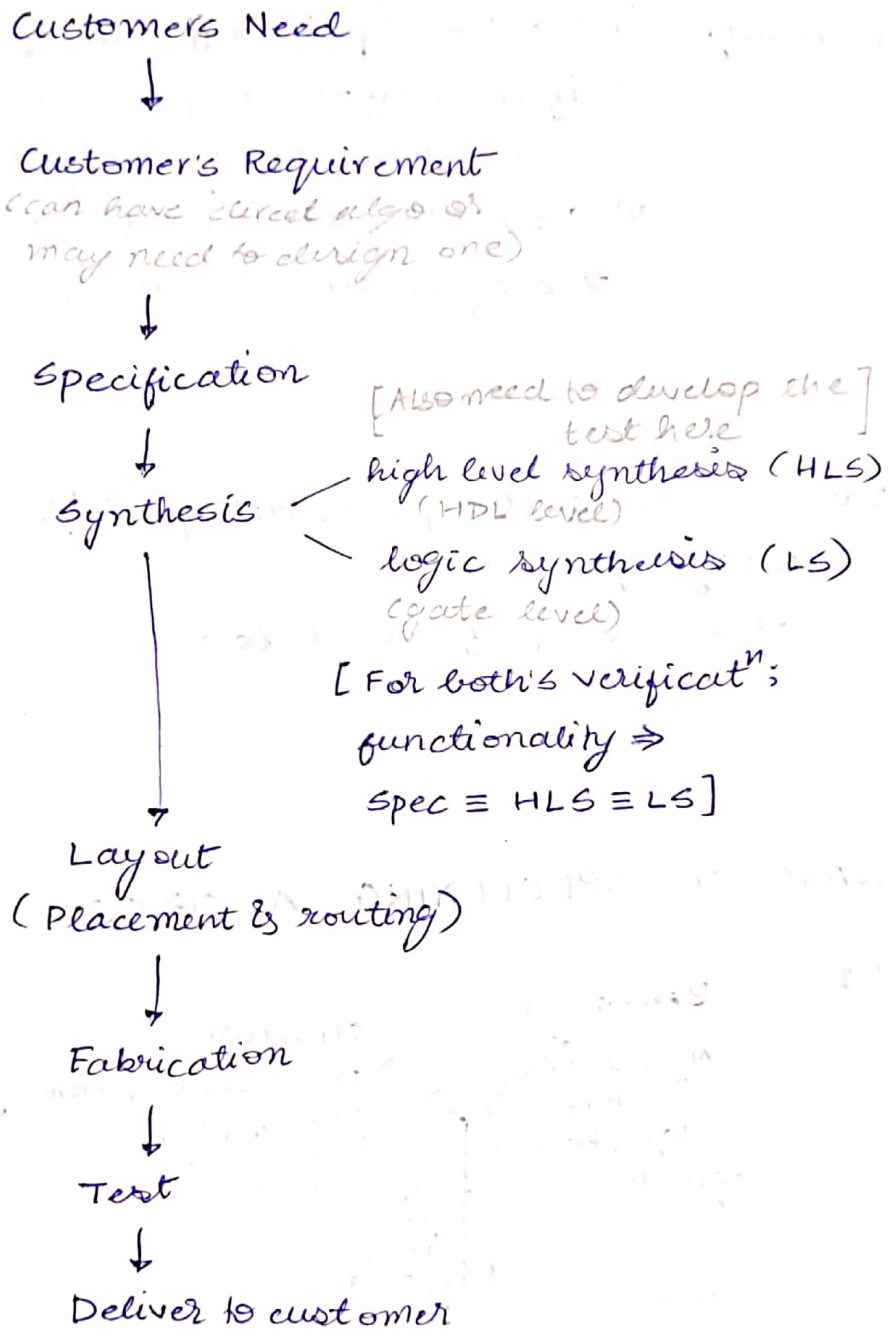
30/7

1. High level synthesis (architectural syn)
2. Logic synthesis
3. Physical design Automation

- VLSI design flow.



- optimization
  - a) time (performance) (2)
  - b) power / energy (3)
  - c) area (1)

chronological order in which features were optimized

→ area
area/performance (AP)
area/performance/testability (APT)

order of optimizat^n

- performance / area / testability (PAT)
- perf / power / area / test
- power / perf / ~~area~~ / ~~test~~ / area (PPAT / PPTA)
  ↳ nowadays not many care abt it
  ⇒ more area ∝ more static power.

{ security : i) 0 & 1 dissipate diff amt of power
⇒ if we analyze the power traits we can extract the key, encrypting a msg
ii) 0 & 1 take diff amt of time
⇒ analyze time taken }

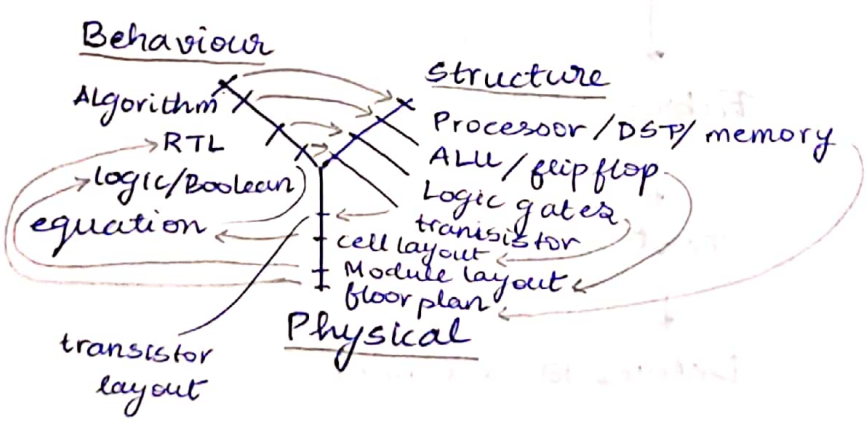
- power / perf / test / area / security (PPTAS)

{ to solve i) equilize power dissipatⁿ by doing dummy operatⁿ
→ however, perf would lower (conflicting req.) }

- power / perf / test / area / security / intelligence (PPTASI)

• WAYS OF SPECIFYING A CIRCUIT

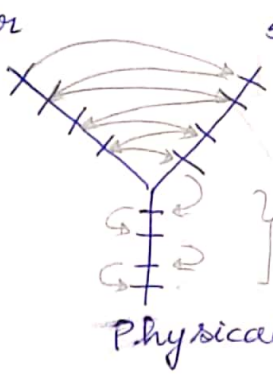
(1)



Y-CHART

⇒ Top-down approach
- based on experience we can skip some steps

(II)



Bottom up approach
(however we have some idea how much area the modules cover)
- makes structure more precise
- but not optimized

- ALGORITHMS $\xrightarrow{\text{(HLS)}}$ RTL
  (High level syntheris)
  ↳ gives macroscopic view of design
  → - components (data path) → functional units (depends on req. spec.)
  - Controller (FSM)

Data path

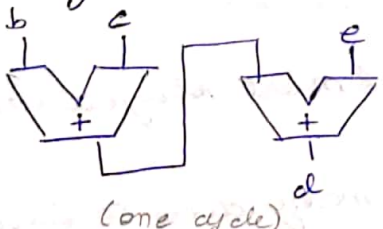
i) Functional units
ii) Memory (temporary storage)
iii) Steering logic → for communication we can have
- broadcast (BUS)
- one-to-one / point-to-point (multiplexers)

eg: We want to perform

$a = b + c$

$d = a + e$

(M1) Binding



(one cycle)

- assigning which ALU will do what operation
- this approach helps when we want do it in small amt of time
- however area not optimised

Performance should be optimized only, ~~as much~~ as long as it makes a difference

(M2) Reg CLK MUX (2 cycles)

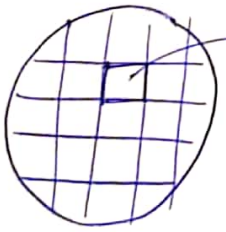
5/8 HLS

Algorithm → RTL / Data Path → FU
→ mem (reg, ~~steering logic~~)
→ steering logic
↓ Broadcast (Bus)
↓ P-to-P (MUX)

Controller → FSM

- Area + Performance
  - FU
  - SL
  - time to execute program
    - it can be
      - optimizatn parameter
      - constraint

(A) P
constraint ⇒ compromize



die

we want to produce more chips (on one 6i disc) to increase the yield (no. of working chips)

- defect density has almost stayed const. over the years
  ⇒ determines how many dies to have on a chip

⇒ Area;
  i) yield
  ii) power

- performance: optimize only as much as req.
  ⇒ excess area budget can be used on optimizing area

- need to have a fair idea about the area and performance of FUs

- Latency: delay between launching i/p and getting the output
  → but by pipelining we can improve the throughput
  → whenever there are dependcies, latency matters; otherwise throughput matters

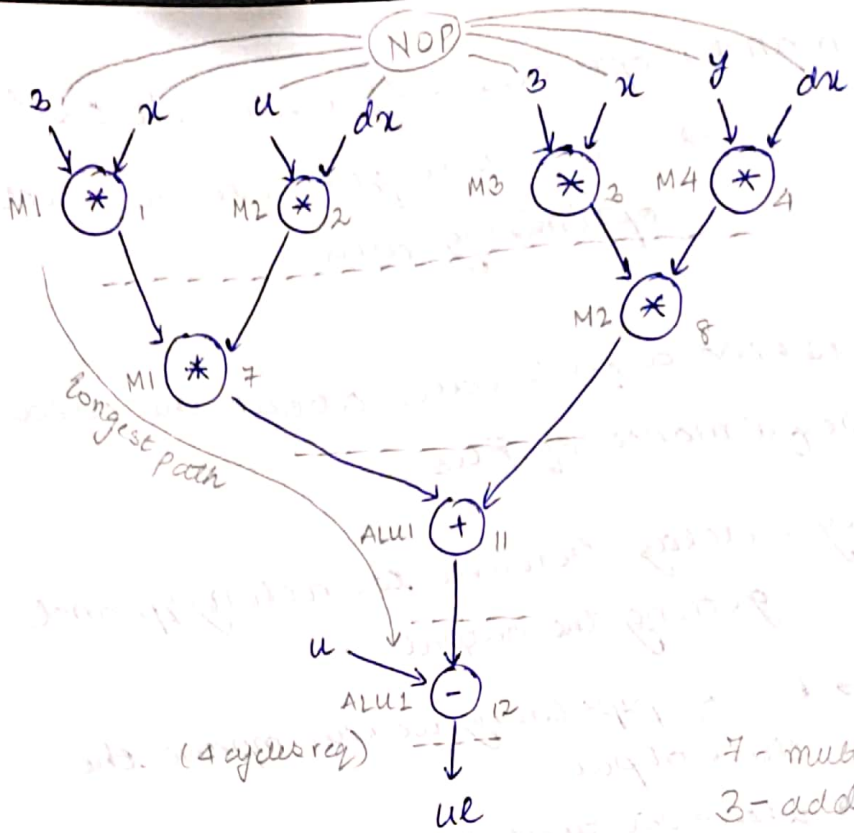
⇒ Performance
  ↙ Latency ↘ Throughput

ex $\frac{d^2y}{dx^2} + 3x\frac{dy}{dx} + 3xy = 0$

$y(0) = y$
$\frac{dy}{dx}(0) = u$
$x = 0$
} initial condⁿ

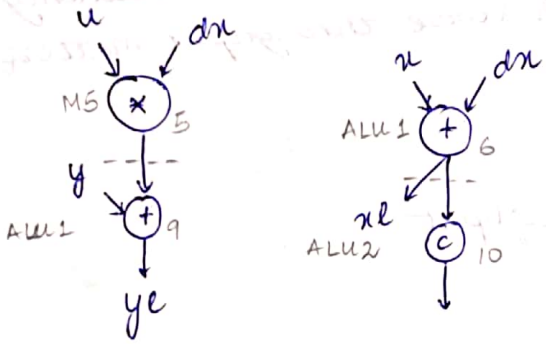
Iteration {
$xl = x + dx$
$yl = y + u\,dx$
$ul = u - 3xu\,dx - 3xy\,dx$
check {c = (xl < a) ⇒ terminate} }

- Data flow graph



7 - multiplier
3 - add
1 - sub
1 - cmpr
} ALU(5)

12 - operations



→ if we use one FU for operation then clock cycle = time taken by 2 multiplier & 2ALU

→ however if we recycle the FU then the minimum time taken = 4 cycles (to get ul)

→ we need only 5 muliplier and 2ALU

→ SCHEDULING: Placing operation in time

→ BINDING: Placing operation in space

→ Since o/p of M1 fed back to M1, it helps to develop the steering logic

⇒ our interest is in data dependency rather than the actual data

→ Upon removing the data from data flow graph we get a SEQUENCING GRAPH

attach all data nodes to NOP

→ Scheduling

time	operation
t=1	1, 2, 3, 4, 5, 6
t=2	7, 8, 9, 10
t=3	11
t=4	12

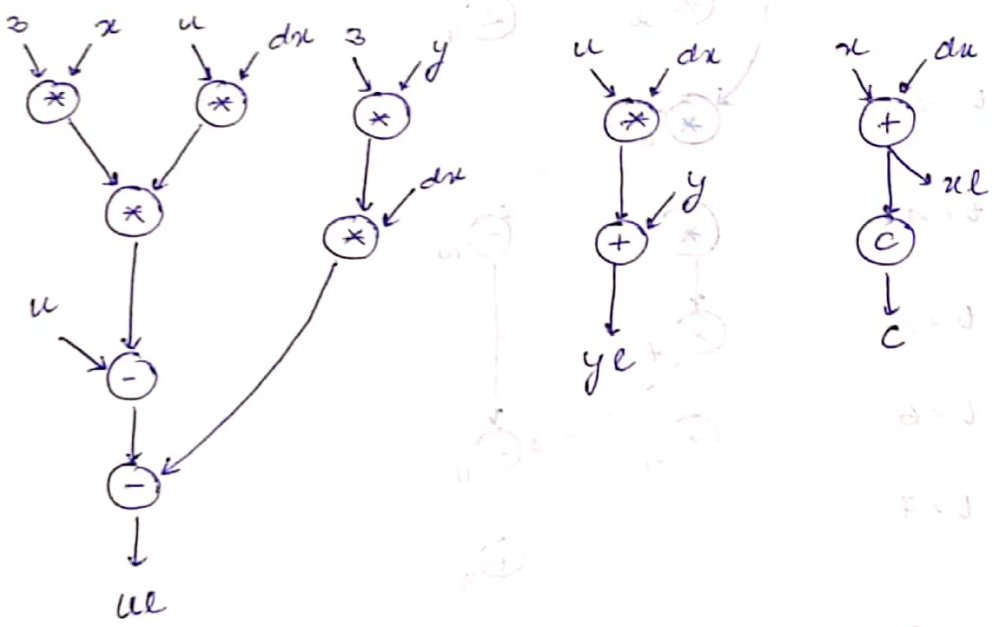
→ Binding

Multiplier	operatn
1	1, 7
2	2, 8
3	3
4	4
5	5

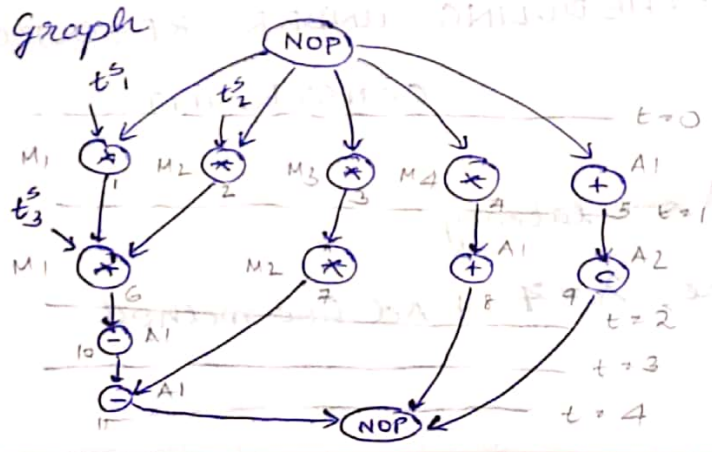
ALU	operatn
1	6, 9, 11, 12
2	10

→ moving yl cycle down 1 cycle helps us reduce 1 multiplier

6/8



⇒ Sequencing Graph



- $\lambda = 4$ (latency)
- Ad-hoc method

- Scheduling: Placement of operation in time (define start time)

  Binding: Placement of operations in space (define a physical resource)

- t_i^s : start time of FU

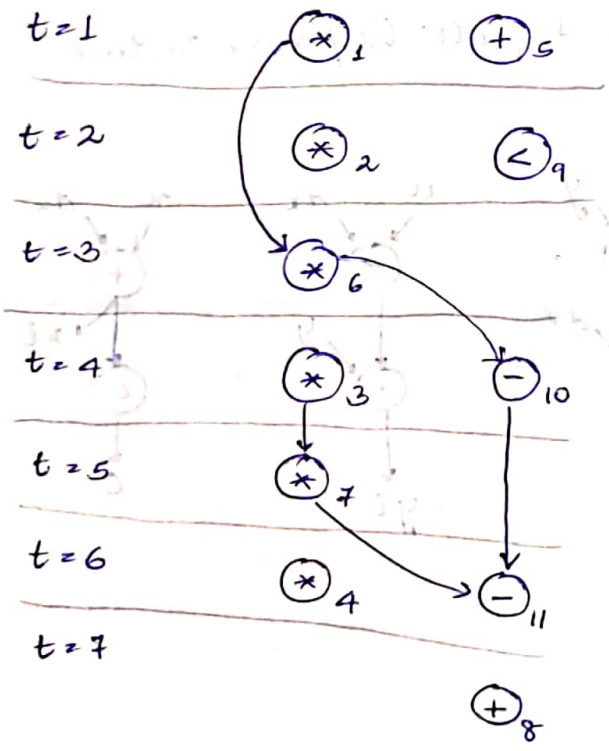
  d_i : delay of FU

  $\Rightarrow t_3^s = \max\{t_1^s + d_1,\ t_2^s + d_2\}$ — puts a constraint of scheduling order

  $\Rightarrow t_i^s = \max\{t_j^s + d_j\}$

  If we have a constraint on resources as well

  → 1 Mul, 1 ALU



⇒ 'SCHEDULING UNDER RESOURCE CONSTRAINT'

$\lambda \leftarrow$ latency

Here, $\lambda = 7$ + Ad-hoc method

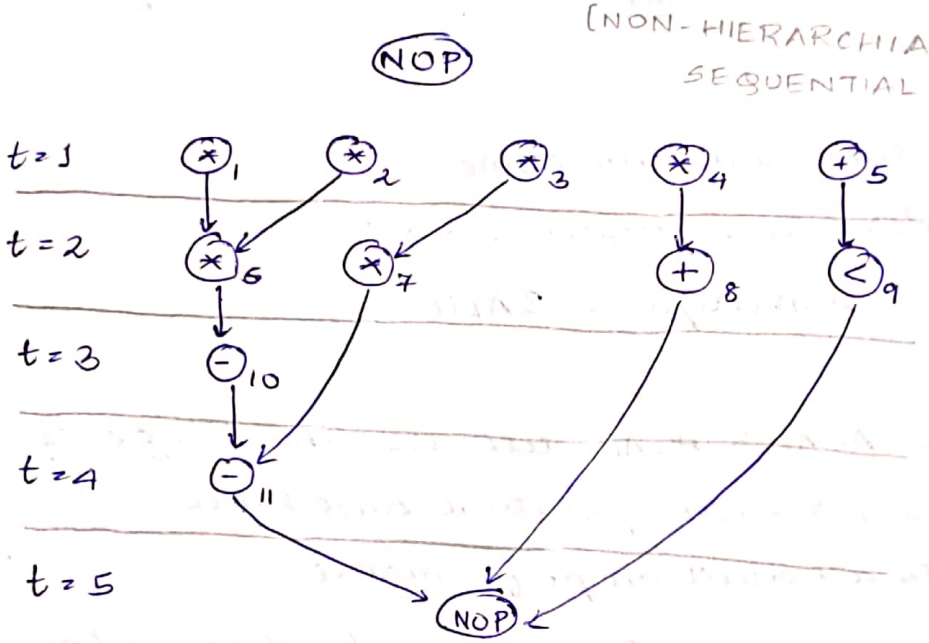
Need to develop a systematic methodology

- ASAP: As soon As possible
  - ⇒ schedule any node ASAP if predecessors have been scheduled

$$t_i^s = \max(t_j^s + d_j) \qquad j \rightarrow \text{Predecessors}$$

→ execution time of NOP = 0 (since it's a function we added)

[NON-HIERARCHIAL SEQUENTIAL GRAPH]



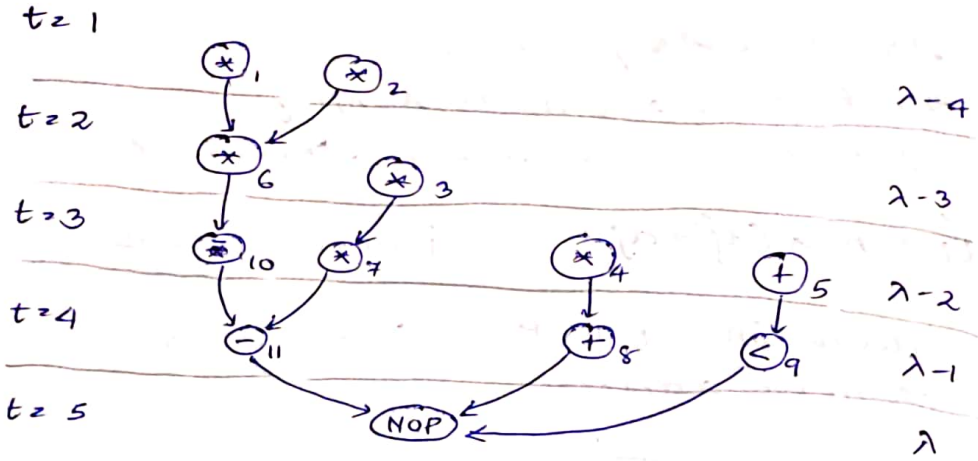
$$\lambda = 5 - 1 = 4$$

time taken by both NOP

⇒ This gives the optimal performance however area not optimized

- ALAP: As Late As Possible
  - ⇒ schedule operations from the bottom

$$t_i^s = \min(t_j^s - d_j) \qquad j \rightarrow \text{predecessor}$$



⇒ time remains same

⇒ Resources needed now:

2 multiplier, 3 ALU (if multiplier costlier than ALU, then we choose ALAP)

for scheduling

→ Now, ASAP & ALAP tell us the range of time for each operation and hence puts a bound on performance

eg: operation 1 & 2 has to be done in t=1

operation 3 can be done in t=1 or t=2

2/8

Having ASAP & ALAP we know the performance constraint. We can optimize area to reduce power consumptn now.

→ operation 1, $t_1^S = 1$ (ASAP)

$t_1^S = 3$ (ALAP, if λ=6) ⇒ provides "SLACK"

{Hence it can start at t = 1, 2, 3 (for λ=6)}

⇒ flexibility (can optimize area)

→ Variants: LP, ILP, MILP, ZOLP
(MILP: mixed; ZOLP: zero one)

→ Need to be able to map scheduling program to ILP

Qn 400 wooden planks, 450 person hours
To construct a

chair : 5 wood + 10 person hrs = Rs 100 profit
table : 10 wood + 15 person hrs = Rs 150 profit

How many chairs and tables to maximize profit?

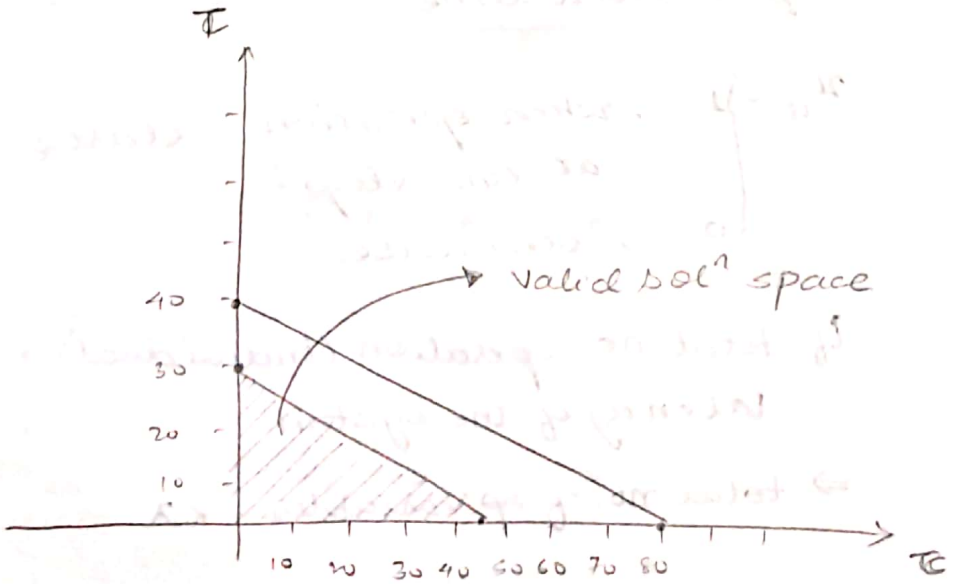
→ Profit = $100C + 150T$

$C \rightarrow 5W + 10P$ $\min\{\frac{400}{5}, \frac{450}{10}\}$
$T \rightarrow 10W + 15P$ $\min\{\frac{400}{10}, \frac{450}{15}\}$

(if we want to make only chair/table)

$W \leq 400$, $P \leq 450$

$5C + 10T \leq 400$ (constraint on wood)
$10C + 15T \leq 450$
$C \geq 0$
$T \geq 0$



- But we might get more profit by simply selling woods and human hours
We need to determine the selling rate/price

$W \geq 0$
$P \geq 0$ } price of wood & human hours

$\Rightarrow 5W + 10P \geq 100$ → amt greater than made by selling a chair

$10W + 15P \geq 150$

$\Rightarrow$ 'DUAL' of the problem

13/8

LP - real variable
ILP - only integer
MLP - both real and integer variables
ZOLP - only zero & one allowed

everything is a subset of this; any no. can be expressed as 0, 1

SCHEDULING

- Placing operation in time
  - define start time of every operation

① Unique start time

$$x_{il} = \begin{cases} 1 & \text{, when operation } l \text{ starts at time step } i \\ 0 & \text{, otherwise} \end{cases}$$

[variables are type $\Rightarrow$ ZOLP]

If total no. operation (individual) = k
latency of the system = λ

$\Rightarrow$ total no. of variables = $k\lambda$

since any operation can have a unique time step only.

$$\sum_i x_{il} = 1$$

eg: $x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 1$ (for $\lambda = 5$, in earlier problem)

but we know that 1 cannot be scheduled at $t = 3, 4, 5$

$\Rightarrow x_{11} + x_{21} = 1$

$x_{16} + x_{26} + x_{36} + x_{46} + x_{56} = 1$

$x_{16} = 0$ (due to ASAP); $x_{46} + x_{56} = 0$ (due to ALAP)

$\Rightarrow x_{26} + x_{36} = 1$

$x_{18} + x_{28} + x_{38} + x_{48} + x_{58} = 1$

$\Rightarrow x_{28} + x_{38} + x_{48} + x_{58} = 1$

we simplified the equation, however we can solve for solution even with the original set of equations

(2) Dependency relationship must be respected

If l^{th} operation is dependant on k^{th} operation

$$\sum i \cdot x_{il} \geq \sum j \cdot x_{jk} + d_k$$ → delay of gate → fixed

$$\sum i \cdot x_{il} - \sum j \cdot x_{jl} - d_j \geq 0$$

($d_j = 1$)

eg: $2x_{26} + 3x_{36} - x_{11} - 2x_{21} - 1 \geq 0$

if $x_{26} = 1$ then x_{21} has to be 0

if $x_{26} = 1$ then either x_{11} or $x_{21} = 1$

$$2x_{28} + 3x_{36} + 4x_{48} + 5x_{58} - x_{14} - 2x_{24} - 3x_{34} - 4x_{44} - 1 \geq 0$$

↳ delay = 1

③ Resource constraint

If we have the constraint of only 2 multiplier present & 2 ALU

then $x_{11} + x_{12} + x_{13} + x_{14} \leq 2$

$x_{15} \leq 2$ } $t = 1$

$t = 2$: $x_{21} + x_{22} + x_{23} + x_{24} + x_{26} + x_{27} \leq 2$

$x_{28} + x_{29} + x_{25} \leq 2$

NOTE : all the equations in ① ② ③ have to be simultaneously satisfied

19/8

- ASAP scheduling gives the best timing
- Max/upper bound (λ) given by ALAP
- Their difference gives the 'margin / SLACK'

- ## Optimization

For, a_1 multipliers ⇐ 4 unit area

a_2 ~~multiplie~~ ALUs ⇐ 1 unit area

a) Area optimization

$C = 4a_1 + a_2$ → minimize

↓ cost

b) Performance optimizat^n

$a_1, a_2, \ldots$ = fixed

$\lambda \leftarrow$ need to be minimized

$\Rightarrow \min \sum_l \sum_i i \cdot x_{il}$ → minimize the time when an operat^n is started

i.e. minimize $\{ 1 \cdot x_{11} + 2 \cdot x_{21} + \cdots + x_{12} + 2 \cdot x_{22} + \cdots \}$

c) Co-optimization

→ need to bring all parameters to same scale and assign some weight to each parameter

$C = S \times \{\text{normalised area}\} + (1-S) \times \{\text{normalised } \frac{\text{time}}{\text{perf}}\}$ $\left(\text{perf} \propto \frac{1}{\text{time}}\right)$

☆ SIMPLEX

• LIST SCHEDULING

make a list of eligible candidates and pick a few which satisfy the constraint
(shortlisting done by assigning priority)

Under resource constraint

→ assuming area constributed by FU; routing logic area negligible

- need to assign weights / priority number
  - start from ALAP and assign each level a number starting from 0 (for NOP)
    but this way operation 1, 2, 3 are alloted 4
  - we some operations have same weight then
    - we can choose randomly
    - choose from a list

eg: $U_1 = \{1, 2, 3, 4, 6, 7\}$ (multiplier)

$t=1$ $U_2 = \{5, 8, 9, 10, 11\}$ (ALU) → unscheduled

$\Rightarrow S_1^1 = \{1, 2\}$

$S_1^2 = \{5\}$

---

then $U_1 = \{3, 4, 6, 7\}$

$t=2$ $U_2 = \{8, 9, 10, 11\}$

$\Rightarrow S_2^1 = \{3, 6\}$ → based on value assigned to operation

$S_2^2 = \{9\}$

$t=3$ $U_1 = \{4, 7\}$

$U_2 = \{8, 10, 11\}$

$\Rightarrow S_3^1 = \{4, 7\}$

$S_3^2 = \phi$

$t=4$ $U_1 = \phi$

$U_2 = \{8, 10, 11\}$

$S_4^1 = \phi$

$S_4^2 = \{8, 10\}$

$t=5$ $U_1 = \phi$

$U_2 = \{11\}$

$S_5^1 = \phi$

$S_5^2 = \{11\}$

20/8 • Number of cycles for diff. FU is diff

eg: multiplier = 2 cycles

ALU = 1 cycle

$t=1$: $U_1 = \{1, 2, 3, 4, 6, 7\}$

$U_2 = \{5, 8, 9, 10, 11\}$

$S_1^1 = \{1, 2\}$

$S_1^2 = \{5\}$

---

t=2: $U_1 = \{3, 4, 6, 7\}$

$U_2 = \{8, 9, 10, 11\}$

$P_1 = \{1, 2\}$ (operations in progress)

$P_2 = \phi$

~~$S_2^1 =$~~

Now, $|S_i^1| + |P_1| \leq 2$

$\Rightarrow S_2^1 = \phi$

$S_2^2 = \{9\}$

---

t=3: $U_1 = \{3, 4, 6, 7\}$

$U_2 = \{8, 10, 11\}$

$P_1 = \phi$, $P_2 = \phi$

$S_3^1 = \{3, 6\}$; $S_3^2 = \phi$

---

t=4: $U_1 = \{4, 7\}$; $U_2 = \{8, 10, 11\}$

$P_1 = \{3, 6\}$; $P_2 = \phi$

$S_4^1 = \phi$ $S_4^2 = \phi$

---

t=5: $U_1 = \{4, 7\}$; $U_2 = \{8, 10, 11\}$

$P_1 = \phi$; $P_2 = \phi$

$S_5^1 = \{4, 7\}$; $S_5^2 = \{10\}$

---

t=6: $U_1 = \phi$; $U_2 = \{8, 11\}$

$P_1 = \{4, 7\}$; $P_2 = \phi$

$S_6^1 = \phi$; $S_6^2 = \phi$

---

t=7: $U_1 = \phi$; $U_2 = \{8, 11\}$

$P_1 = \phi$; $P_2 = \phi$

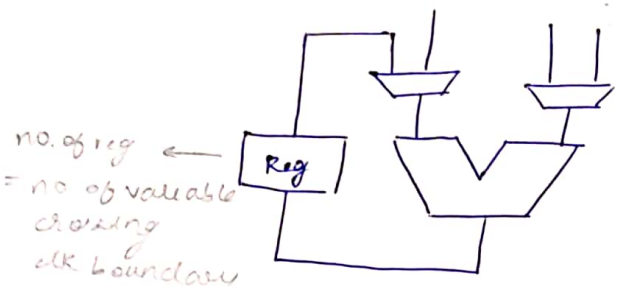
$S_7^1 = \phi$; $S_7^2 = \{8, 11\}$

- 2 multiplier → M_1 (2 cycle, more expensive)
  → M_2 (1 cycle)

2 ALU → A_1 (1 cycle)
→ A_2 (1 cycle)

22/18

$a = b + c$
$d = a + e$ } a crossing clk boundary

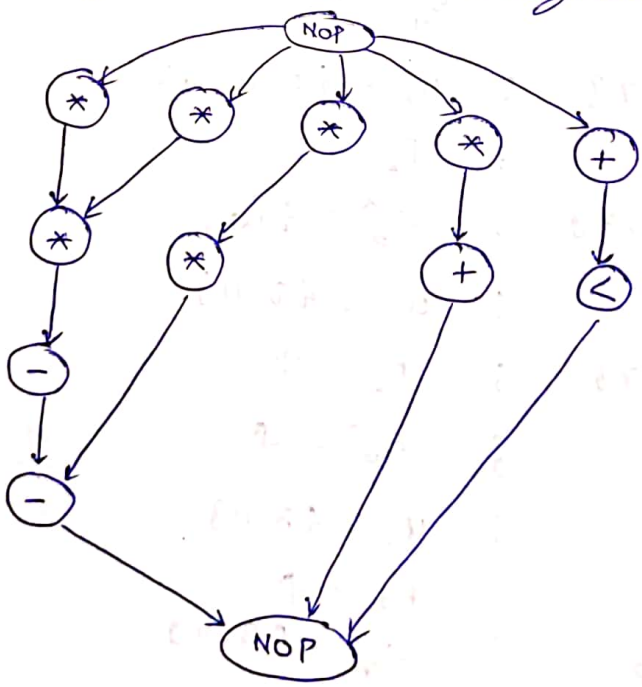


no. of reg ← = no of variable crossing clk boundary

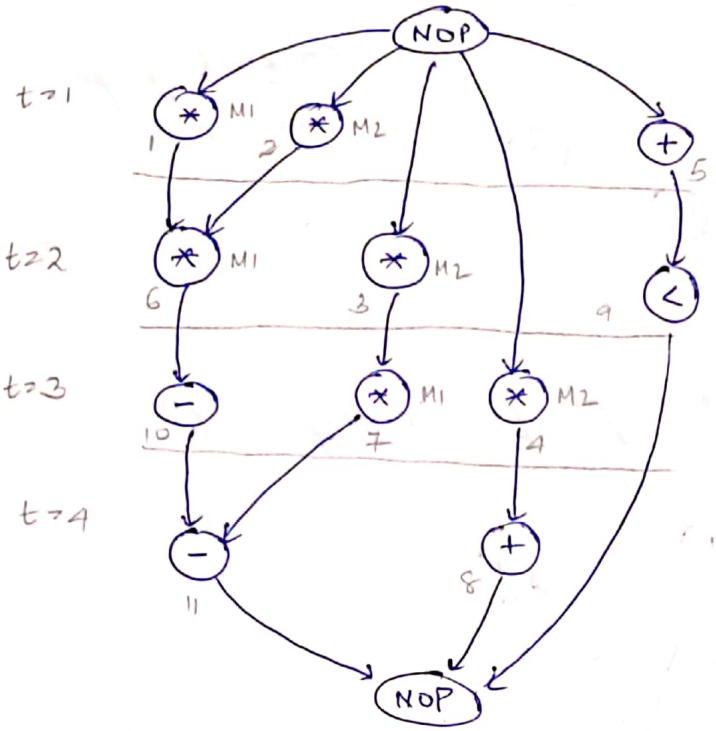
Binding of operation to physical resource

share resource (if no conflicting reg. i.e. not in concurrent use)

Binding of variables to register



Afte scheduled sequencing (2-M, 2-ALU)



Bind 2 Multi to 6 multiplicatⁿ operatⁿ such that operations which are concurrent don't get same resource

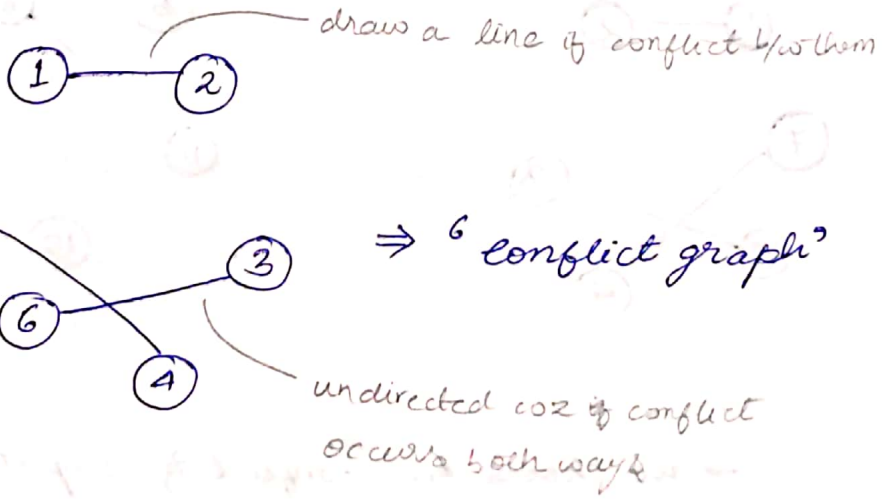
- BINDING OPERATION TO PHYSICAL RESOURCE

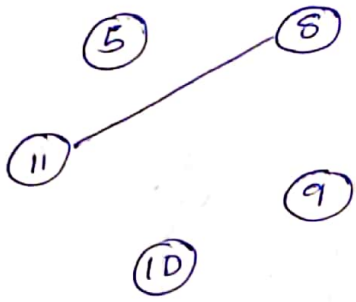
Based on above mapping we can devise a graph

1) conflict graph
2) compatibility graph

- Build them based on operation's type

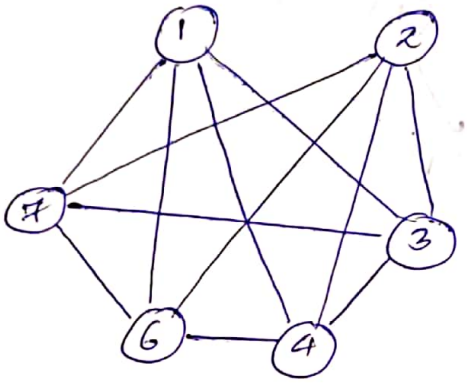
For Multiplication operatⁿ:





conflict graph for ALU

compatibility graph for multiplicat^n:

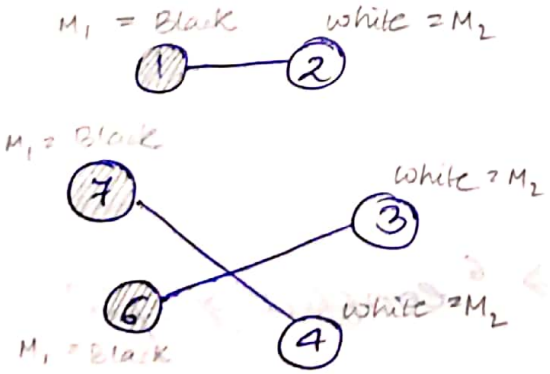


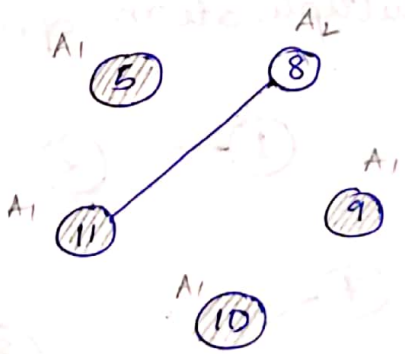
* graph colouring problem

(adjacent nodes shouldn't get same colour, colour graph with min^m colours)

chromatic number

→ assigning resources based on graph is a similar problem



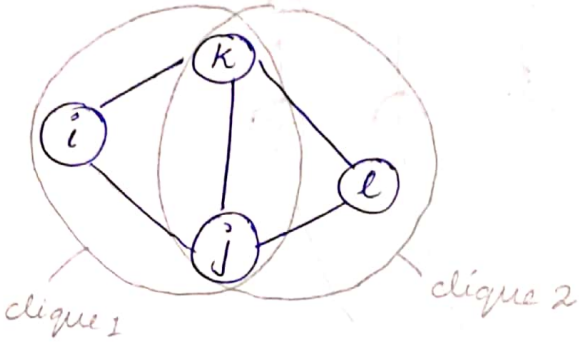


→ we choose b/w conflict & compatibility graph based on which is more sparse

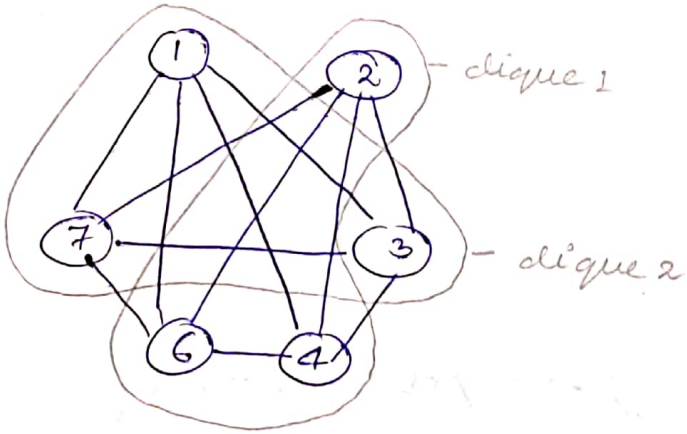
complete graph: every node connected to every other node

eg: 

clique: subgraph inside a graph which is complete



→ we want to find the minimum number of cliques and assign 1 resource to one clique



- Solving with ILP

Input: scheduled sequencing graph.

x_{il} → operat^n l starts at time step i

↳ given i/p we know their values

let y_{lk} → operat^n l binds with resource K

$1 < l < n$ (total operat^n)

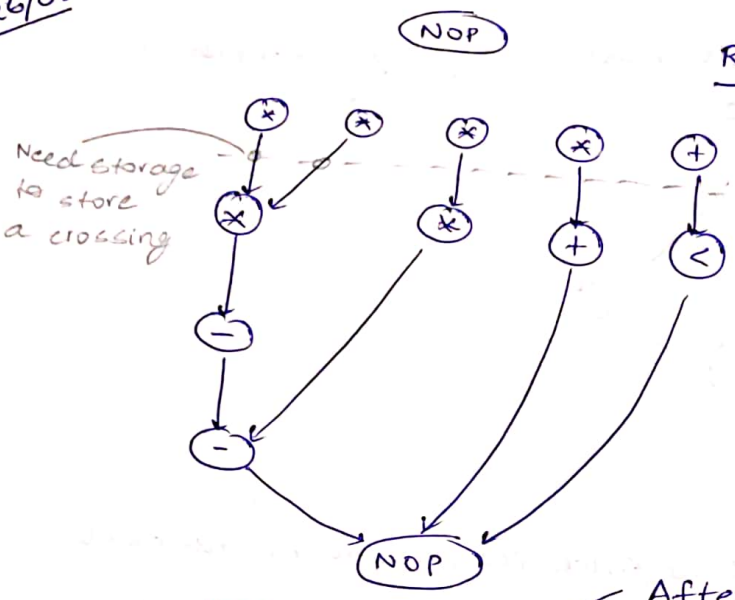
$1 < k < m$ (total resources)

① uniqueness

$$\sum_k y_{ik} = 1$$

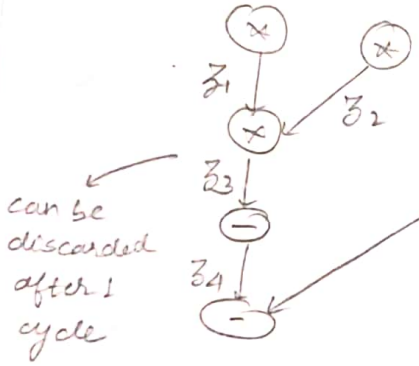
26/08

REGISTER SHARING



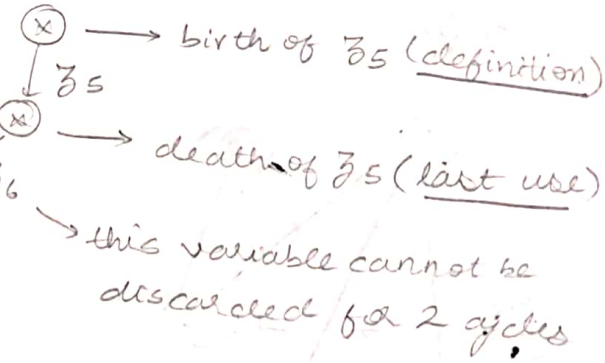
share registers if not concurrently used

DFG



can be discarded after 1 cycle

After scheduling and operatn binding



birth of z_5 (definition)

death of z_5 (last use)

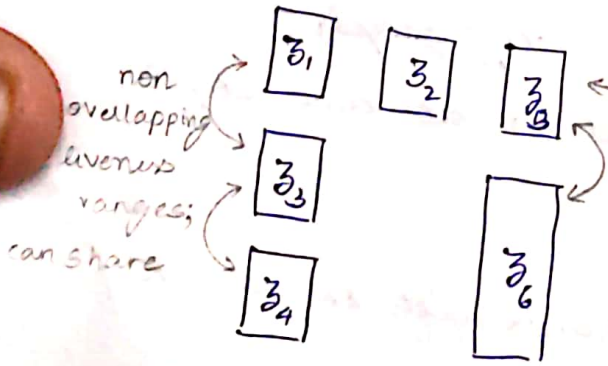
lifetime of variable

⇒ LIVENESS RANGE

z_6 → this variable cannot be discarded for 2 cycles

Need one register per liveness range

⇒ no overlapping liveness range variables can share register



← concurrent, cannot share

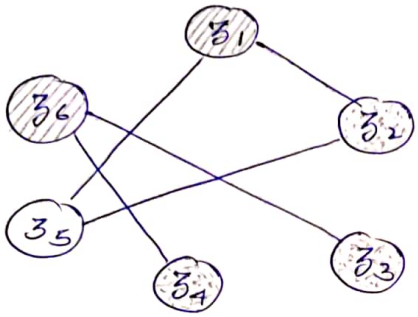
⇒ Liveness Ranges of variable

$R_1 = \{z_1, z_3, z_4\}$

$R_2 = \{z_2\}$

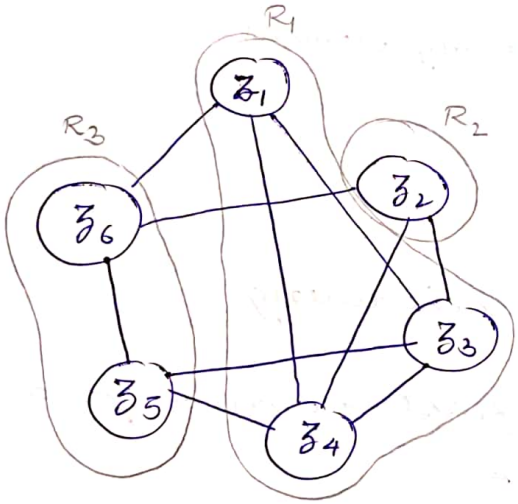
$R_3 = \{z_5, z_6\}$

To systematize the problem transform it into a graph



[conflict graph]

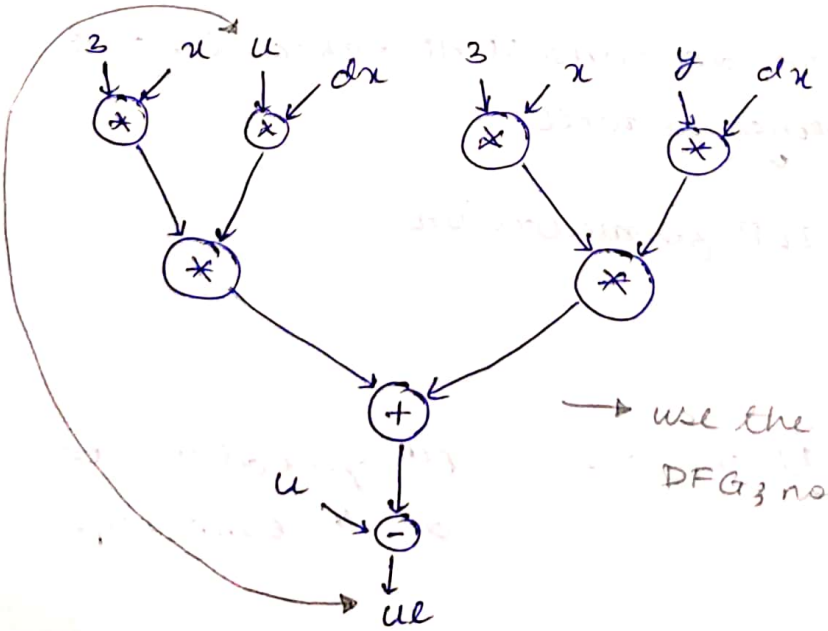
→ Graph colouring



[compatibility graph]

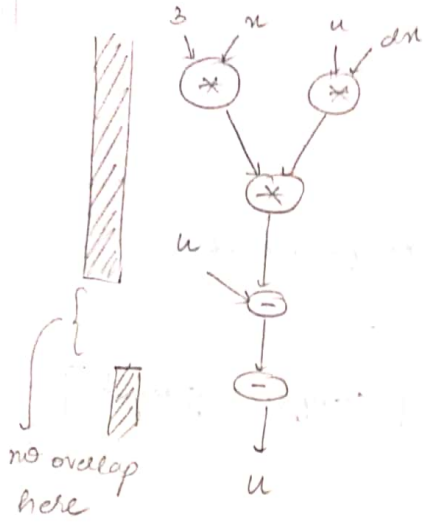
→ Minimum clique

→ storage elements also have to store external variable



liveness range of u

→ we the optimized DFG; not this



27/10/06

Scheduling
↓
Binding → FU
↓ → Memory elements
- Register
- If multiported memory (assigned of variable to memory port)

Helps develop 'Steering logic' (assuming till now that its area is minimal)

⇒ # resources → scheduling

- If we do binding first, we allot operations to resources first
  Helps when steering logic has significant area

- We can do <u>concurrent scheduling & binding</u> as well

  → ILP formulation
  Need to place operation in time and space together

  1) $x_{il} = 1$, l^{th} operation starts at i^{th} time step

$b_{lr} = \begin{cases} 1 & \text{if } l^{th} \text{ operation carried out by } r^{th} \text{ resource} \\ 0 & \text{else} \end{cases}$

Total no. of operation = n_{op}

latency = λ

total resources = n_{res}

total variable (x_{il}) = $n_{op} \times \lambda$

total variable (b_{lr}) = $n_{op} \times n_{res}$

optimization

Minimization of latency

$$\sum_i \sum_l i \cdot x_{il}$$

Minimization of resources

$$\sum \sum w_i a_r$$

a_r → area

w_i → weight assigned to area

Total cost

$$S(\text{norm. latency}) + (1-S)(\text{norm. latency})$$

constraints

1) unique resource binding

$$\sum_r b_{lr} = 1 \qquad r : T(v_l)$$

→ should belong to relevant set (ALU can't do multiplicatn)

eg: $b_{11} + b_{12} = 1$

$b_{21} + b_{22} = 1$

$b_{31} + b_{32} = 1$

$b_{83} + b_{84} = 1$

$b_{83} + b_{84} = 1$

$b_{93} + b_{94} = 1$

2) unique start time

$$\sum_{i=1}^{\lambda} x_{il} = 1$$

eg: $x_{11} + x_{21} + \underbrace{x_{31} + x_{41} + x_{51} + x_{61}}_{0} = 1$

3) Dependency

l^{th} operation dependant on k^{th} operation then it cannot be started before k^{th} operation

$$\sum_i i \cdot x_{il} \geq \sum_i i \cdot x_{ik} + \underbrace{d_k}_{1}$$

eg: $2 \cdot x_{26} + 3 \cdot x_{36} - 1 \cdot x_{11} - 2 x_{21} - 1 \geq 0$

if $d_k = 2$

then $x_{26} = 1 \Rightarrow x_{11}$ & $x_{21} = 0$

$\Rightarrow$ not possible

4) limited resource

$t=1$: $x_{11} + x_{12} + x_{13} + x_{14} + x \leq 2$

$t=2$: $x_{21} + x_{22} + x_{23} + x_{24} + x_{26} + x_{27} \leq 2$

$+ \underbrace{x_{11} + x_{12} + x_{13} + x_{14}}_{\text{if } d_k = 2 \text{ and they are still running}}$ $\underbrace{\qquad}_{\text{cannot be scheduled}}$

2) Unique assignment

$$\sum_{i=1}^{\lambda} x_{il} = 1$$

eg: $x_{11} + x_{21} + \underbrace{x_{31} + x_{41} + x_{51} + x_{61}}_{0} = 1$

3) Dependency

l^{th} operation dependant on K^{th} operation then it cannot be started before K^{th} operation

$$\sum_i i \cdot x_{il} \geq \sum_i i \cdot x_{iK} + \underbrace{d_K}_{1}$$

eg: $2 \cdot x_{26} + 3 \cdot x_{36} - 1 \cdot x_{11} - 1 x_{21} - 1 \geq 0$

if $d_K = 2$

then $x_{26} = 1 \Rightarrow x_{11}$ & $x_{21} = 0$

$\Rightarrow$ not possible

4) Limited resource

$t=1$: $x_{11} + x_{12} + x_{13} + x_{14} + x \leq 2$

$t=2$: $x_{21} + x_{22} + x_{23} + x_{24} + \underbrace{x_{26} + x_{27}}_{\text{cannot be scheduled if } d_K = 2} \leq 2$

$+ \underbrace{x_{11} + x_{12} + x_{12} + x_{14}}_{\text{if } d_K = 2 \text{ and they are still running}}$

$t=3$:

$x_{21} + x_{22} + x_{23} + x_{24} + x_{26} + x_{27}$

$+ x_{33} + x_{34}$

6) Binding

$$\sum_{l} b_{lr} \sum_{p=i-d_l+1}^{i} x_{pl} \leq 1$$ → non-linear due to multiplication

operations binded with a resource shouldn't be concurrent

eg: $b_{11}x_{11} + b_{21}x_{12} + b_{31}x_{13} + b_{41}x_{14} \leq 1$

(for resource = 1, time step = 1)

$b_{12}x_{11} + b_{22}x_{12} + b_{32}x_{13} + b_{42}x_{14} \leq 1$

(for resource = 2, time step = 1)

29/08 Quiz 1 Solution

1) a) ASAP and ALAP both give max. performance (upper bound)

b) Cannot have min^m resources by ASAP, gives min^m time

c) Binding only assigns physical resource, not time step

2)

HW write ILP for chain

- scheduling and binding

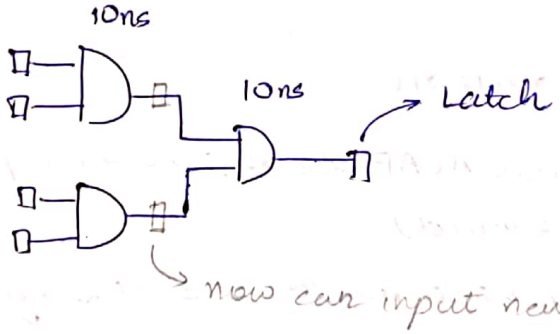
→ 4 cycles

→ cycle time = max (delay multi, delay ALU)
= 10ns

→ total itration = 40ns

↳ to reduce time and increase throughput we need to do pipelining

eg:

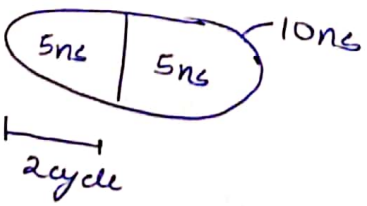


Initially cycle time = 20ns

finial cycle time = 10ns

→ easier to split directed acyclic graph (no loops)

- Data Introduction Interval: how often we can enter new data



If cycle time = 2.5ns

→ DII = 5ns

PIPELINI

1) Structu
- reso
- no

eg: mu

2) Fun

PIPELINING

1) Structural Pipelining

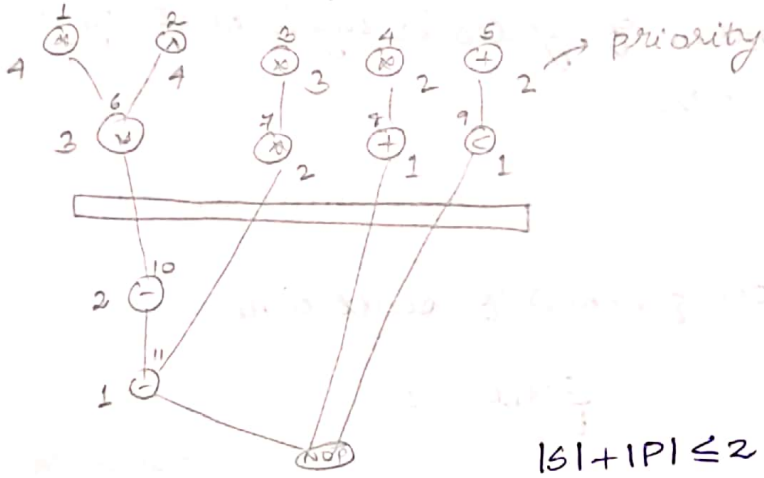
- resources are pipelined
- non-hierarchical sequencing graph used

eg: multiplier

Latency = 2 cycle

DII = 1 cycle

2) Functional Pipelining



$|S|+|P| \leq 2$

List scheduling:

t=1 : $U_1 = \{1,2,3,4\}$ $U_2 = \{5\}$

$P_1 = \phi$ $P_2 = \phi$

$S_1^1 = \{1,2\}$ $S_2^1 = \{5\}$

---

t=2 $U_1 = \{3,4\}$ $U_2 = \{9\}$

$P_1 = \{1,2\}$ $P_2 = \phi$

$S_1^2 = \{3,4\}$ $S_2^2 = \{9\}$

due to pipelining

---

t=3 $U_1 = \{6\}$ $U_2 = \{\}$

$P_1 = \{3,4\}$ $P_2 = \phi$

$S_1^3 = \{6\}$ $S_2^3 = \phi$

	$P_1 = \{6\}$	$P_2 = \phi$
	$S_1^4 = \{7\}$	$S_2^4 = \{8\}$
$t=5$	$u_1 = \phi$	$u_2 = \{10\}$
	$P_1 = \phi$	$P_2 = \phi$
	$S_1^5 = \phi$	$S_2^5 = \{10\}$
$t=6$	$u_1 = \phi$	$u_2 = \{11\}$
	$P_1 = \phi$	$P_2 = \phi$
	$S_1^6 = \phi$	$S_2^6 = \{11\}$

- number of cycles is high but cycle time is less

- **ILP**

1) Uniqueness of start time

$$\sum_i x_{il} = 1$$ (every operation should be started only once)

eg:

2) Dependancy

if k^{th} operation depends on l^{th} operation

$$\sum i \cdot x_{il} \leq \sum_j x_{jk} + d_k$$

3) Resource

$$x_{11} + x_{12} + x_{13} + x_{14} \leq 2$$

$$x_{15} \leq 1$$

$$x_{21} + x_{22} + x_{23} + x_{24} + x_{26} + x_{27} + \underbrace{\quad} \Big\} \leq 2$$

no need to add due to Pipelining

[if δ = data introduct^n interval

$t, t+\delta, t+2\delta, \ldots$ time step operation should be concurrent]

03/09

A

B

A

B

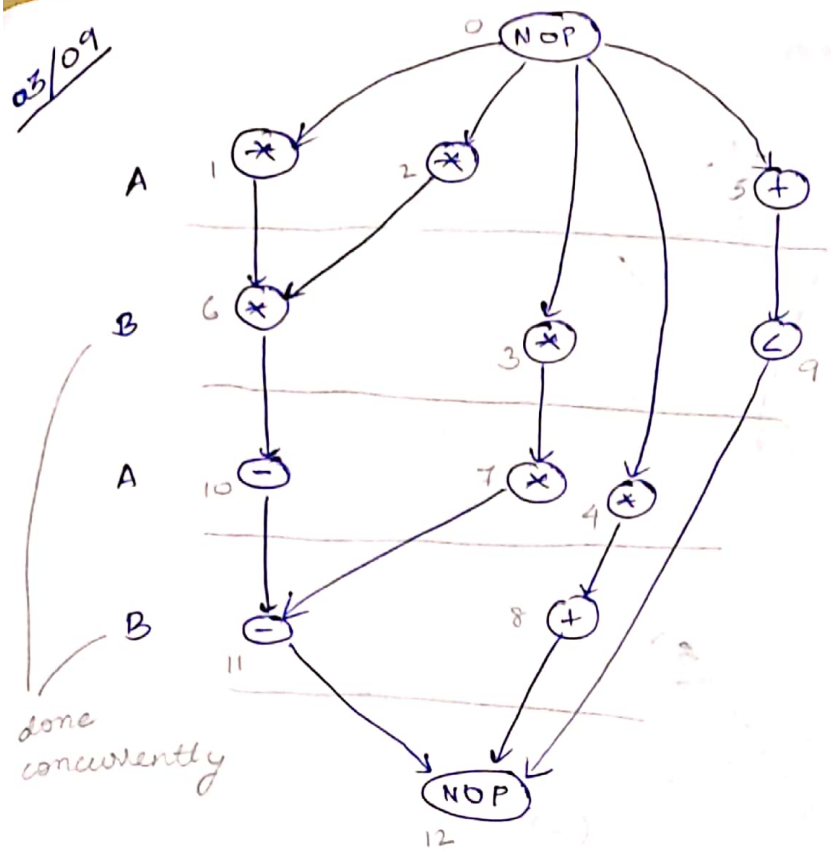
done concurr

- IL

(P

Scanned by CamScanner

05/09



4 cycles latency can be split into 2 using pipelining

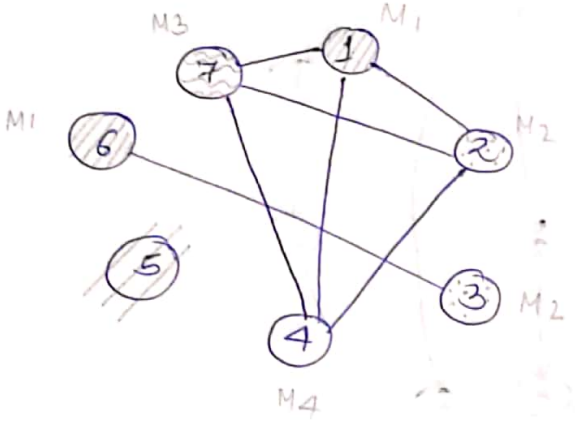
- ILP resource constraint contd.

$$\left(\begin{array}{c}\text{potential operations}\\ \text{in cycle 1}\end{array}\right) + \left(\begin{array}{c}\text{potential operations}\\ \text{in cycle 3}\end{array}\right) \leq \text{Resources}$$

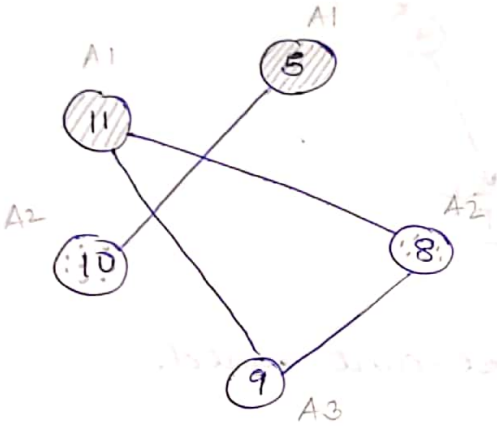
$$\left(\begin{array}{c}\text{pot. op. in}\\ \text{cycle 2}\end{array}\right) + \left(\begin{array}{c}\text{pot. op. in}\\ \text{cycle 3}\end{array}\right) \leq \text{Resources}$$

- Let's say for binding we have scheduled sequencing graph.
  - earlier throughput = $\frac{11}{4}$
  - now throughput = $\frac{11}{2}$

- conflict graph



⇒ need 4 multiplier at the same time



⇒ need 3 ALUs

- pipelining improves throughput but resource requirement also increases

- PPTAS1
  - P — power
  - T — testability

Power usage is static & dynamic
- static: leakage
- dynamic: switching activity
- both depend on the inputs

Dynamic power - depends on how many outputs are switching upon switching the inputs

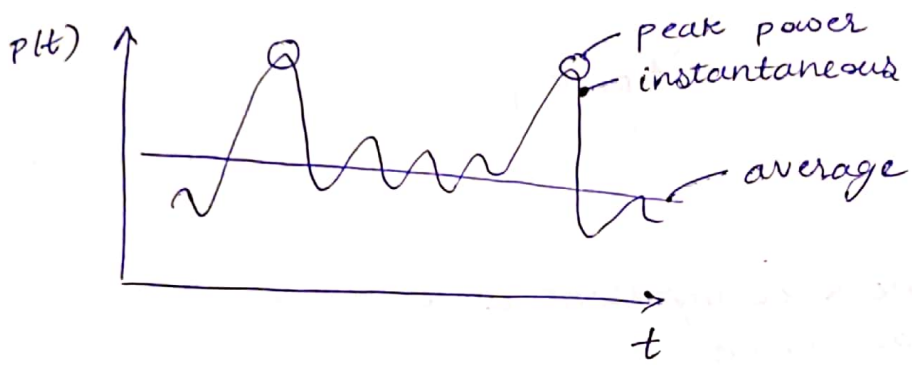
Static power - which MOS is on depends on the input; hence those would consume power

- at idle state what set of input to apply so that leakage is minimum.
- minimize the switching activity (however i/p comes from environ.)
- power = energy / time

  $= \alpha C V^2 f$ (average)

  ↳ number of times it toggles

  $P(t) = V(t)\, I(t)$ (instantaneous)



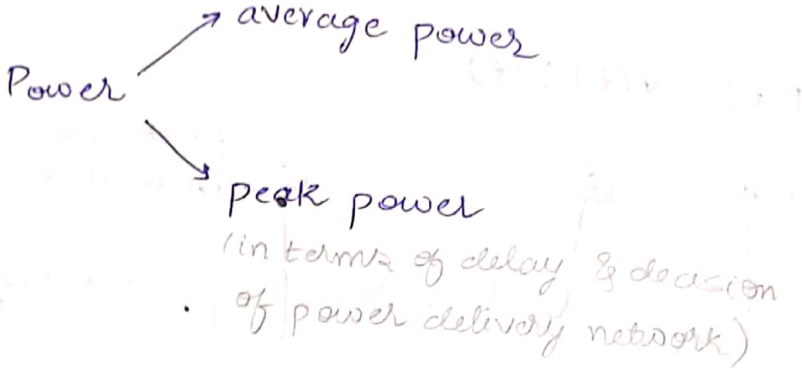
- Average power determines the heat dissipitated and hence defines all cooling requirement
- avg. power relates to energy consumptn and hence determines our battery life
- heating determines burnout of chip

- we mostly define peak as energy consumed over a clock period
  - at peaks we don't know what the device output would be
  - if V_{DD} goes down, then charging time ↑ ⇒ delay ↑ ⇒ lose synchronisatn ⇒ wrong result

    correct result is crucial

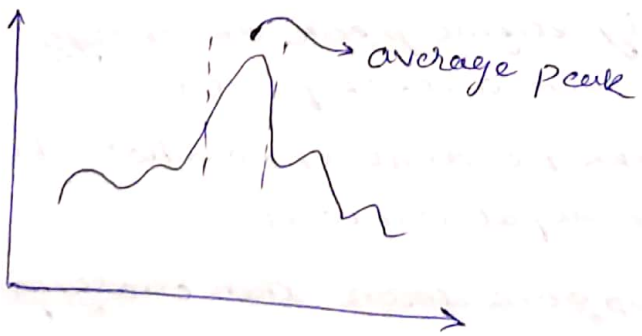
09/09

- average of power is important in the cooling system
- radio, display and processor are the power dissipation sources in phones
- radio & display are mostly const. over the generation



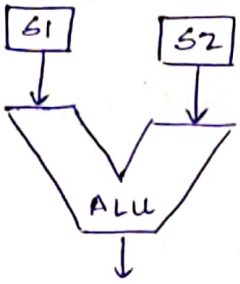
- peak is important to decide the powerline
- if the power supply isn't strong enough and voltage is higher there is drop in voltage called voltage drop
- Today chips are outputing at 0.8 - 1V & $V_T = 0.4V$, so we don't have enough margin

  $V_{DD} - V_T \uparrow \Rightarrow$ delay ↑, so it may not be able to propagate the signal



- real peak is instantaneous

- Dynamic power is coming from switching. Switching / Toggle count will tell us the dynamic post.



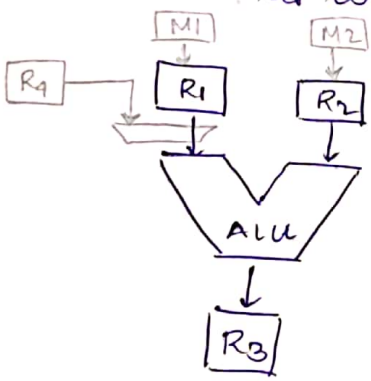
Need to find power dissipation of ALU from switches

Toggle = F(input)

- Static power = no. of gates

Static power has become low coz of finfets, high-K etc.

- Our main aim is to reduce dynamic power



$R_3 = R_1 + R_2$

→ synchronize I/P switching

↓

Register Binding

↳ for reg. binding we saw 2 graphs, whichever gives less power that'll have good synchronizatn

- T → testability

Modelling faults:

- single stuck-at faullt (SSAF)

only one fault can occur at a time. It doesn't mean that if it is stuck at 0 then it is grounded.

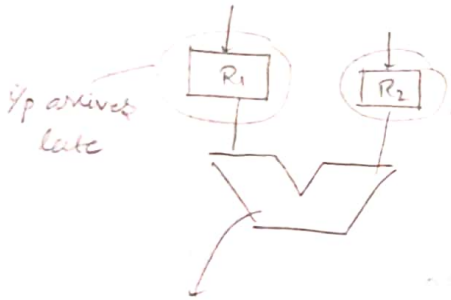
- of course its difficult. Practically there're multiple stuck-at fault
- no. of faults linearly proportional to no. of gates.
- suppose clk freq is 1GHz, there are 4 billion gates so total delay will be 4 secs, which is very small and good

10/09

Toggle problem

- rebinding of variables to registers
- may need additional registers (so that switching occurs at same)

switching should happen at the same time (for P_1 & P_2)



so even if avail. in cycle 1, we defer it to cycle 4

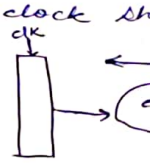
raw ALU will toggle once
⇒ dynamic power reduced

- if switching occurs at a different time then excess power spent unnecessary

- static power already considered during minimizatn of area
- Reduce useless power dissipatn

cause → share the resources

- Ideally clock sh

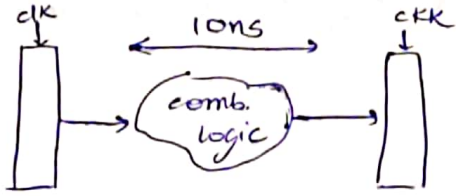


- if we wan
disable f
area so t
- delay of
(hence cl
- thus
than
- In CAD t
(not the
occur)
(combi
used)
- Scalabil
- TESTABILI

Def

Fa

- Ideally clock should be same to all flipflops



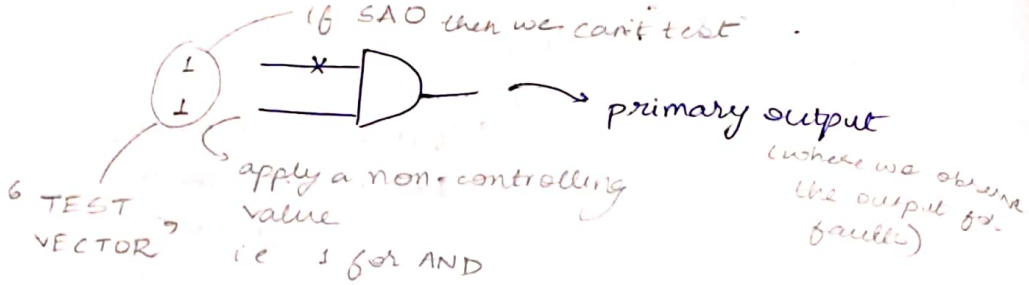
↳ If this clock comes 1ns earlier then effectively we are giving comb. logic 9ns time

- if we want to suppress clock, we don't disable for one FF but for an entire area so that skew (delay) doesn't occur
- delay of gate varies upto 30% (hence clock gating leads to skew)
  - thus we mostly directly use the clock than gating it
- In CAD try to separate out domain (not the most optimized but less mistakes occur)
  (combinational feedback not generally used)
- Scalability requires uniformity

- TESTABILITY

Defect
↓ (surfaces as)
Fault
← SSAF model : logic fault
← timing fault

- while testing gate, we toggle each i/p & o/p

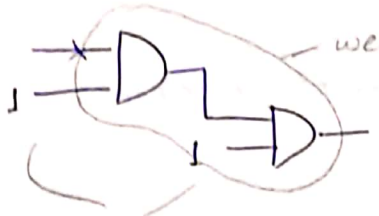


Diagnosis: Analysing and uncovering which exact fault is there
(we usually care if defect is there or not)

- Defect
  - systematic (occurs in all chips)
  - random

i) we first identify which defect occurs
ii) then we do a fault diagnosis
iii) Failure Mode Analysis (FMA): peel off chip layer by layer

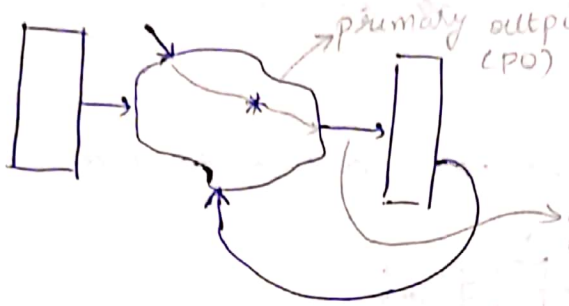
- Timing Defect: chip fails as we increase the frequency of the clock
  - → can be used for lower speed applications (at a lower price)
  - → need to find the frequency at which it works
  - → since each testing costs something we need to ~~test~~ lower the freq at optimum intervals
  - → can't lower too much each revenue would be lower than product cost



we can devise an algorithm to test combinational ckt

Keep adding non-controlling value

Sequential Ckt



but the i/p may not take through the path such that we reach PO)

Time frame Expansion

→ we pass the output from FF again to the CL
⇒ another set of test vector required
⇒ while CL has 1 set of Test vector, SL may have many sets before we reach primary output

Sequencial depth
Maximum no. of clks it passes through for any pair of i/p & o/p
→ want to minimize this

SA1 → no test vector available

(Redundant Fault)
→ so we abort testing

- want logest path to be small so that testing takes lesser time
(i.e. minimize sequential depth)
→

12/09

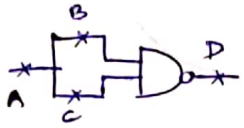
- For testing fault coverage 99% stuck-at full check needs to be there

1% are redundant faults

$$\left(\text{Fault } \cancel{\text{Efficiency}} \text{ coverage}\right) = \frac{\#\text{ detectable faults}}{\#\text{ faults}}$$

- every net can have 2 faults

⇒



No. of stuck-at faults possible } = 2×4 = 8

No. of non-testable faults (Redundant faults) } = 2 ($SA1_A$, $SA1_B$)

↳ don't cause any error

⇒ Faults coverage = $\frac{6}{8} \times 100\%$

$$\text{Fault Efficiency} = \frac{(\#\text{ detectable faults})}{*/(\#\ (\#.\text{faults}) - (\#\text{ redundant faults})}$$

$$= \frac{6}{8-2} \times 100$$

$$= 100\%$$

- Difference between FE and FC tells us the amount of redundancy we have in our circuit
- Sometimes, redundancy are intentional (like in the above case we could've used NOT directly - but we wanted uniformity)

- Redundancy = no test vector available to test the fault

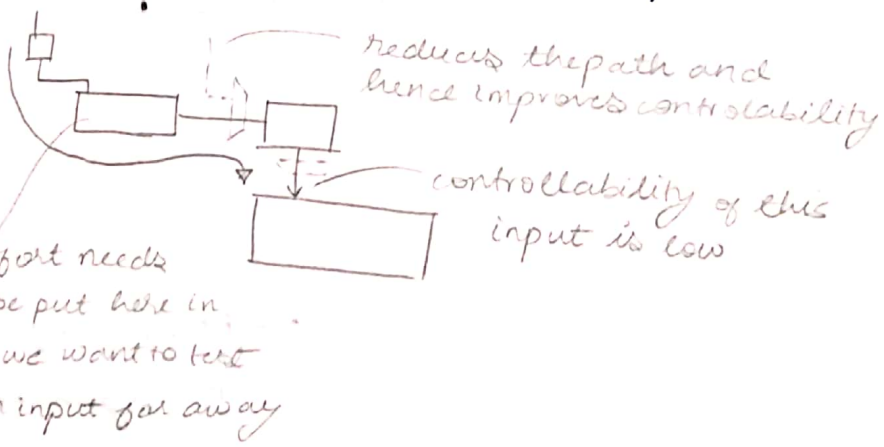
- generating test vector for redundant faults may take a lot of time so somewhere one needs to draw the line and simply acknowledge that the fault is redundant (although they may not be)
  - → but how to decide this limit
  - problem becomes severe for sequential logic
    (because if we have n inputs, we applied 2^n combinations, however as we unroll, no. of inputs becomes 2n
    ⇒ 2^{2n} combinations
    ∴ need to control unrolling

- Sequential Depth

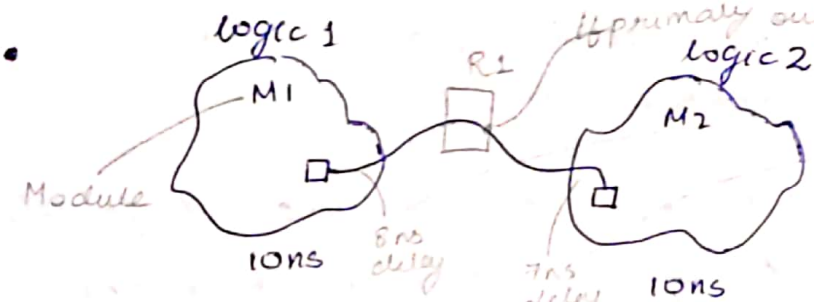
  max. no. of flipflops between a pair of input/output

- Controlability. [Ease of assigning value]

  How easily we can control an input



  - improvement of controllability is also important

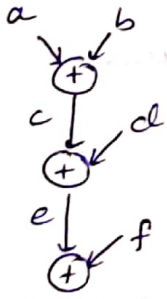


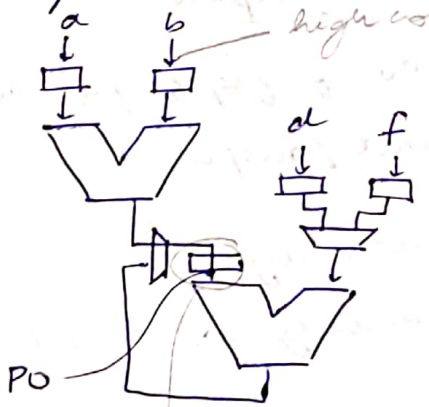
input → FF
input → out
FF → FF
FF → out
} delay

Observability

How easily an output can be observed
- improve observability

- To make test generation easier
  - improve controllability
  - improve observability
  - reduce seq. depth





bad
good controllability
but good observability

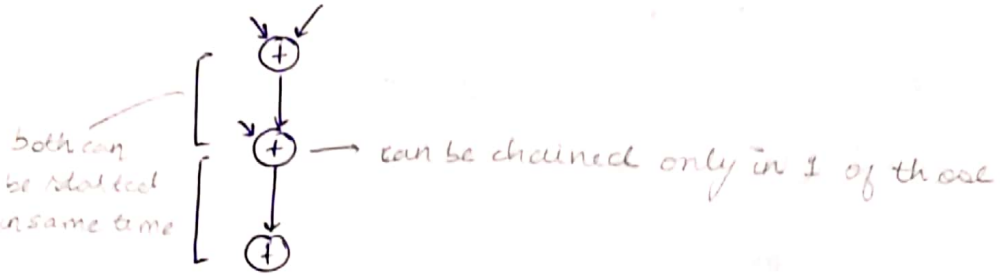
⇒ need to change register binding

→ Re-binding variables to registers

- sequential depth can be optimised during re-scheduling and re-binding
(improve slack)

23/09

- Chaining (ILP in chaining)



→ uniqueness remains
→ dependancy

(only successive operation can be chained)

⇒ $C_{ij} = 0$ if they don't have edge inbetween

⇒ multiply earlier expression for dependency with C_{ij}. However makes ~~it~~ non-linear

- Testability - Security

1) controllable (register should have easy access from primary input)

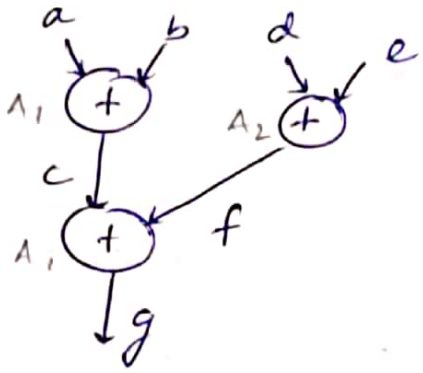
2) observable (register should have easy path to primary output)
↳ how many register it is passing through before coming at PO

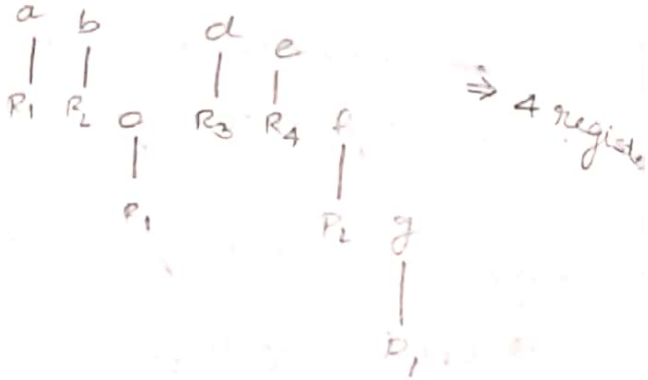
3) Sequential Depth (reduce SD as far as possible
→ defines the no. of registers to pass through

⇒ all the 3 can be determined by
6 Register Allocation'
↳ need to allocate relatively

give different control, observe, SD

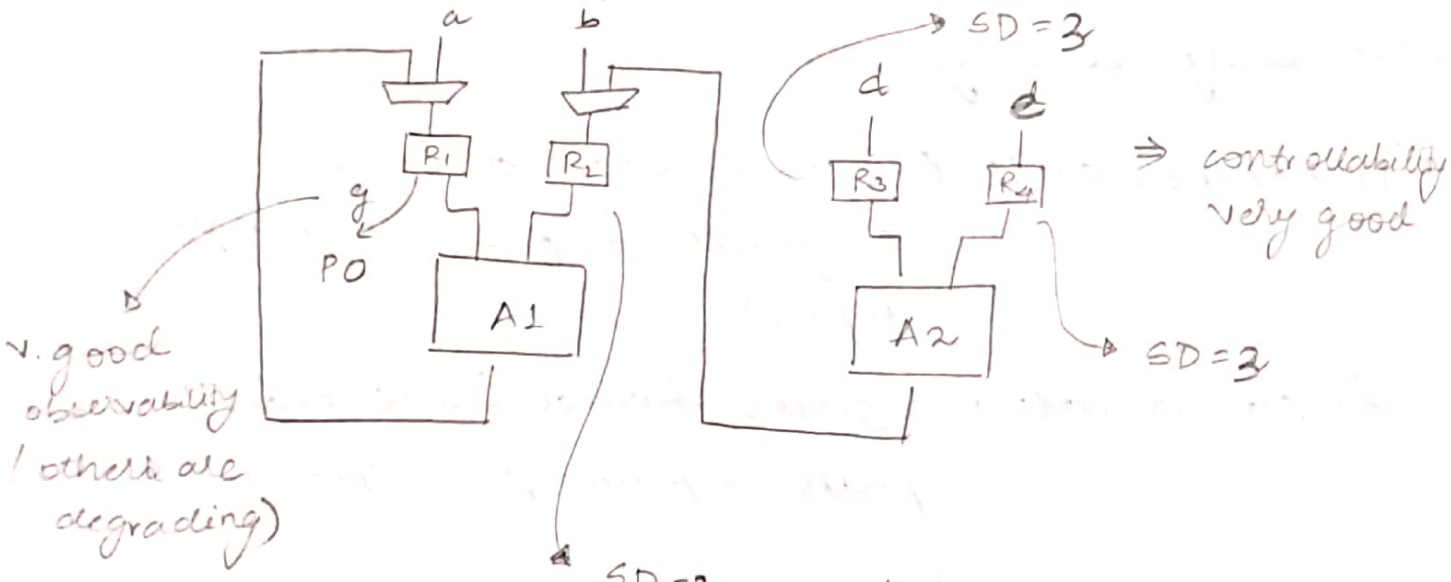
Ex:





⇒ Left Edge ~~Assignment~~ algorithm

or Right Edge Algorithm

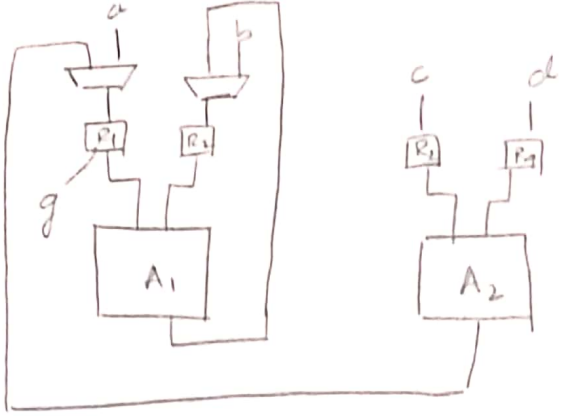


LEFT EDGE ALGORITM

d : SD = 3
e : SD = 3
b : SD = 2
a : SD = 1

- Right Edge Algorithm

$P_1 = \{a, f, g\}$
$P_2 = \{b, c\}$
$P_3 = \{d\}$
$P_4 = \{e\}$

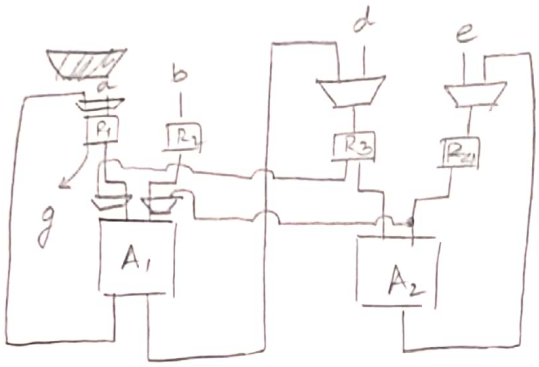


$R_1 = \{a\}\{g\}$

$R_2 = \{b\}$

$R_3 = \{d, c\}$

$R_4 = \{e f\}$



Observability:

$d = 2$ } improved (SP also improves 111 cy)

$e = 2$

$b = 2$