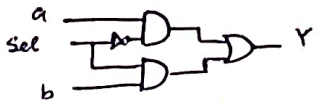
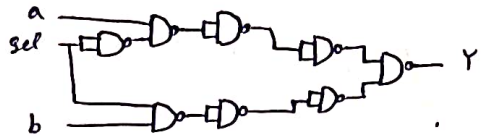
15-10-2024

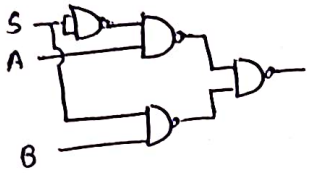
HDLSS

-Kedar Kulkarni

```
always @*
begin
  if (cout)
  begin
    a = 1'b1;
  end
  else
  begin
    a = 1'b0;
  end
end
```







Sel	Y (out)
0	a
1	b

◦ HDL Design flow

→ Design description

i/p - MRD

o/p - PDD, HLA, MA (µA)

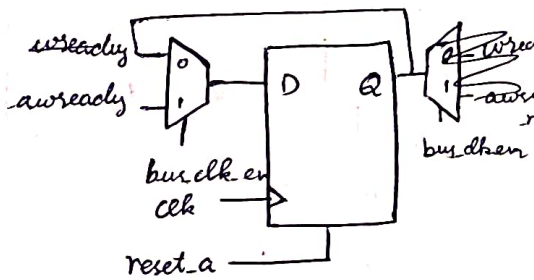
→ Verification

→ Lint → latch
→ overflow, rapround
→ CDC

→ Synthesis → (reuse of gates)
→ mapping (based on FPGA/Synt block)

```
always @(posedge clk)
begin
  if (rest_a == 1'b1)
  begin
    awready_reg <= 1'b0;
  end
  else
  begin
    if (bus_clk_en == 1'b1)
    begin
      awready_reg <= awready;
    end
  end
end
```

```
always @(posedge clk)
begin
  if (rest_a == 1'b1)
  begin
    wready_reg <= 1'b0;
  end
  else
  begin
    if (bus_clk_en = 1'b0)
    begin
      wready_reg <= wready;
    end
  end
end
```



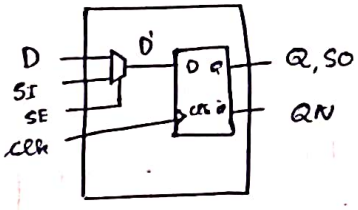
- Gate level simulation → ton, toff, means nase → very slow → to catch x propagation
- Implementation → Mapping → Placement → Routing

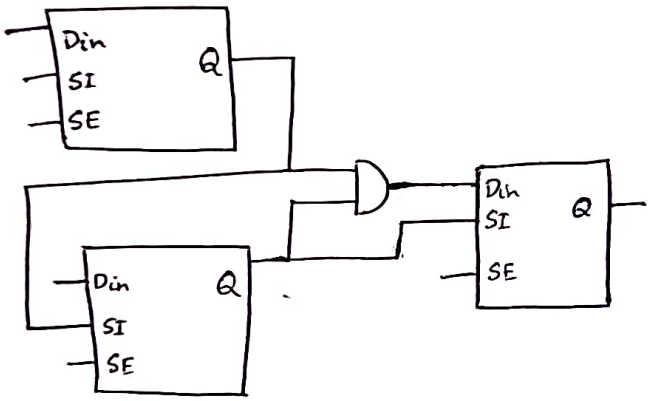
Synthesis Flow (ASIC)

→ Registered o/p → fixed delay at o/p, not change.

→ DFT → Scan flops → (to check gates after fab)
→ Stuck at faise model →
→ Scan chain to check flip-flop chain

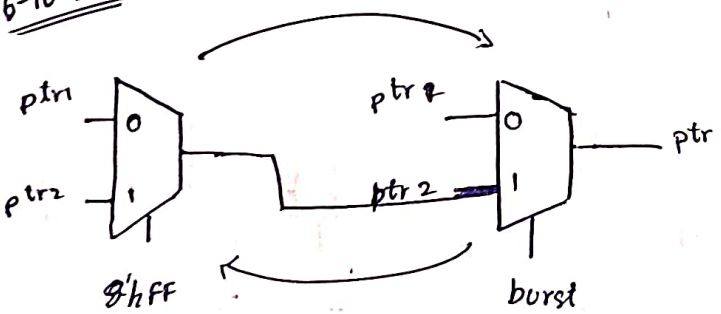
ATPG → Tool used to generate test logic





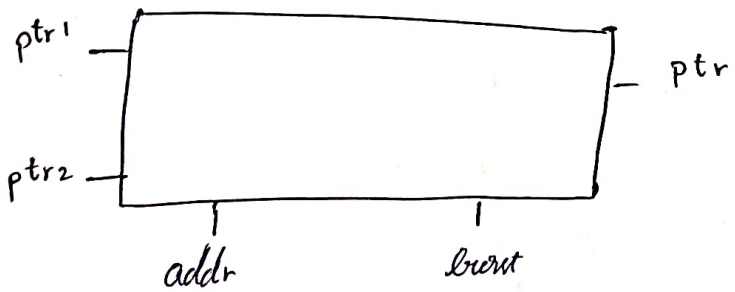
Pattern at i/p
→ 00110

16-10-2024



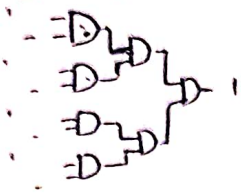
nested if-else

```
if (adelr)
  if (burst)
    ptr = ptr2
  else
    ptr = ptr1
else
  ptr = ptr1
```



addr == 8FF

8bit x2 +1 => 15 and gate



NAND Gates.

Clock tree synthesis

-> to make clock available to all ~~FF~~ Sequential element at same time.

Y, -X, H, tree

GDSII for fabrication of ASIC

Technology library

-> standard cell library => gates, mux, t_{delay}.

LVT - less switching power.
- more leakage power
- no change in area considerable

Primitive
- Flop, mux, Low

Macro
- RAM, FIFO, DSP

IP cores
- USB, HDMI, ethernet, PCIE

Event based simulation & Cycle based simulation.

on change of event.
-> to detect glitches
-> time consuming

Event scheduling in simulation

Active ⟶ Non-blocking Queue → Postponed Queue

Time scale

`timescale 1ns/1ps

<time_unit> / <time_precision>

Modeling concepts

Behavioral modeling -> equation, functionality.

Dataflow modeling -> gates, operators

Structural modeling -> predefined block

ASIC synthesis

Memory Replacement ⟶ `reg [n:0] mem[0:x]; // fake ram`

⇒ Replace with real memory provided by vendor
- → timing, routing
- ⇒ single port, dual port (multiplexed)
- → B_RAM hardware avail in FPGA

Block D-RAM gen by code.

Doc. management.

Organizing.

Version control.

Design Unit

{{}} - replication operator

=== - to compare x, z also

parameter, local parm (cannot be over written from top level model)

JK → Toggle

clk	Q_0	Q_1	Q_2	Q_3
init	0	0	0	0
↓	1	0	0	0
↓	0	1	0	0
↓	ϕ	ϕ	ϕ	0
↓	0	0	1	ϕ
↓	ϕ	0	1	1
↓	0	1	0	1
↓	0	0	0	0
↓	0	0	0	
↓	1	0	0	

Q_1	Q_3	clr
0	0	1
0	1	1
1	0	1
1	1	0

	Q_3	Q_2	Q_1	Q_0
0-	0	0	0	0
1-	0	0	0	1
2-	0	0	1	0
3-	0	0	1	1
4-	0	1	0	0
5-	0	1	0	1
6-	0	1	1	0
7-	0	1	1	1
8-	1	0	0	0
9-	1	0	0	1
10-	1	0	1	0

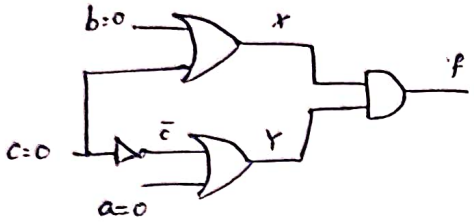
Ripple counter

→ any async clk design
→ delay gets propagated
→ prone to glitch, uncertainty of o/p, clk

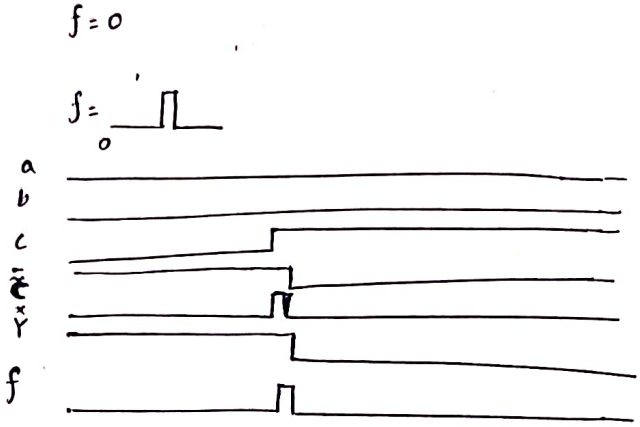
∴ constraint for generated clocks

Hazards

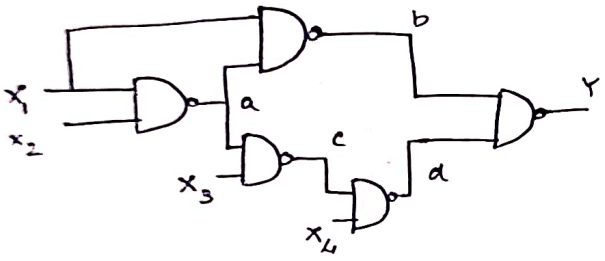
Static 0

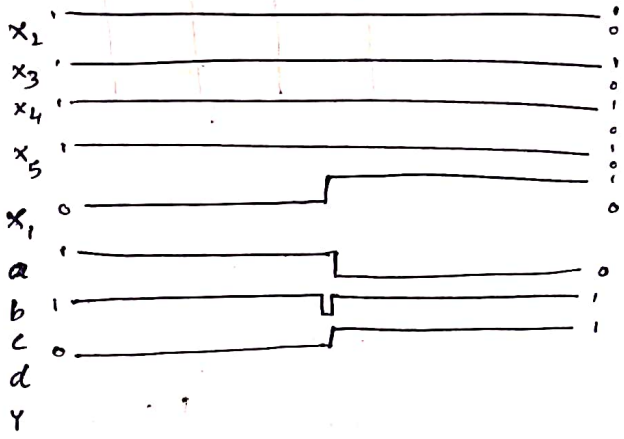


for change in c from 0 → 1



A B	Y = AB̄
0 0	1
0 1	0
1 0	0
1 1	0





x_2, x_3, x_4 - 1

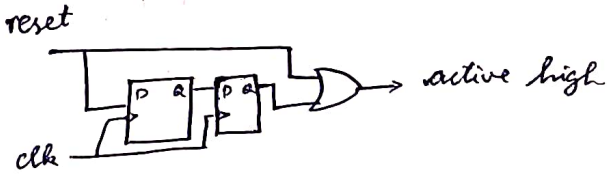
x_1 0 → 1

Clk & reset

Trem, Trec
↳ removal ↳ recovery
↓ de-assertion of clk signal.
time reset must be stable before clk edge

} reset assertion synchronous / async
de-assertion ~~only~~ synchronous.

use of synchronizer for reset

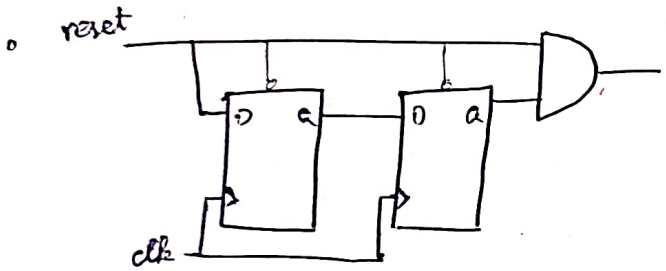


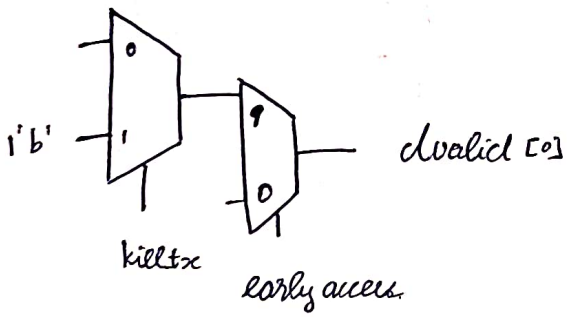
17-10-2024

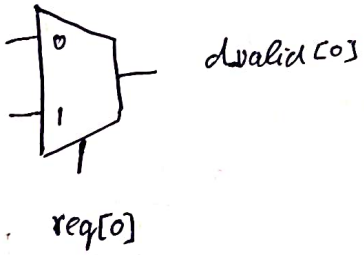
```
always @(posedge clk or negedge rst)
begin
    if (!rst)
      rsta <= 2'b00;
    else
    begin
      rst_a[0] <= 1'b1;
      rst_a[1] <= rsta[0]
    end
end

assign  global rst and rst_a;
```

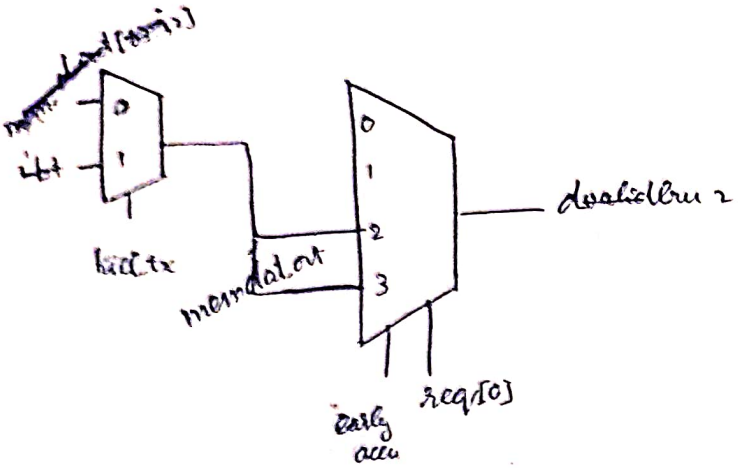
→ change in gate

→ reset value = 0 of each flop.









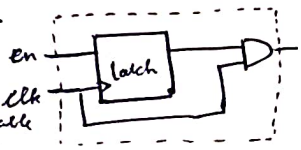
×

Clk & reset

→ Single & multiple clk domain → avoid multiple contact points. (Cint throws warning msg) multiple sync required.

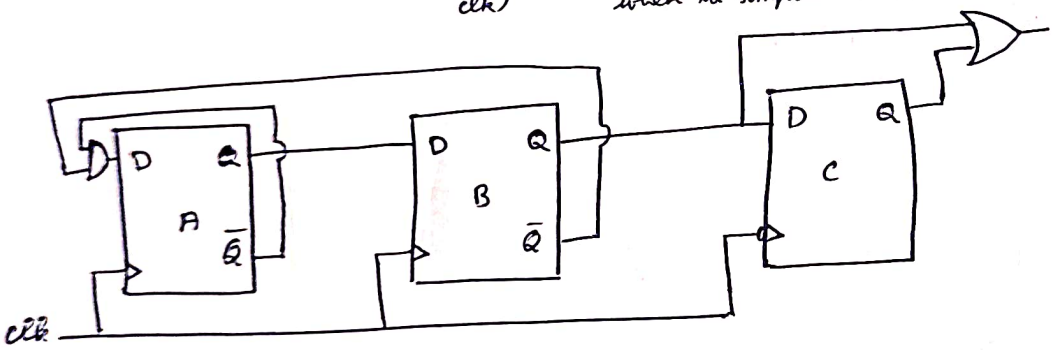
→ Clk gating (off/on save power) → data line level sync not need FIFO (VLSIVERIFY.com) as data not certain at a time

→ Internal generated clk → PLL, bus clk (generate clk) (to know when to sample.



→ make sure finally remain IDLE, please disable data

→ mixed clk → DDR



clk	A	B	C	o/p
	0	0	0	0
↓	1	0	0	0
↓	0	1	0	1
↓	0	0	1	1
↓	0	0	0	0
↑	0	0	0	1
↓	0	0	1	1
↑	1	0	0	0

divide by 3

Synthesis - Simulation mismatch

Translate on, OFF ⇒ directives

if-else nested → avoid redundant logic case <priority encoder>

case (1'b1)
a_in: ...
b_in: ...
endcase

case

Blocking & non-blocking assignment

Function automatic → saves memory space for repeated call

Arithmatic operation → adder at the end Y = x ? c : d + x ? a : b;

→ A+B+C+D vs (A+B) + (C+D)

18-10-2024

Constant propagation

```
assign a=1,b=1;
always @*
begin
  op = a+b
end
```

constant 2 assigned to 'op'

```
assign sel=1;
if (sel==0)
  x=y
else
  x=z;
```

No mux infered

```
always @*
begin
  case (sel)
    2'b00 = a
    2'b01 = b
    2'b10 = c
    2'b11 = d
  endcase
```

sel[1] always-0

2'b00 = a, 2'b01 = b → 2:1 mux

2'b10 = c, 2'b11 = d → x

Register duplication

IOW/RAZ → Return always zero → on implementation need not be generated.

↳ Ignore on right

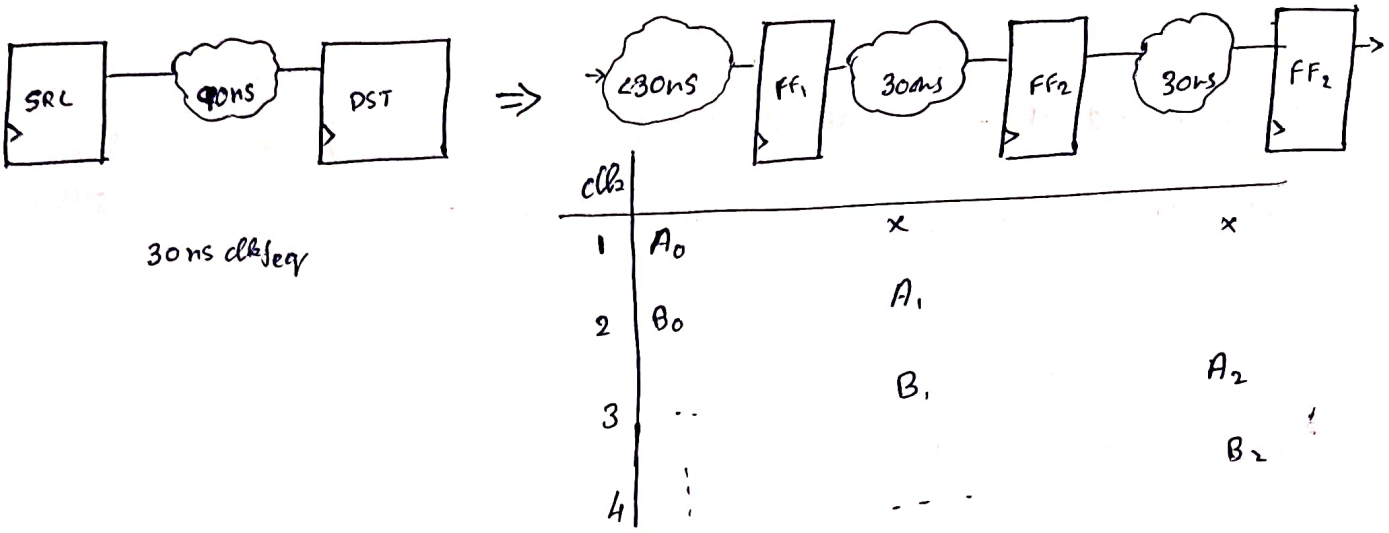
op can drive more other logic

→ replication, optimization setting to auto.

Fan out constrain can be provided.

Pipelining

→ done at design level not at synthesis
↓2
architectural.

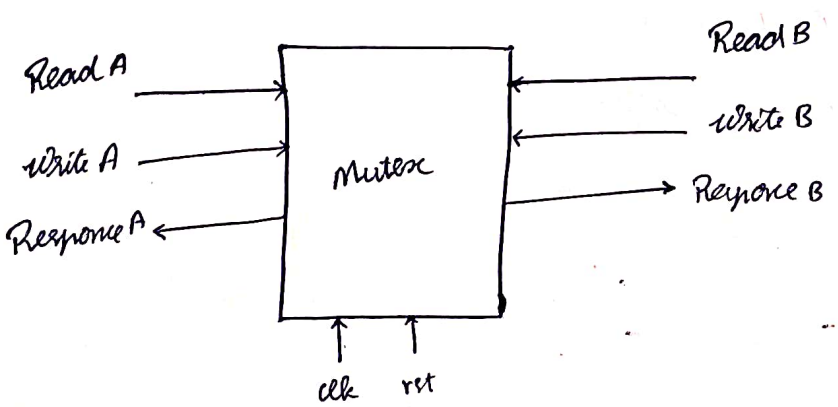


30ns clk seq

MUTEX

1. Read is performed by a entity (A or B) to reserve resource (mutex)
2. If a write is performed by some entity that has reserved the resource 'Succes' responce is given by mutex module in the next cycle after reciving clk.
3. If a write is performed by an entity but the mutex is reserved for other entity the response to write will be fail.
4. Which ever entity perform the last read will have the mutex

Read_A → Mutex reserved for A
Read_B → Mutex reserved for B



Low power design

Activity, Frequency, capacitive load ....

- Dynamic power → increase linearly
- Static power → increases exponentially

] trend in shrinking (new tech libraries)

Design technique

Clock gating, enable, coding style

FSM encoding

FSM state → Binary, Gray, One-hot

Static power dissipation → sub threshold leakage ↑
→ Gate leakage ↑
→ Gate induced drain leakage ↑

Dynamic power ↳ Square of voltage ↑

Multi Vdd → Voltage domain.
, power up power down sequence

Level shifter → Low → high (have significant delay)
→ placed on reciver end.

Static voltage scaling -

Multi level voltage scaling -

Dynamic voltage frequency scaling -

Adaptive voltage frequency scaling -

Multi Vt → Voltage technology.

Power gating - more low power than clk gating.

- power up / down sequence to be followed (to avoid uncertainities)
- isolation cells. (o/p fixed, x must not propagate) (avoid crowbar current)

Exam section.

- retension registers (preserve data during power down)

FIFO, memory => not possible

Follow power gating sequence

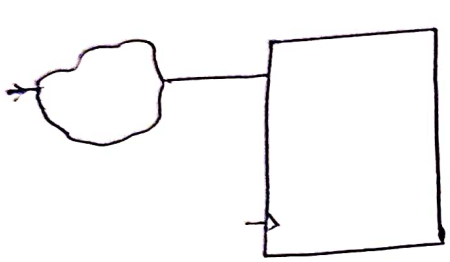
Memory → light sleep
→ deep sleep
→ shutdown.

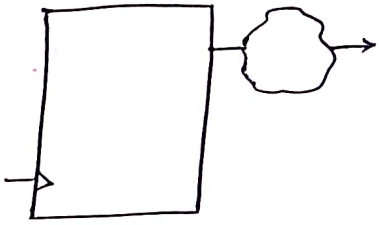
Coding guidelines.

- module name & file name as same.
- inputs in seperate line.
- dub global input // as comment.
- naming variables as required FF1 to FF2
- don't change name in protocol SDA, SDA
- and functionality to prefix tx, rx, counter.
- use `define for fixed value memwidth.
- use name for FSM
- All caps variable name local param,
- _clk at end, _state, _state_next, _state_r [not _reg]*
- _a for async inputs (reset a)
- _n for active low

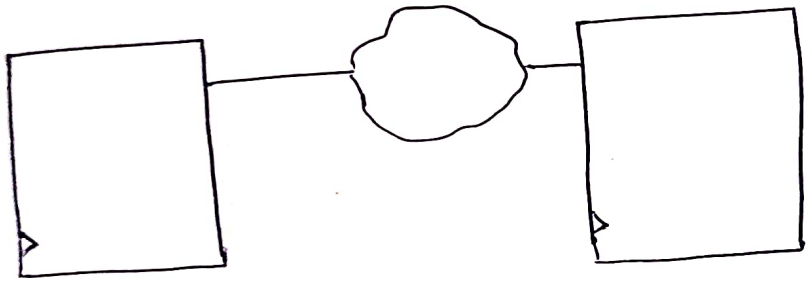
- _cg (clk gate) _sel (select line).
- : always_PROC (name as visible in VCS)
  _reg_PROC (for sequential reg)
- _PASS, _READY, _HOLD, _STALL
- _lh (one hot)
- assign with multiple condition, can be written in multiple line.

STA

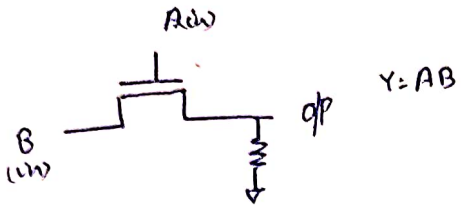


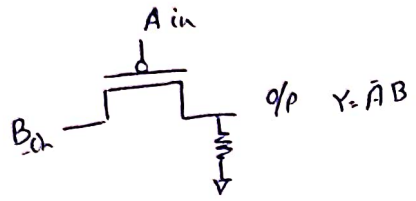


↳ rare
- latched, combi
- low latency

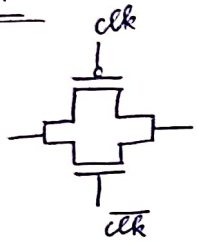


19-10-2024

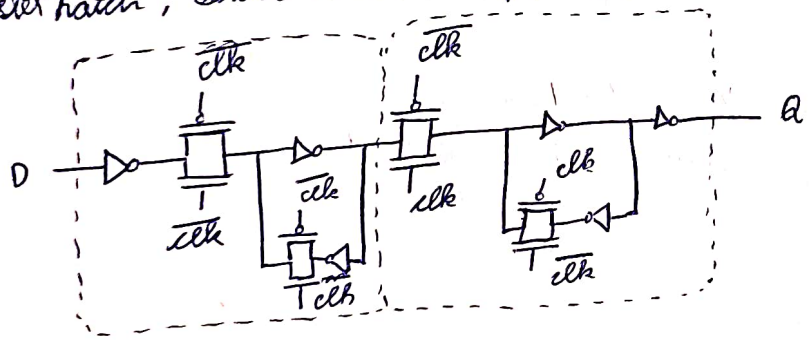


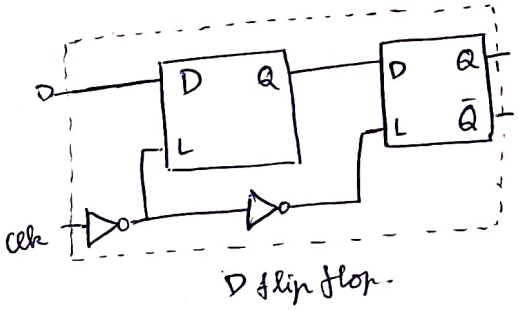


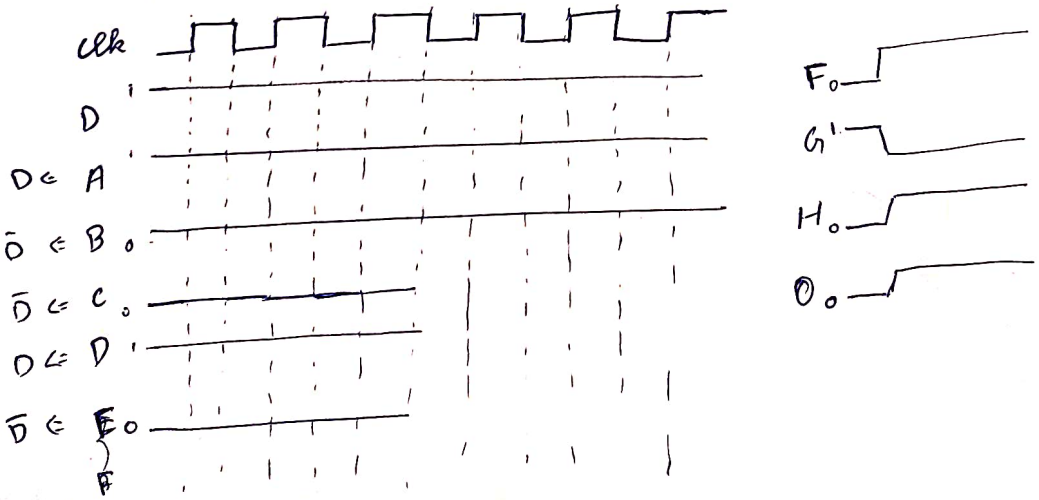
Pass transistor



Master latch, Slave latch => D flip flop using pass transistor







STA

t_{setup} , t_{hold} , $t_{clock\ q}$ delay

t_{setup} violation → reduce logic delay

t_{hold} violation → increase logic delay

→ $t_{setup} = 9ns$ $t_{clock} = 12ns$ $t_{transport} = 25ns$ $t_{logic} = 54ns$

max delay = $t_{setup} + t_{clock} + t_{transport} + t_{logic}$

max-delay = 100 ns

$f = \frac{1}{100} \times 10^{-9}$

$= 10MHz$

→ $t_{setup} = 5ns$ $t_{clock} = 10ns$ $t_{transport} = 10ns$ $t_{logic} = 13$ 0-1
$t_{logic} = 23$ 1-0

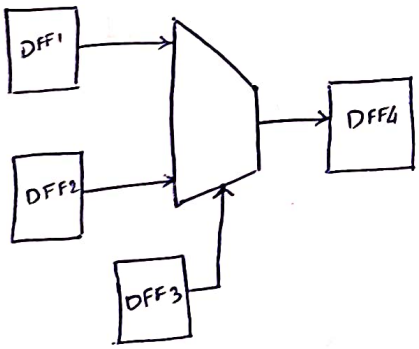
max T = 25 + 23

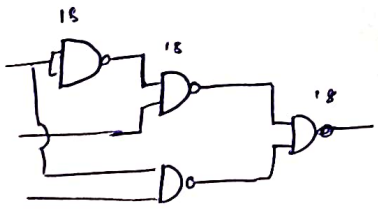
= 50ns ⇒ f = 20MHz

→ $t_{setup} = 10ns$ $t_{transport} = 0$ $t_{clock_1} = 8ns$ $t_{clock_2} = 10$ $t_{clock_3} = 12ns$

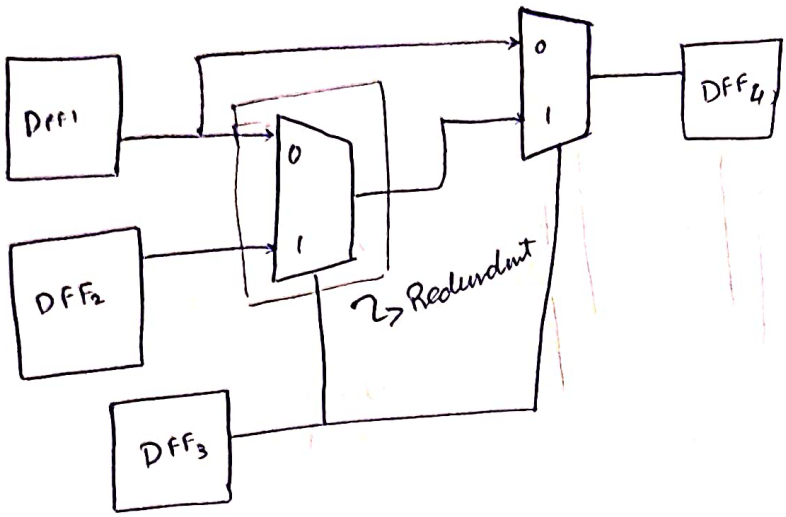
t_{logic} (NAND gate) 95 ns - 0 → 1

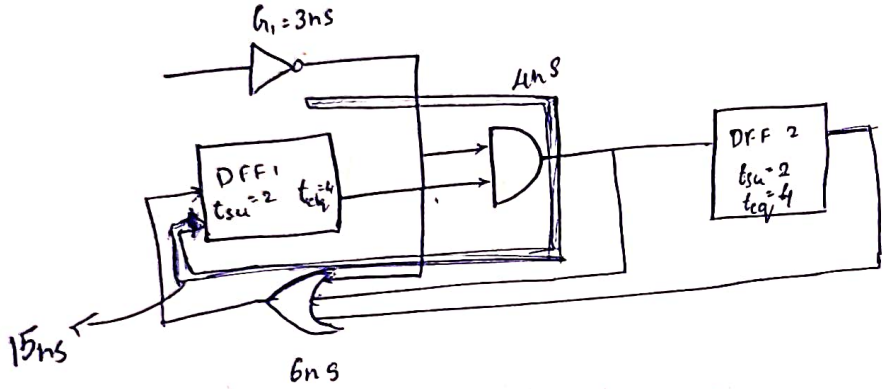
t_{logic} (NAND gate) 18 ns - 1 → 0





⇒ 54ns





$3+2+4+4+ \checkmark \Rightarrow 13\,ns$

$\Rightarrow$ STA braked after a flop → next delay calculated after

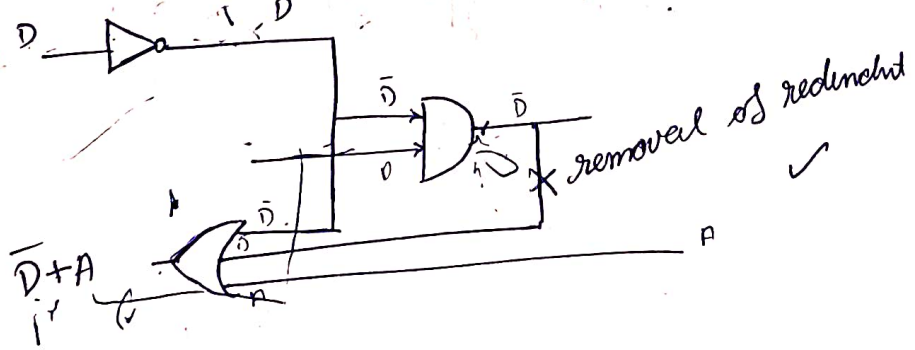
combo → 9 ns

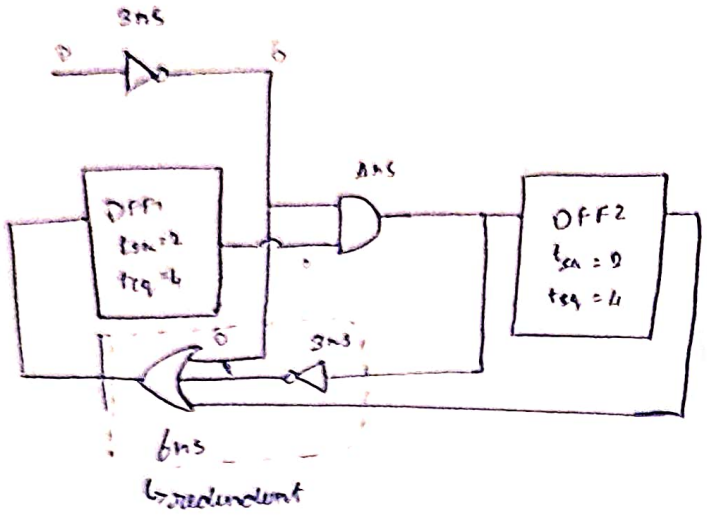
→ 15 ns

Reg - Reg → 10ns

12 ns [tctoq → next tsetup]

src destid...





$D + \bar{D} = 1$

$D + 1 = 1$

Combo $\Rightarrow 3+4+3+6 \Rightarrow 16\,ns$

Reg - Reg $\Rightarrow 4+4+2 \Rightarrow 10\,ns$

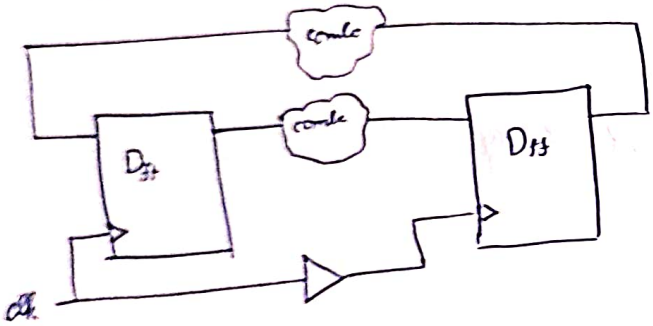
$FF_1 - FF_2$

$FF_1 - FF_1 \rightarrow 6+4+6+9 \Rightarrow 25\,ns$

$FF_2 - FF_1 = 4+3+6+2 \Rightarrow 15\,ns$

Positive skew ✓ Negative skew ×

positive skew adjustment, propagate to next flop.



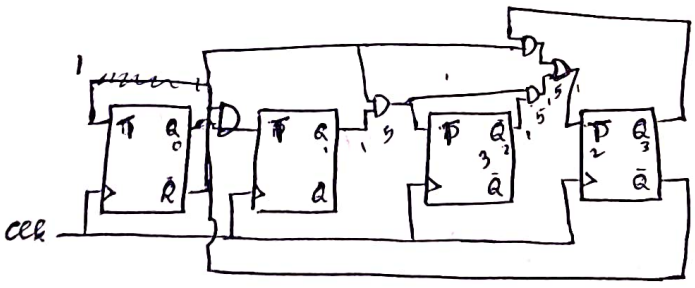
21-10-2024

Decade counter

$t_{su} = 2ns$ $t_{cq} = 3ns$

gate delay = 5 nS

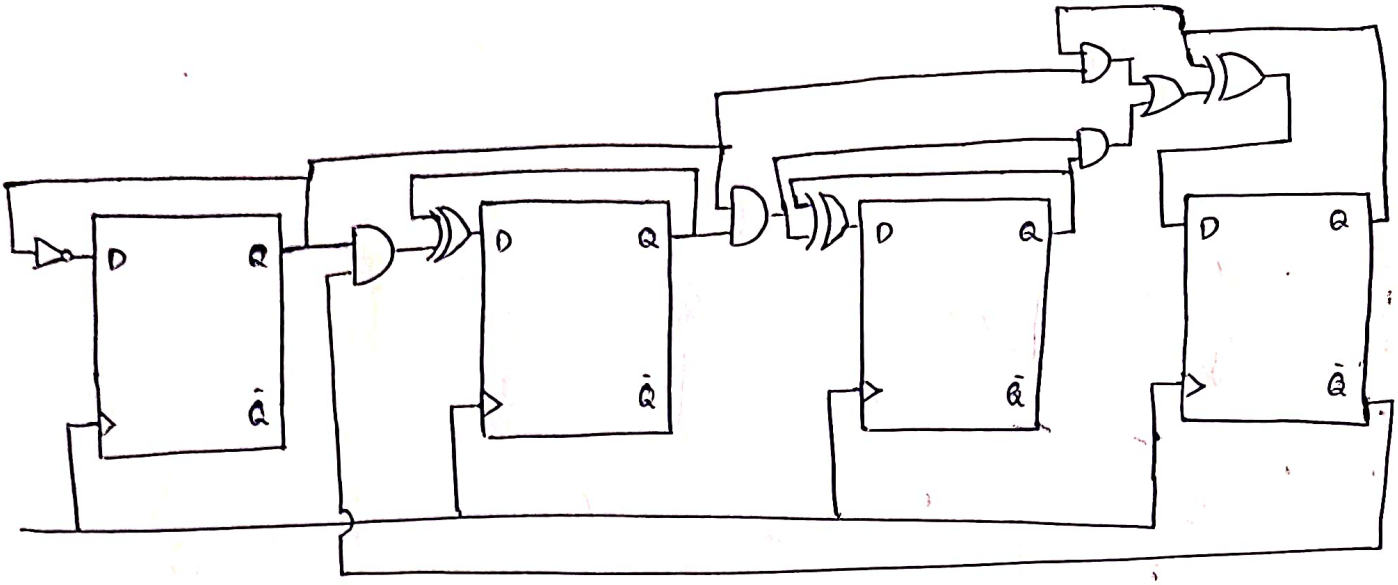
transport delay = 1nS

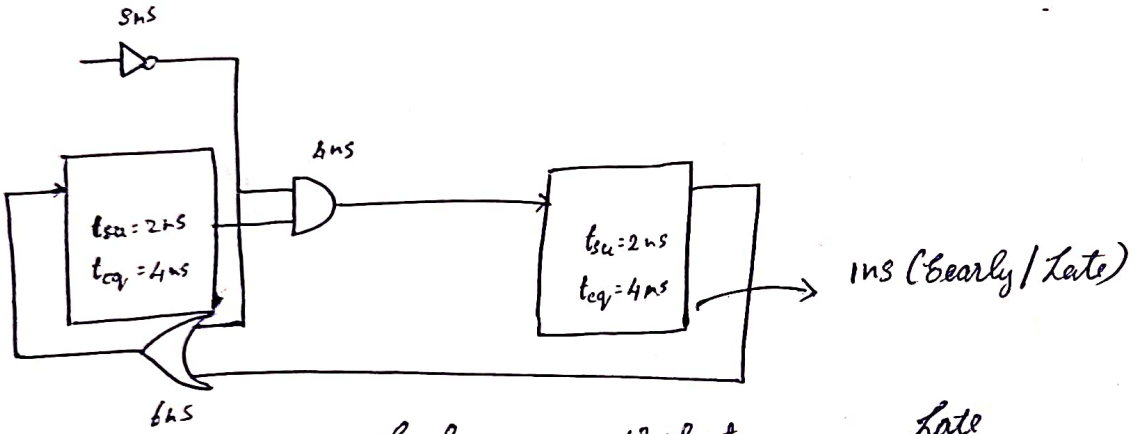


$= 1+5+1+5+1+2$

$= 18nS + 7$

$= 25nS$



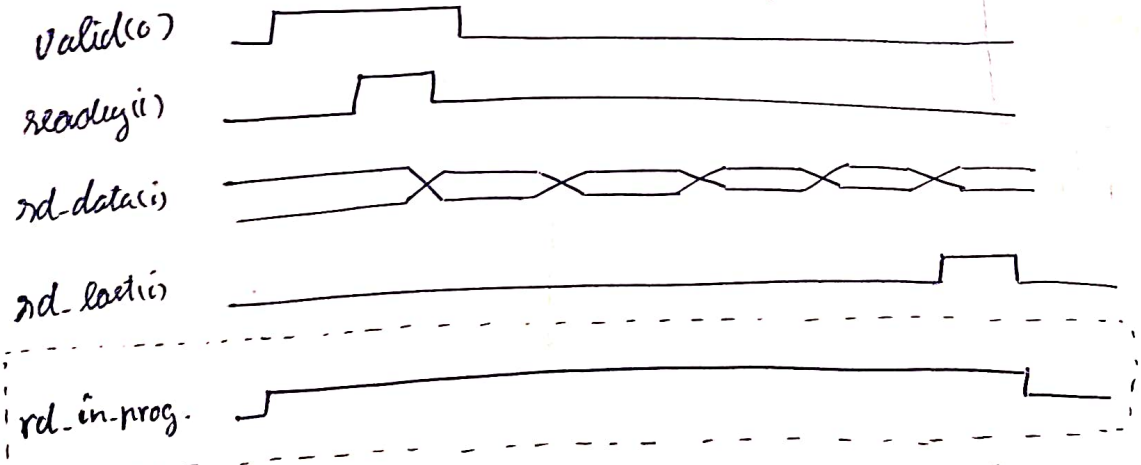


Early
= 11 ns

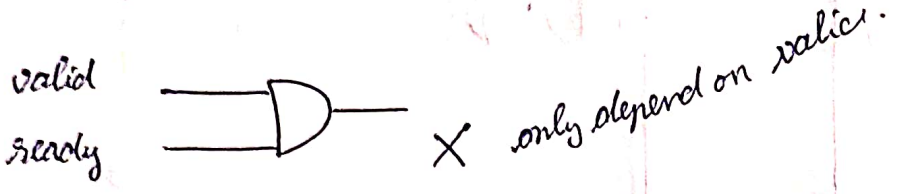
Without
= 12ns

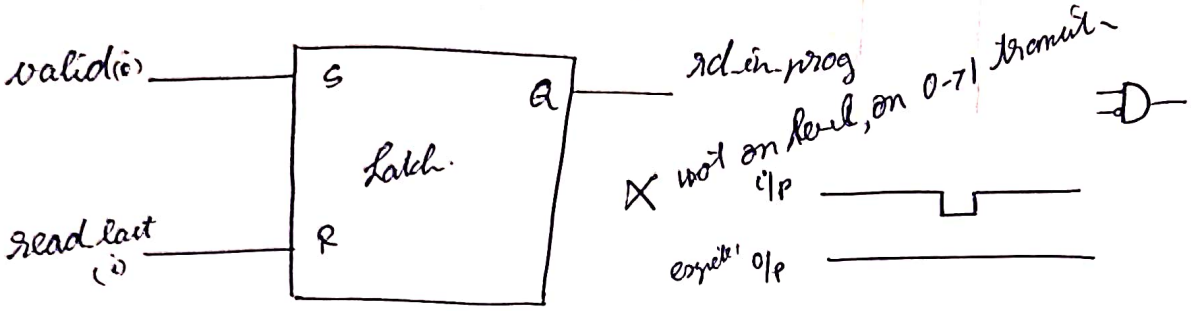
Late
= 13ns

22-10-2024



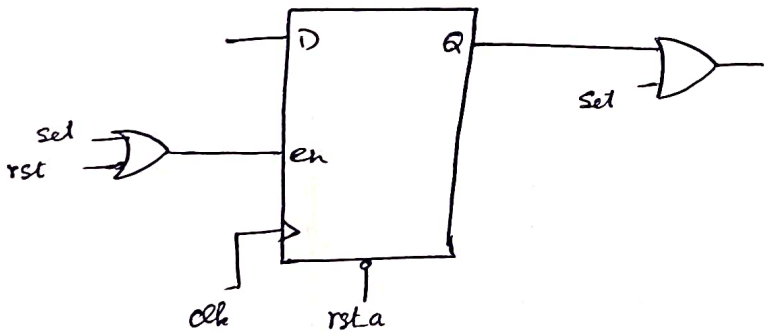
Implement without FSM





- identify 0→1 or 1→0 transition → exor to detect
  → inequality detector
  →

- a flop to get o/p



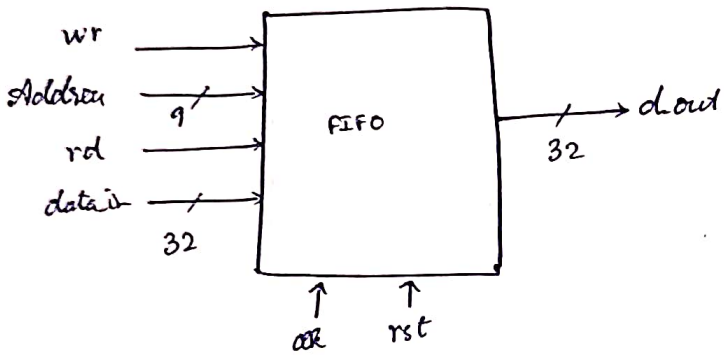
case =

1) Full case
2) Parallel case } Synthesis engineers.

↳ bizz, bzz1 => complicated logic

False path →
- identification
- & elimination

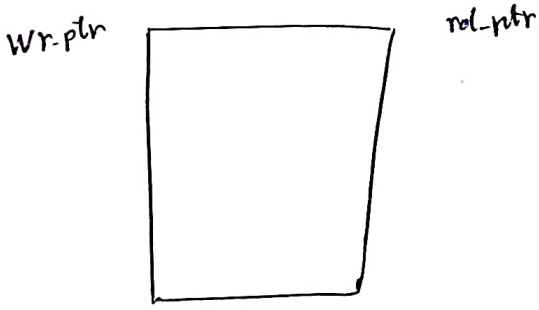
23-10-2024



512 x 32 => total

Read can happen only when entries are ≥ 16

SISO → 16



(wr_ptr == rd_ptr) → FIFO empty

0
1
2
3
↓ write

0
1
2
3
↓ read

full & empty programable

- wr_ptr

- diff_ptr → give no of entries present at FIFO at given time

- rd_ptr

---

diff_ptr ≥ 16

'b 10000 => 16

-------10000 => ≥16

OR

Multi-cycle path

↳ different t_combo delay

t_combo 10ns, 8ns

2-clk to complete some

→ use of enable

→ use of clock gating.

Constrains

Timing constrains

create_clock

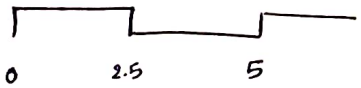
[source_object]

[-period period_value]

[-waveform edge_list]

[-edge name clock_name]

create_clock -period 5 [get_ports clk1]

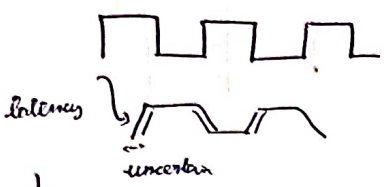


create_clock period 5 -waveform {1 2} [get_ports clk2]

↓
waveform

→ latency, uncertainty, transition time

1. Source
2. Network



↓

set_clock_latency [-rise] [-fall] [-source] [-early] [-late] [-clock clock_list]

set_clock_uncertainty 1 [get_clocks clk]

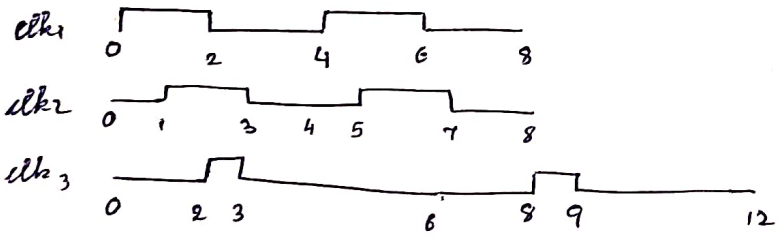
set_clock_uncertainty 2 (from set_clock

set_clock_transition transitiontime [-rise] [-fall] [-min] [max]

create_clock_period 4 (name) clk1 -waveform {0 1 2}

create_clock_period 4 clk2 -waveform {1 3}

create_clock_period 6 clk3 -waveform {2 3}



Asynchronous clk

→ share a common source

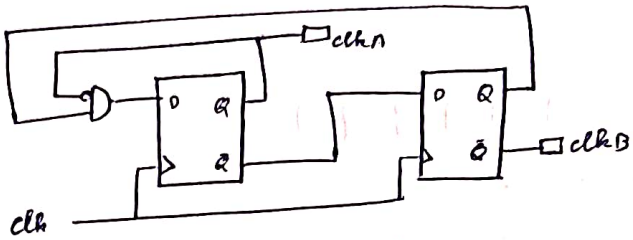
→ have a fixed phase relationship

set_clock_gating_check [-setup margin] [-hold margin] [-rise] [-fall]
[-high] [low] [object list]

create_generated clock -

÷ by 2 ÷ by 3

00
01
10
00
01
10



get_port, get_net, get_cell, get_pins

set_false_path -from register1 -to register2

set multicyclepath -setup 2 -from dff1 to dff2 -

24-10-2024

Vivado → Project settings.

Synthesis → settings

- gated_clock_conversion

:

Implementation → settings.

False path types

1. different clk
2. architecture
3. i/p based static

→ through → can be used to bypass checks but include other.

set_false_path -to [get_clocks clk]

- rise_from ⇒ 0→1
- fall_to ⇒ 1→0

→ Setup hold →

- setup 2 → setup check cycle
- hold 0 → hold = setup - 1 (count in reverse from setup set)

i/o latency → set_input_delay delay_value -clock clock_ref [-max] [-min] [-clock_fall] [-rise] [-fall] input port
output --- --- --- --- --- input port

Async - clock group

set clock_group -asynchronous -group {clkA, clkA_div1by2} -group {clkB, clkB_by3}

Virtual clock

to check async interrupts,

→ Set_input_delay 1 -clock [get_clock axi_clk] [get_port axi_*]
[get_ports axi_wdata]
2
↳ last will over ride previous

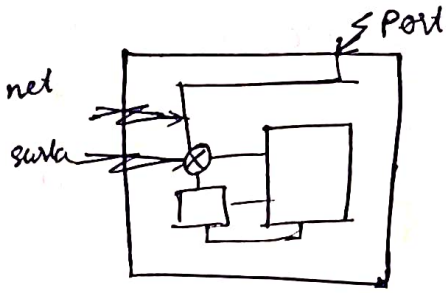
set_max_delay

set_min_delay.

UPF -> Unified power format

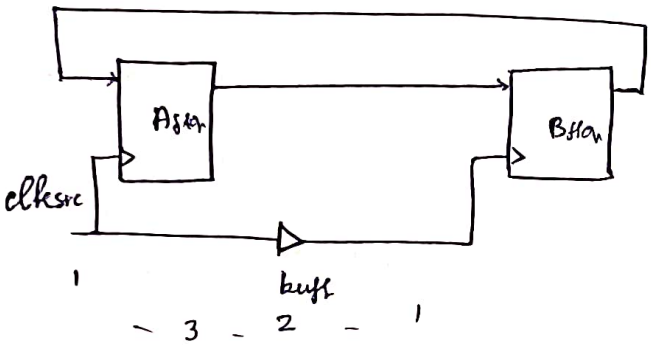
-> Limitation in Verilog

-> Functional modeling of power gating
isolation
retention



assertion & coverage

- supply net - power domain - supply port - power switch - level shifter - isolation cell - retention cell



clksrc to FlopA t_{min}=1ns
t_{max}=3ns

clk src to buffin t_{min} = 3ns
t_{max} = 5ns

buff delay t_{min} = 2ns
t_{max} = 4ns

buff out to flop B = t_{min} = 1ns
= 3ns

~~5~~ ns - 2 => 3 ns

25-1024 Fish tail -> synopsys.

- Assertion based ... .pdf
- HLA ... report document