Lec 01

Profiling is a form of dynamic program analysis that measures the space/time complexity of a program to aid program optimization. It could be implemented in H.W.

FPGA: No fabrication needed, limited routing resource.

ASIC: • Cell-based Design flow
- use pre-designed logic cells known as standard cells and micro-cells e.g microcontroller.

• Full-Custom Design Flow
- Design everything by yourself.

Cell based Design Flow:

Specification Development & System models

↓

RTL code development & Functional Verification

↓

Synthesis & Timing Verification

↓

Physical Synthesis / Place and Route / Physical Verification

↓

Prototype / Build and Test.

1

Design a Calculator — Specify I/O

RTL Code — 1. + 2. − 3. × 4. ÷ ... — Write RTL code

Pre-Sim — Make sure the calc. is working — Cadence ncverilog, irun, Synopsys Verdi (nWave)

2.

Choose suitable components from standard cell library — Run synthesis by Design Compiler

Gate-level Netlist — Optimize result and generate gate-level Netlist

Gate-Sim — Check correctness after adding timing info.

3.

Floorplan — Decide each Components position

Placement — Place transistor in best position

Routing — Connect ports from different modules

Cadence innovus tool ...

Layout

Post-Sim — Check correctness after adding timing information

Verification — Check all function work and no violation — Design rule check (DRC), Layout versus schematics (LVS)

↓ Tape Out

Continuous Assignment → for wire assignments.

Procedural Assignment → for reg assignments.

Unsigned / Signed Mix Operation.

- If there's one unsigned operator, the operation would be regarded as unsigned.

```
wire [2:0] a, mod1;
assign mod1 = a % 7
             ↑   ↑   ↑
             u   u   s
                 (u)
```

```
wire [2:0] a;
wire signed [2:0] mod1;
assign mod1 = a % 7
              ↑   ↑   ↑
              s   u   s
                  (us)
```

```
wire signed [2:0] a;
wire [2:0] mod1;
assign mod1 = a % 7;
             ↑   ↑   ↑
             u   s   s
                 (s)
```

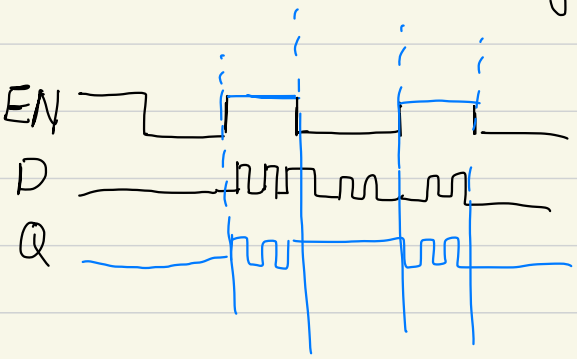
⚠ May be erroneous.

```
wire signed [2:0] a, mod1;
assign mod1 = a % 7
              ↑   ↑   ↑
              s   s   s
                  (s)
```

Lec 02

Sequential Circuits not only depends on the current input values, but also on preceding input values.

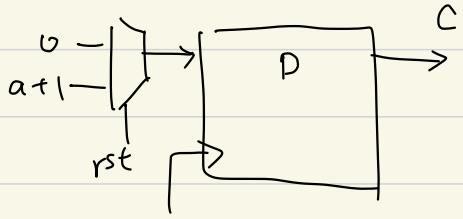
Latch: level sensitive, edge triggered



D Flip-Flop: Edge triggered

- Synchronous Reset.

```
always @(posedge clk) begin
  if(reset) c <= 0;
  else c <= a+1;
end
```

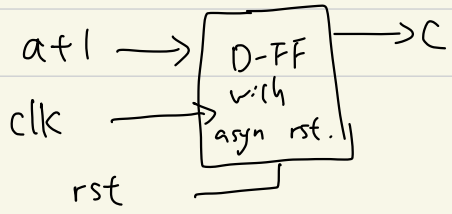


Pro:
Glitch filtering from reset comb. logic

Con:
- Can't be reset without clk.
- May need pulse stretcher
- Larger Area / Incress critical Path

- Asynchronous Reset.

```
always @(posedge clk or posedge rst) begin
  if(reset) c <= 0;
  else c <= a+1;
end.
```



Pro: Reset indep. clock signal
Reset is immediate
Less Area

Con: Noisy Reset line could cause unwanted reset
- Metastability.

For Loop in Verilog.

- Duplicate same function
- Very useful for doing reset and iterated operation.

```
reg [3:0] temp;
integer i;
always @(posedge clk) begin
  for (i=0; i<3; i=i+1) begin : for_name
    temp[i] <= 1'b0;
  end
end.
```

=

```
always @(posedge clk) begin
  temp[0] <= 1'b0;
  temp[1] <= 1'b0;
  temp[2] <= 1'b0;
end
```

Generate in Verilog:

```
module A();
endmodule

module B();
  genvar i;
  generate
    for (i=0; i<3; i=i+1) begin
      A uA(...);
    end
  endgenerate
endmodule.
```

⇒

```
module A();
endmodule

module A();
endmodule

module A();
endmodule
```

Mealy machine

- Output depend on the current state and input.
- If input changes, output also changes.

Pro: Less # of states needed
Con: More H.W required for implementation

Moore machine

- The outputs depend on the current state only.
- Inputs affect outputs but not immediately.

Pro: Safer, Output changes at clock edge
Con: More States required.

Lec 03

Verification : Demonstrate the functional correctness of a design.
Testing : Verify the design is manufacturing correctly.

Stages of Verification:
- Preliminary verification → Specification (ex. Output =0 after rst)
- Broad-Spectrum verification → Test Pattern (ex. Random test)
- Corner-case verification → Special test pattern (ex. Boundary).

Pattern:
Directed Testing → Check what you know
Random Testing → Find what you don't know.

Pattern:

Preliminary Verification (0 at rst) → Broad-Spectrum Verification → Corner-case Verification → Difficult to reach corner-case Verification

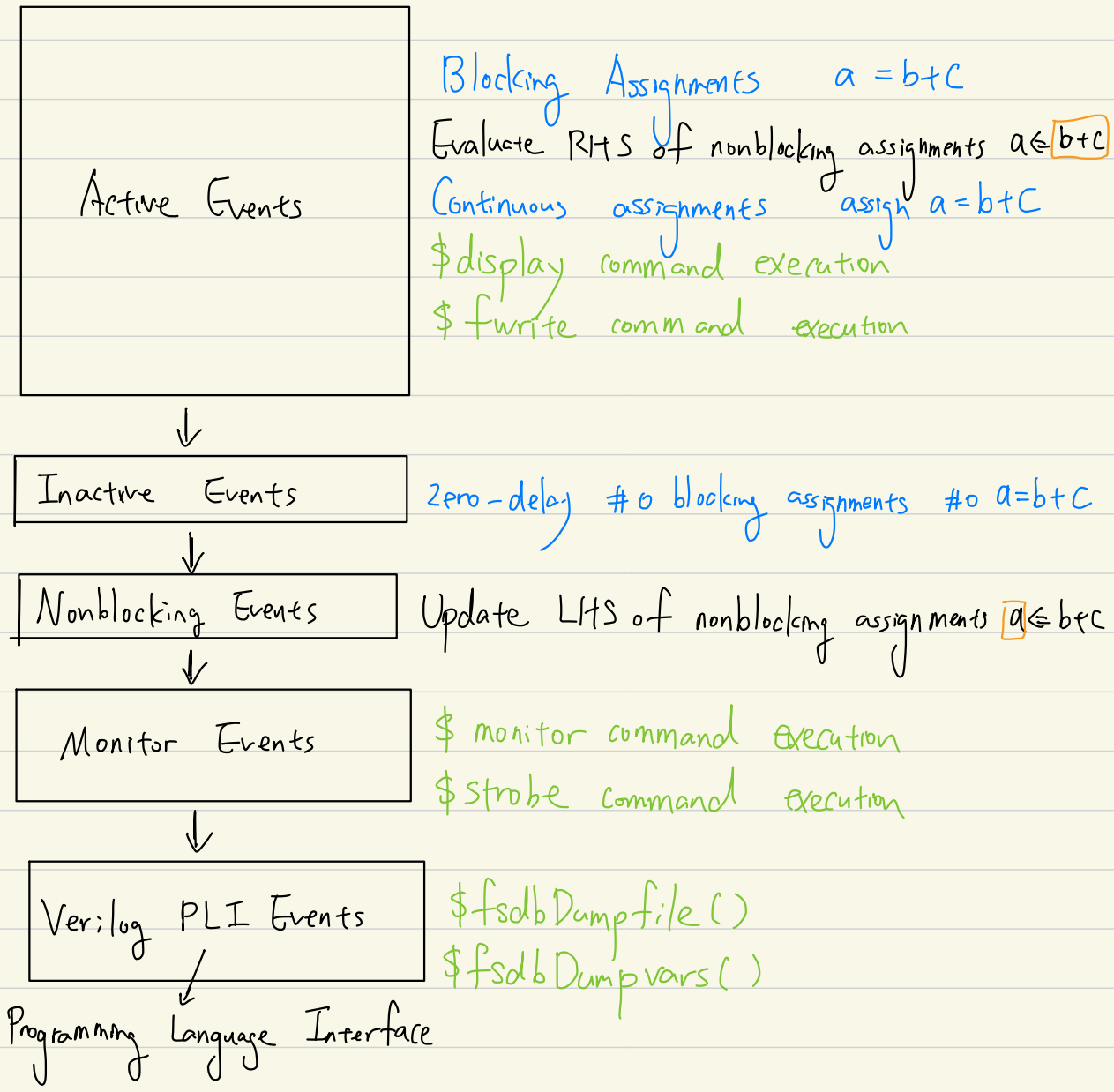
Preliminary Verification → Directed
Broad-Spectrum Verification → Broadly Constrained Randomization
Corner-case Verification → Narrowly Constrained Randomization
Difficult to reach corner-case Verification → Directed.

```
integer SEED = 123;
reg [3:0] number;
number = $random( ) % 'd7;
```

SEED (123)
number (4'b)
'd7 (7)

Stratified Event Queue of Verilog.

Lec03 P.24



Asynchronous reset:

- clock signal should be forced to 0 before reset signal is given.

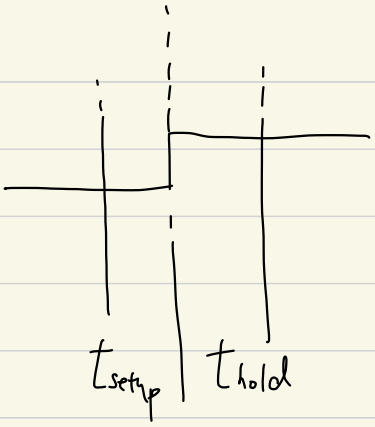
  force clk = 0 ----- release clk;

`timescale : specifies the unit of measurement for time and the degree of precision.

`timescale	Delay	Time Delay
10ns / 1ns	#5	50ns
10ns / 1ns	#5.738	57ns
10ns / 10ns	#5.5	60ns
10ns / 100ps	#5.738	57.4ns

Lec 04

- Setup time (t_{setup})
- Hold time (t_{hold})



Contamination delay
The minimum amount of time from an input changes until any output starts to change

- Clk-to-Q contamination delay (t_{ccq})
- Logic contamination delay (t_{cd})

Propagation delay
The maximum amount of time from input changes until all output reaches steady state.

- Clk-to-Q propagation delay (t_{pcq})
- Logic propagation delay (t_{pd})

Setup Time Criteron

Setup Time Check:

- data Required Time : $T_{cyde} + T_{skew} - t_{setup}$
- data arrival Time : $t_{pcq} + t_{pd}$
- Slack = data required time - data arrival time.

Hold Time Criterion
- data requireed time = $t_{skew} + t_{hold}$
- data arrival time = $t_{ccq} + t_{cd}$.

Slack = data arrival time - data required time

Pipelining doesn't help latency of single task, but throughput of entire.
Pipeline rate are limited by the slowest stage.

IP (Intellectual Property)

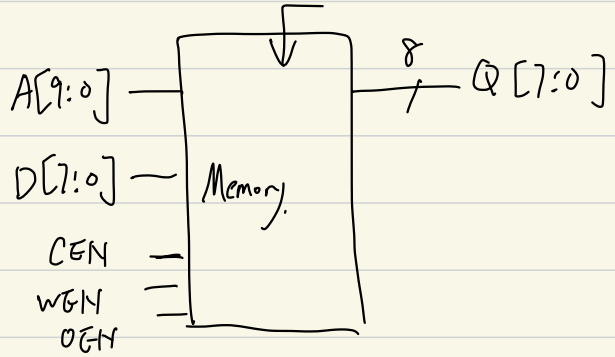
- Hard IP : GDSII Format, high performance but technology dependent.
- Firm IP : Netlist resource, less used.
- Soft IP : RTL design, requires verification.

Lec 05

SRAM

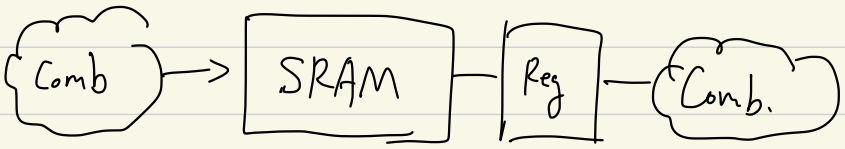
- Read / Write Data only
- Memory has less area than registers
- Memory is slower than registers.
- Only one address

SRAM.



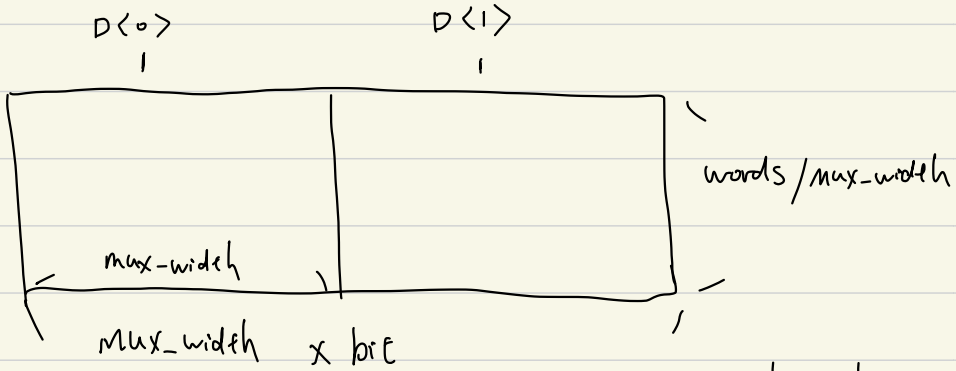
CEN: Chip enable negative
OEN: Output enable negative
WEN: Write enable negative

To avoid critical path causing timing violations:
Add registers after the hard macro.



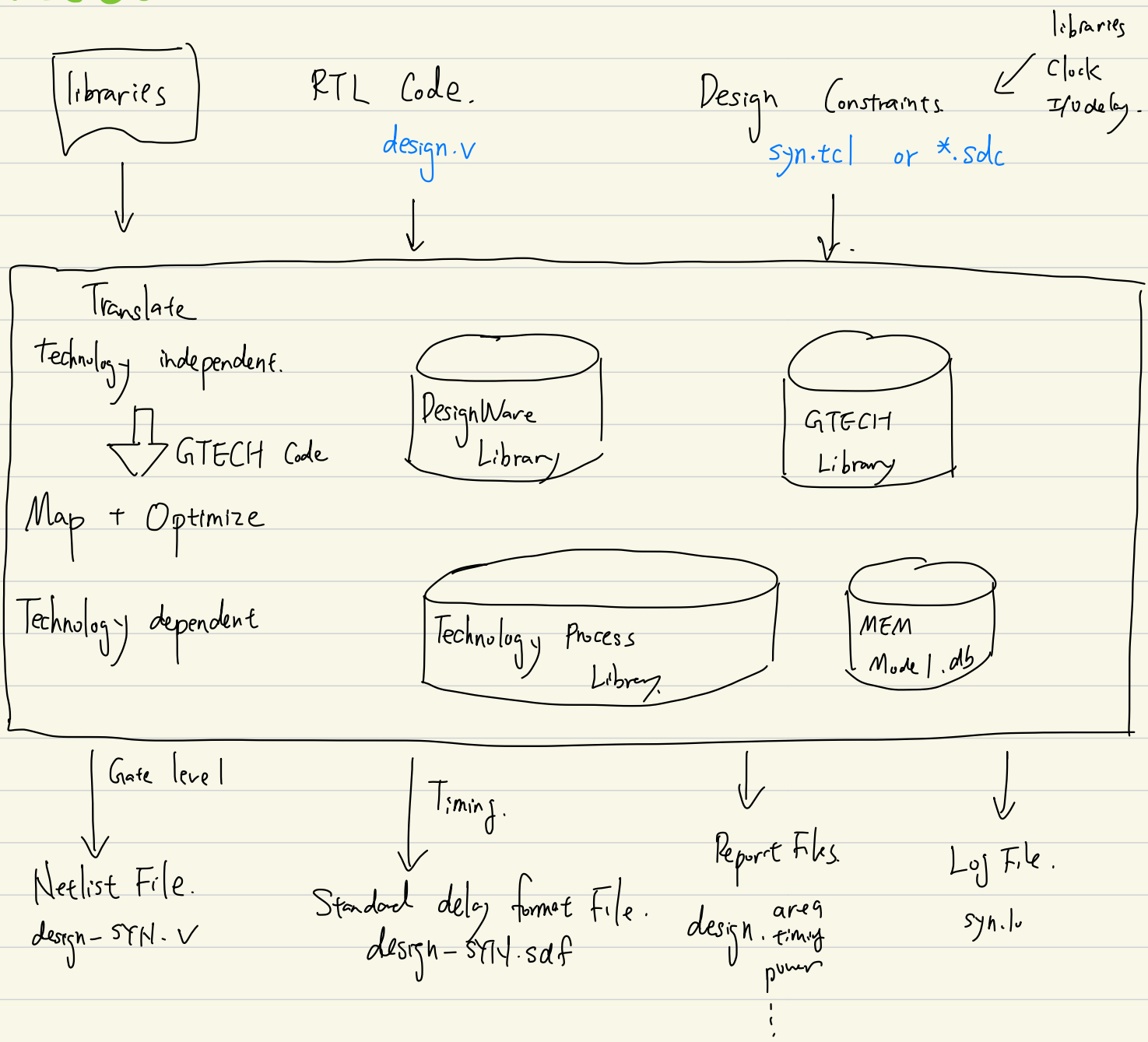
Memory Compiler.

64 words, 2bit, mux-wid = 16



$\Rightarrow$ mux_width x bit = words ÷ mux_width

Lec 06


libraries
RTL Code.
design.v
Design Constraints.
syn.tcl or *.sdc
libraries
Clock
I/O delay.
Translate
Technology independent.
GTECH Code
Map + Optimize
Technology dependent
DesignWare Library
GTECH Library
Technology Process Library
MEM Model.db
Gate level
Netlist File.
design-SYN.V
Timing.
Standard delay format File.
design-SYN.sdf
Report Files
design.
area
timing
power
Log File.
syn.lo

Develop HDL files.

↓

Specify Libraries.

.lib → .db

① Synthetic Lib : additional DW lib for optimization
② Link lib : list of libs Design Compiler could use to resolve design ref.
③ Target lib : Includes timing.

link_library synthetic_library target_library

↓

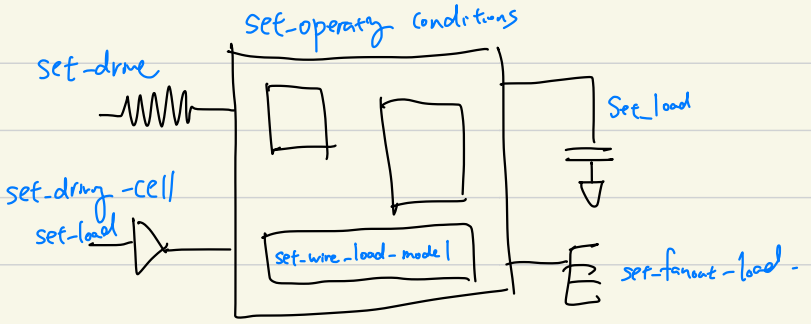
Read Design

read_file, read_verilog
analyze -f , elaborate

↓

Develop Design Environment

Define the environment in which the design is expected to operate in by specifying operating conditions, wire load models, system interface,



Before APR, wire load models can be used to estimate cap. resis. and area overhead, top, enclosed, segmented.

↓

Set Design Constraints

Design Rule Constraints
- reflects technological-specific constraints. must hold

- set_ideal_network {clk, rst_n}

Design optimization Constraints.
- user defines speed (timing) and area optimization goals for the design compiler.

- create_clock