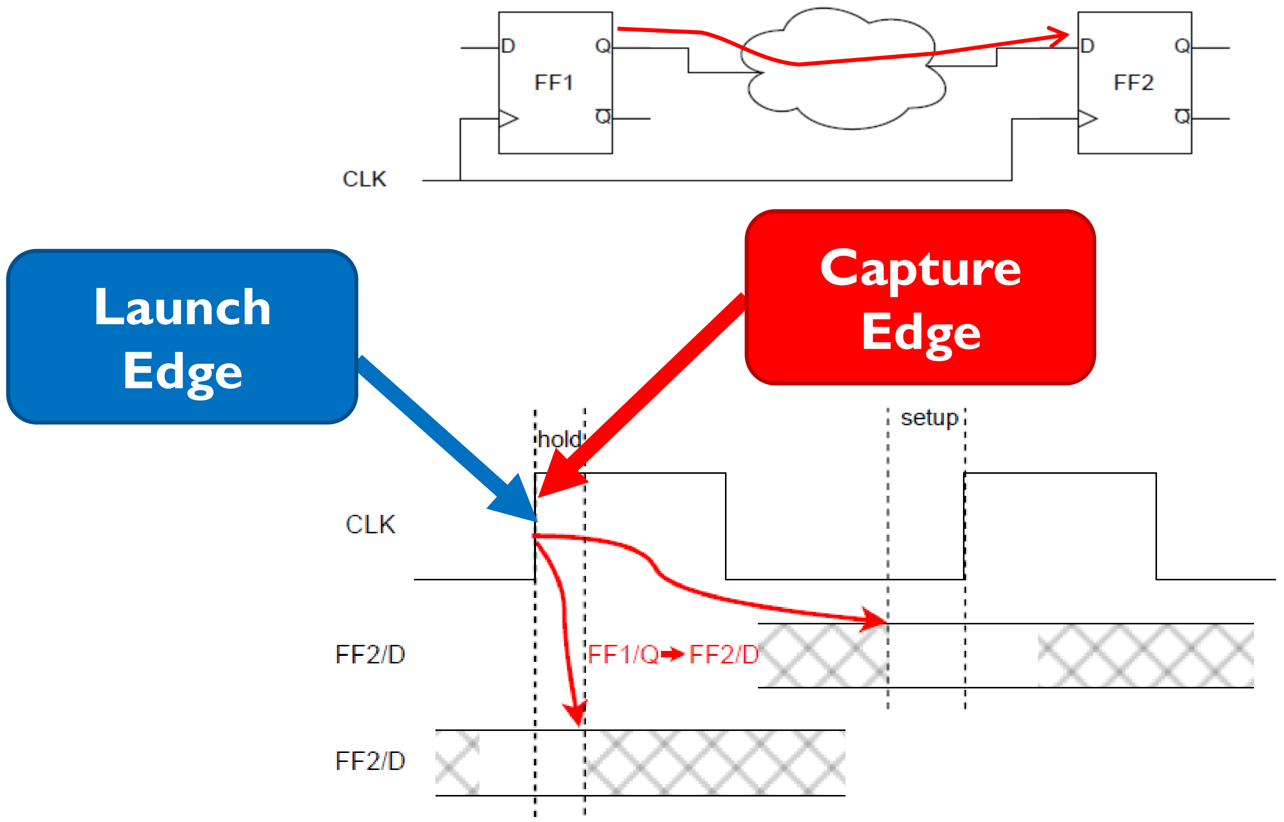
Data Launch and Capture

- For Setup Timing Check:



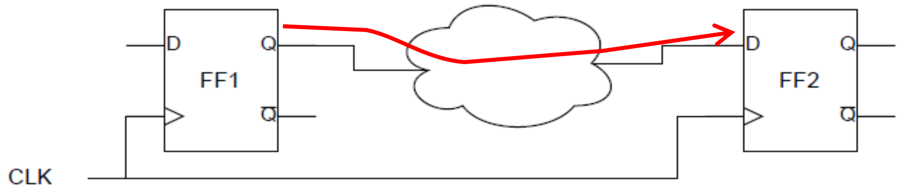
Data Launch and Capture

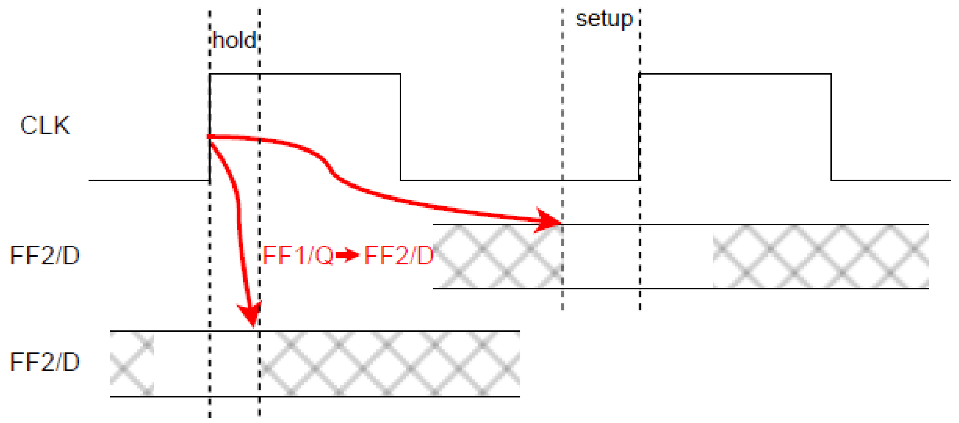
- For Hold Timing Check:



Timing Checks

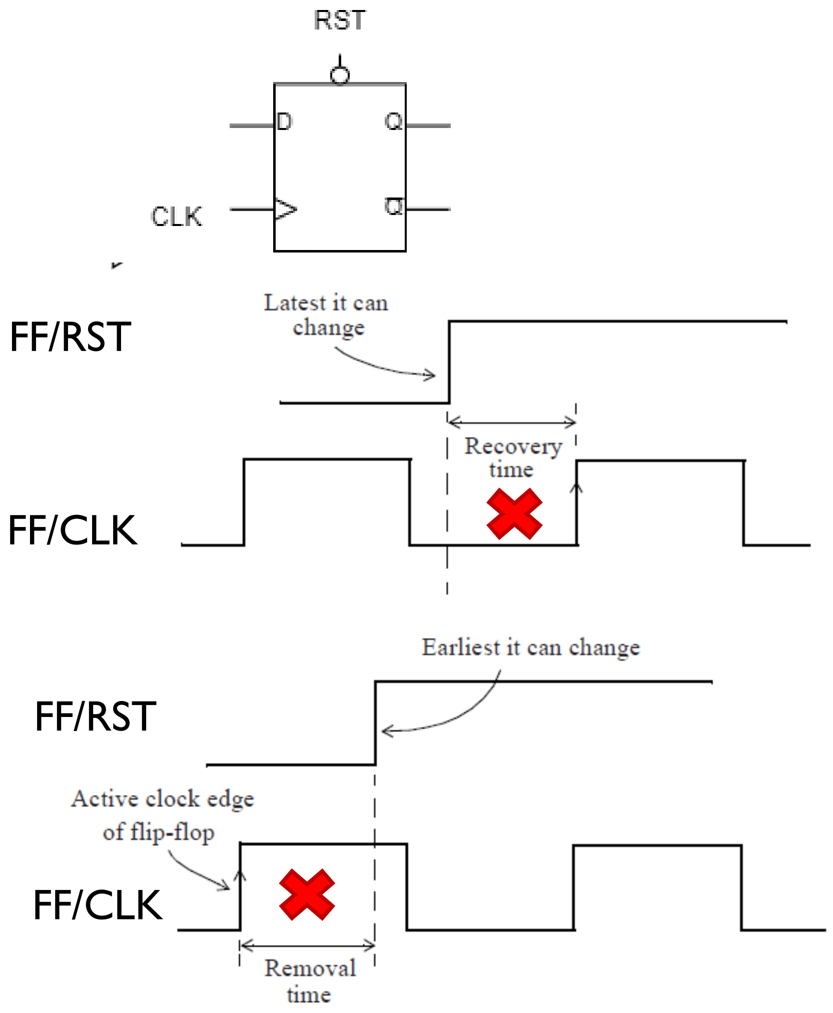
- Setup timing check
  - Data must arrive to path endpoint, before a specific time margin from next clock active edge
- Hold timing check
  - Data must remain stable at path endpoint, for a specific time margin after the same clock active edge, which triggered data launch





Timing Checks

- Recovery timing check
  - Reset should arrive before a specific time margin from next active clock edge
  - Resembles Setup timing check
- Removal timing check
  - Reset should arrive after a specific time margin from last active clock edge
  - Resembles Hold timing check



Part II

- Create C/C++/Python/Bash program
  - Use extracted SPICE deck (magic, ext2spice) from Part I
  - Run SPICE for multiple stimulus and load combinations
    - 1 run per input slew, output load pair
    - Measure delay and slew
    - Store measurement data
    - Tools:
      - ng-spice/Xyce/HSPICE/Spectre
  - Calculate setup, hold, recovery, removal times for Flip-Flops
    - Trial and error methodology
  - Write measurement data to your own custom Liberty File (.lib)
    - You will be provided with a template Liberty File
    - You will only need to write specific fields, NOT the entire file from scratch

Verilog

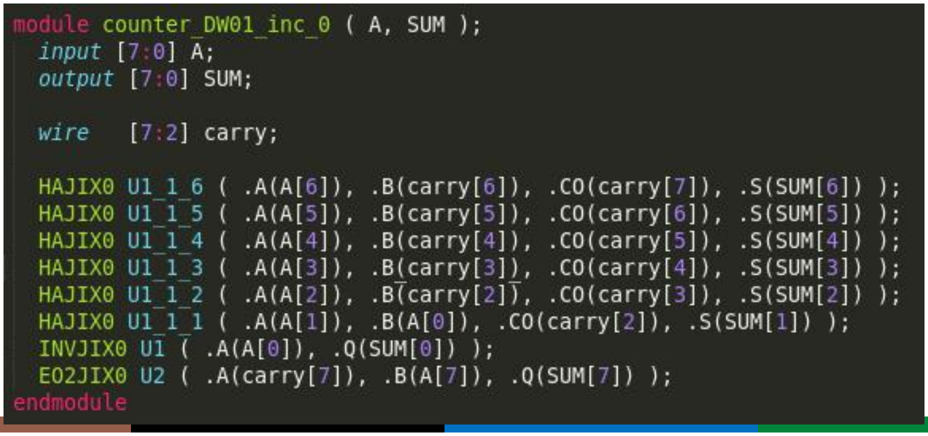
- Verilog is one of the 2 most widespread Hardware Description Languages (HDLs)

Verilog

- Verilog is one of the 2 most widespread Hardware Description Languages (HDLs)
  - The other one being VHDL

Verilog

- Verilog is one of the 2 most widespread Hardware Description Languages (HDLs)
  - The other one being VHDL
- We will only use it in its Gate-Level Netlist form
  - Wires
  - Standard cell instantiations
  - I/O ports and top module declaration



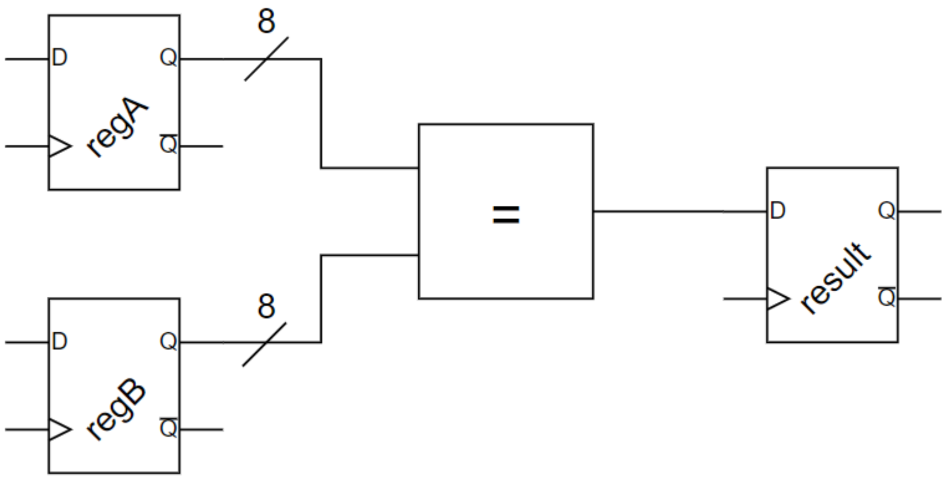
```
module counter_DW01_inc_0 ( A, SUM );
  input [7:0] A;
  output [7:0] SUM;

  wire   [7:2] carry;

  HAJIX0 U1_1_6 ( .A(A[6]), .B(carry[6]), .CO(carry[7]), .S(SUM[6]) );
  HAJIX0 U1_1_5 ( .A(A[5]), .B(carry[5]), .CO(carry[6]), .S(SUM[5]) );
  HAJIX0 U1_1_4 ( .A(A[4]), .B(carry[4]), .CO(carry[5]), .S(SUM[4]) );
  HAJIX0 U1_1_3 ( .A(A[3]), .B(carry[3]), .CO(carry[4]), .S(SUM[3]) );
  HAJIX0 U1_1_2 ( .A(A[2]), .B(carry[2]), .CO(carry[3]), .S(SUM[2]) );
  HAJIX0 U1_1_1 ( .A(A[1]), .B(A[0]), .CO(carry[2]), .S(SUM[1]) );
  INVJIX0 U1 ( .A(A[0]), .Q(SUM[0]) );
  EO2JIX0 U2 ( .A(carry[7]), .B(A[7]), .Q(SUM[7]) );
endmodule
```

Part III

- Use the std. cells you created and characterized in Parts I & II and create a Verilog module which contains
  - 2 8-bit registers
  - A comparator which compares the above registers
  - 1 register to store the comparator result

Part III

- Use the std. cells you created and characterized in Parts I & II and create a Verilog module which contains
  - 2 8-bit registers
  - A comparator which compares the above registers
  - 1 register to store the comparator result



Part III

- Create Verilog `‘specify’` blocks for the std. cells you created
- Create Testbench and make sure your design is functioning properly
  - Tools: Modelsim
- Use an STA engine to analyze timing
  - We will provide you with the script
  - Tools:
    - OpenROAD OpenSTA
    - Synopsys PrimeTime
    - Cadence Tempus
- Find the critical path of your design

Bonus II

- Use the higher drive std. cells you created in Bonus I
  - Optimize your design for highest possible performance
  - Try to perform area recovery
- Comment on your thought process and efforts to achieve the above