Instead of BTB, we use Stack.
This is called Return address Predictor.
By using stack.

Advance ILP.

Simple Pipelining.

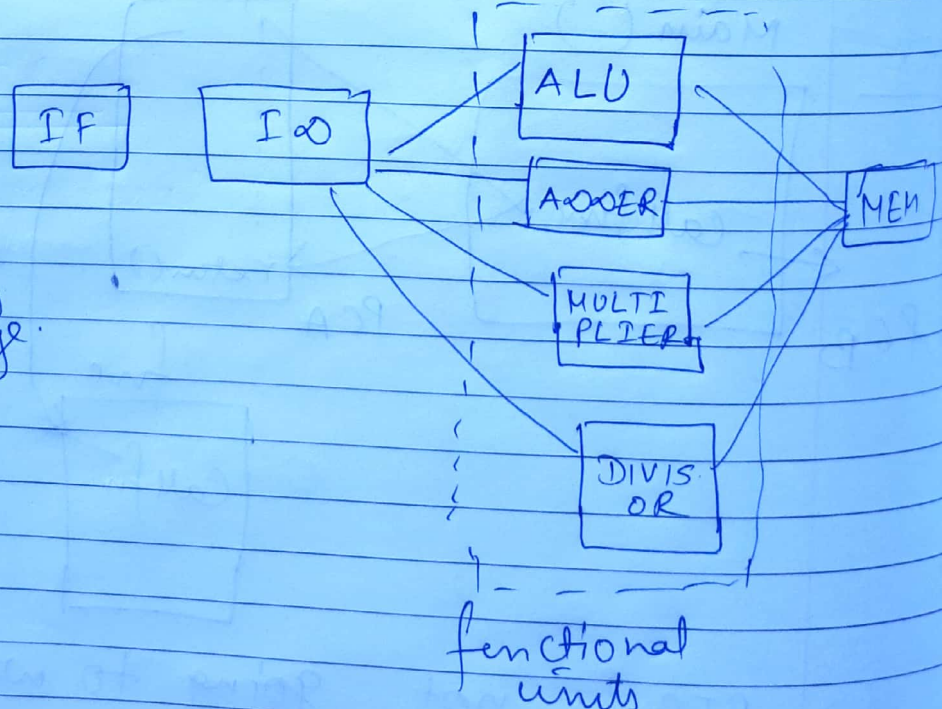
1cc	1cc	1cc	1cc	1cc
IF	ID	EX	MEM	WB

Ex- ADD F0 F1 F2 = 5cc

ADD F0 F1 F2 = 5cc

Ex. ADD.D F0 F1 F2 → 4cc
MULT.D F0 F1 F2 → 10cc
DIV.D F0 F1 F2 → 40cc

Pipelining.



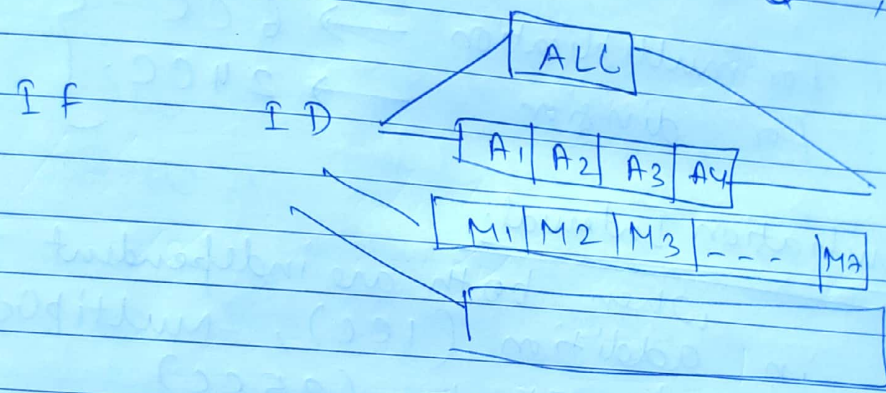
multicycle pipelining
execution stage.

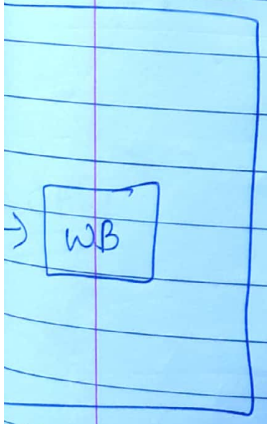
Scanned by CamScanner

Add: 4 CC
Mult: 10 CC
DIV: 40 CC

Ex: Div. ∞ F0 F1 F2 = 44 CC
Add. ∞ F3 F4 F5 = 9 CC
Mult. ∞ F6 F7 F8 = 16 CC

Q. Adder: 4 CC, Multi → 7 CC, DIV: 25 CC, ALU = 1 CC





Latency → of two instructions of same type first one is producing the result and second one is consuming the result, The no. of clock cycle waiting by the second instruction is latency.

The gap in terms of CC b/w the two instruction are of same type.

Scanned by CamScanner

ADD.∞ F0 → F1 F2

ADD.∞ F3 F0 F4

1	2	3	4	5	6	7
IF	ID	A1	A2	A3	A4	MEM
	IF	ID	–	–	–	A1

latency of 3 cc.

for multiplication → 6 CC
for division → 24 CC.

Initiation Interval:

when both are independent like in addition (1 CC), multiplication (1 CC
But in division (25 CC)
↳ As it is complex.

Hazards.

- Structural
- Data
- Control

Scanned by CamScanner

MULT.∞

IF I ∞ M1 M2 M3 M4 M5 M6 M7 MW BW

ADD.∞

II ∞ A1 A2 A3 MW

9 10 11 12

A3 A4 MEM WB

Solⁿ is Extra buffer/Register

Data:

→ RAW (Read after write)

→ WAR

→ WAW

44cc.

DIV.∞ F0 F1 F2

RAW: ADD.∞ F3 F0 F4

9cc

WAR:

DIV.D F0 F1 F2

ADD.∞ F1 F3 F4

WAW:

DIV.D F0 F1 F2

ADD.∞ F0 F3 F4

⋮

ADD.∞ Fx F0 F5

Scanned by CamScanner

Techniques to ∅Hs and CHs

→ Software techniques / compiler techniq
→ H/w (Hardware techniques) Static s

compiler techniques.

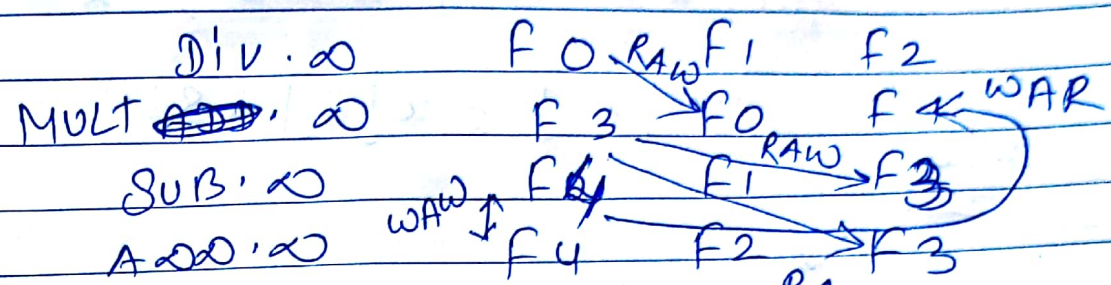
→ Register renaming
→ loop unrolling } Data Hazards.
→ Software pipelining
→ trace scheduling } Control Harads
→ Predicate instructions
→ Multiple Issue processor

H/w Techniques / dynamic Scheduling.

→ ~~dynamic~~ score boarding
→ Tomasulo's Algo → Reorder Bu
→ Tomasulo's Algo + ROB
→ Tomasulo's Alg + ROB + LSQ
↓
(Load store Queue)

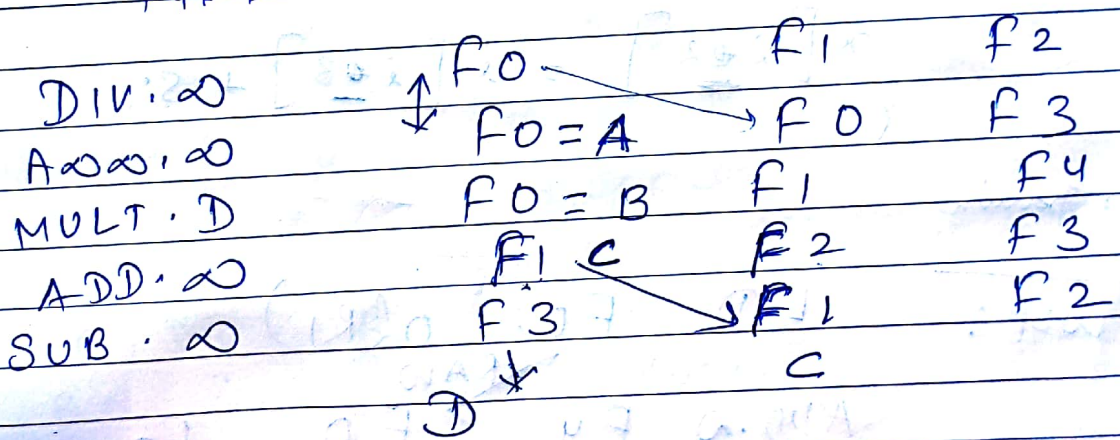
Scanned by CamScanner

Register Renaming.



We have a pool of registers {A, B, C, ∞, ---.Z}
By using this, RAW, WAR, WAW can be removed.

F4 : A. Renamed 3rd row
F4 : B 4th row



Scanned by CamScanner

Page No.:

Loop Un Rolling.

```
for(i=999; i>0; i--)
    x[i] = x[i] + s;
```

1000 times.

⇓

```
for(i=1000; i>0; i=i-4)
{ x[i] = x[i] + s;
  x[i-1] = x[i-1] + s
  x[i-2] = x[i-2] + s;
  x[i-3] = i[i-3] + s;
}
```

250 times.

Assembly Programming.

```
loop: LD   F0, 0(R1)
      ADD.D F4, F0, F2
      Store F4, 0(R1)
      DADDUI R1, R1, #-8
      BNE    R1, R2, loop.
```

RAW: LD F0 → ADD.D F4

RAW: ADD.D F4 → Store F4

RAW: DADDUI R1 → BNE R1

DADDUI ↙ double Addition Unsigned immediate.

Scanned by CamScanner

Unrolling 4 Times:

① LD F0 0(R1)
ADD.∞ F4 F0 F2
SD F4 0(R1)

② LD F0 -8(R1)
ADD.∞ F4 F0, F2
SD F4, -8(R1)

③ LD F0 -16(R1)
ADD.∞ F4 F0, F2
SD F4 -16(R1)

④ LD F0 -24(R1)
ADD.∞ F4 F0, F2
SD F4 -24(R1)

DADDUI R1 R1 #-32
BNE2 R1 R2 loop.

The latency b/w LD and ADD is 1cc and b/w ADD.∞ and SD is 2cc.

Scanned by CamScanner

```
        LD    F0,  0(R1)
ADDO    F4, F0, F2
SO      F4   0(R1)
LO      F6  -8(R1)
AOOO    F12  F6  F2
SD      F12  -8(R1)
LD      F8   -16(R1)
AOOO    F14  F8  F2
SD      F14  -16(R1)
LD      F10  -24(R1)
AOO.O   F16  F10, F2
SD      F16, -24(R1)

~~OOO~~ OAOOUI R1 R1 ≠ -32
        BNE2 R1 R2 loop.
```

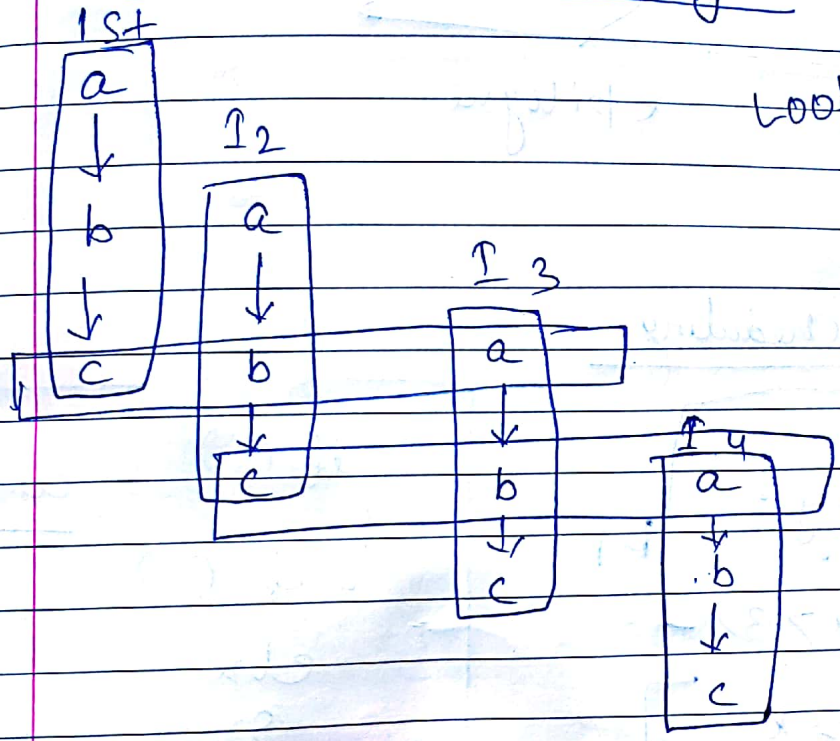
```
a   LD F0, 0(R1)
b   LD F6, -8(R1)
c   LO F8, -16(R1)
a   LD F10, -24(R1)
b   AOO.O F4, F0, F2
c   AOO.O F12, F6, F2
a   AOO. O F14, F8, F2
b   AOO. O F16, F10, F2
c
a   SD . . F4 (0(R1)
b   SO F12 . -8(R1)
c   SO F14 -16(R1)
a   OAOOOI R1 R1 ≠32
b   SO F16 +8(R1)
c
OO  BNE 2 R1R2 loop;
```

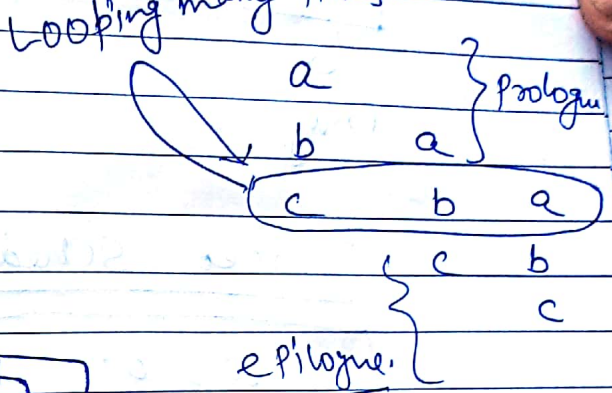
Scanned by CamScanner

Page No.:

Drawbacks of loop unrolling:

- Software Pipelining.

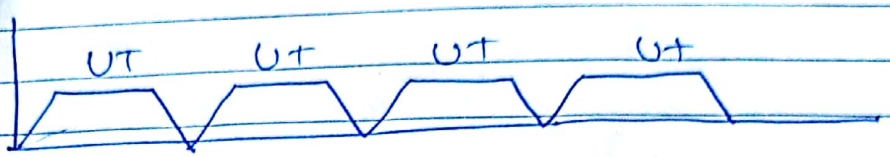




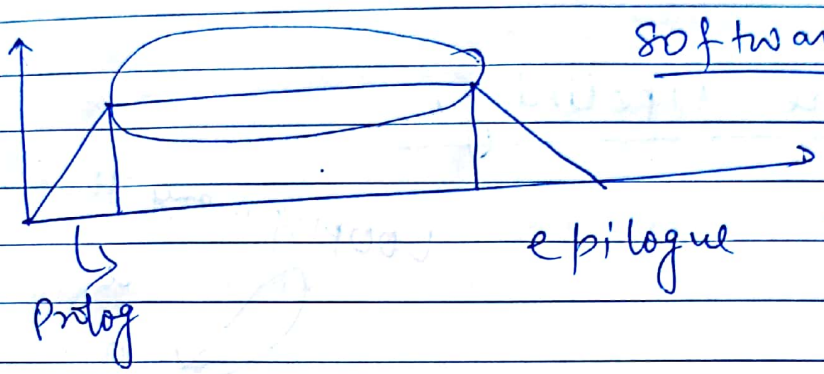

```
        LD    F0   0(R1)
        ADD.∞ F4   F0  F2
        LD    F0   0(R1)
        SD    F4   0(R1)
loop:   ADD.∞ F4   F0  F2
        LD    F0,  0(R1)
        DA∞∞UI
        BNEZ                 R1, R2 ≠ ?.
        SD    F4   0(R1)     R1 R2 loop
        A∞∞.∞ F4   00/F2
        SD F4 0(R1)
```

Scanned by CamScanner

loop unrolling.



Software Pipelining.

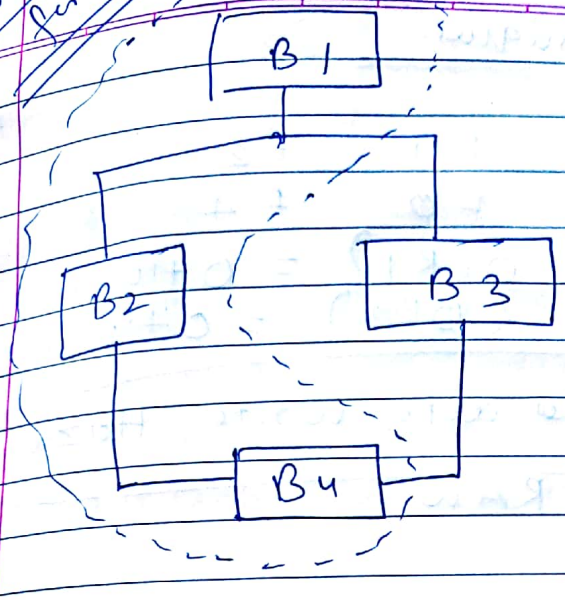


Traces Scheduling.

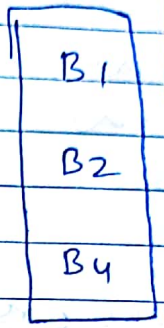
```
w = 0;
x = x+y;
y = 0;
if (x>3)        B1
{ y = x
  x++;          B2
}
else
  y = z
  z++;          B3
}
w = x+z ;       B4
```

```
if ( )
{ if ( )
else
  {
    if ( )
  { else
  }
```

Scanned by CamScanner



=>




```
w = 0;
x = x + y;
y = 0;
y = x
x++
w = x + z;
if (l < m)
```

Tail computation

Recovery code

→ if → else.

if 50% is true and false then this technique is inefficient in tail computation

Predicate instructions.

```
if (R1 == 0)
    R2 = R2 + R4
else
    R6 = R3 + R5
    R4 = R2 + R3
```

=>

```
R7 = !.R1 | predicated on R7
R8 = R2
R2 = R2 + R5
R6 = R3 + R5
R4 = R8 + R3
```

→ Predicated on R7

→ Predicated on R1

Scanned by CamScanner

Addressing Mode

Page No.:

H/w Techniques.

MULT∞	F0	F1	F2
ADD.∞	F3	F0	F4
SD	F4	0(R1)	= 0+1024 = 10
LD	F3	(0R2)	= 0+1024 = 10

Memory read after write Hazard

RAW

Memory WAR

LD F4 0(R1)

SD F0 0R(R2)

memory WAW

SD F4 (0R1)

S∞ F0 0(R2)

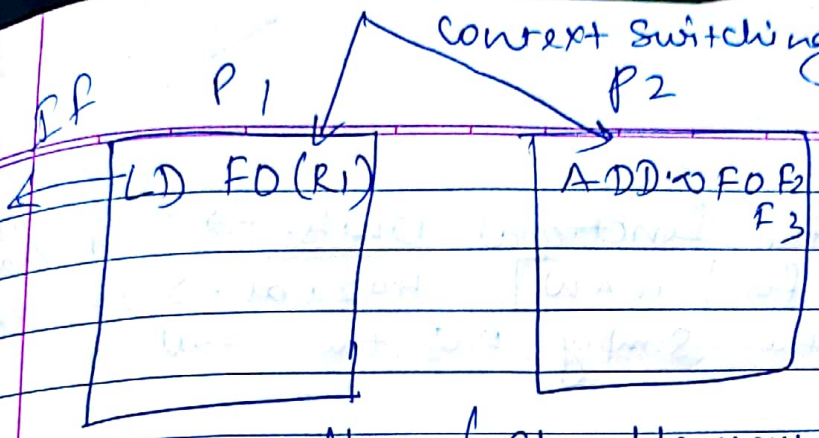
Div.∞ f0 f1 f2 — /0 Exception is detected.

ADD.∞ F1 F3 F4

When any exception is happened after that no instruction should be completed due to error.

0 chan to 0.2

Scanned by CamScanner



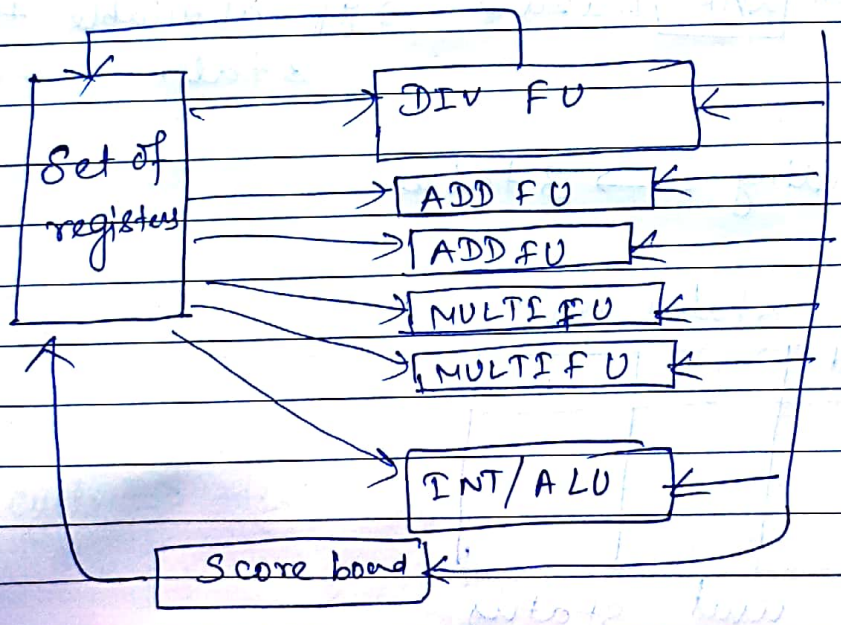
Hyper threading (Simultaneous Multi threading

11/03/2018

H/W Techniques.

1) Score Boarding
- out of order execution

FU → functional unit



Simple pipelining IF ID EX MEM WB

Score boarding IF IS (Issue) RD Read operand EX WR Write results

Scanned by CamScanner

Issue stage:

→ check for functional units. → If unavailable do stall

→ check for WAW Hazards. → If it exists then simply put the stall.

Read operands.

→ Both the operands are available.

Execute: perform execution using functional units

Write results → writing into the destination register.

check WAR Hazard → If available then stall.

Scoreboarding → 3 tables.

Instruction status unit

IS	Rd	Ex	WR

Functional unit status

BUSY	OP	Fi	Fj	Fk	Rj	Rk
	DIV Multi Sub					

Fj, Fk → inputs

Fi → destination register

Qj

Qk → Who will give Fj, Fk

Rj, Rk → Whether operand is available

Register status.

Scanned by CamScanner

Instruction Status Unit

	IS	Rd	Ex	WR
LD F6 34(R2)	1	2	3	4
LD F2 45(R3)	5	6	7	8
MULT D F0 F2 F4	6	9	19	20
SUB ∞ F8 F6 F2	7	9	11	12
DIV D F10 F0 F6	8	21	61	62
ADD ∞ F6 F8 F2	13	14	16	22

functional unit.

	Busy	OP	Fi	Fj	Fk	Qj	Qk	Rj	Rk
INT	~~NO~~ yes	LD1	F6	34	R2	—	—	—	—
MULTI	NO								
ADD	NO								
DIV ∞	NO								

Register Unit

F0	F2	F4	F6	F8	F10

clock cycle => (0)

Adder = 2CC
Div ∞ = 40CC
MULT = 10CC
INT = 1CC

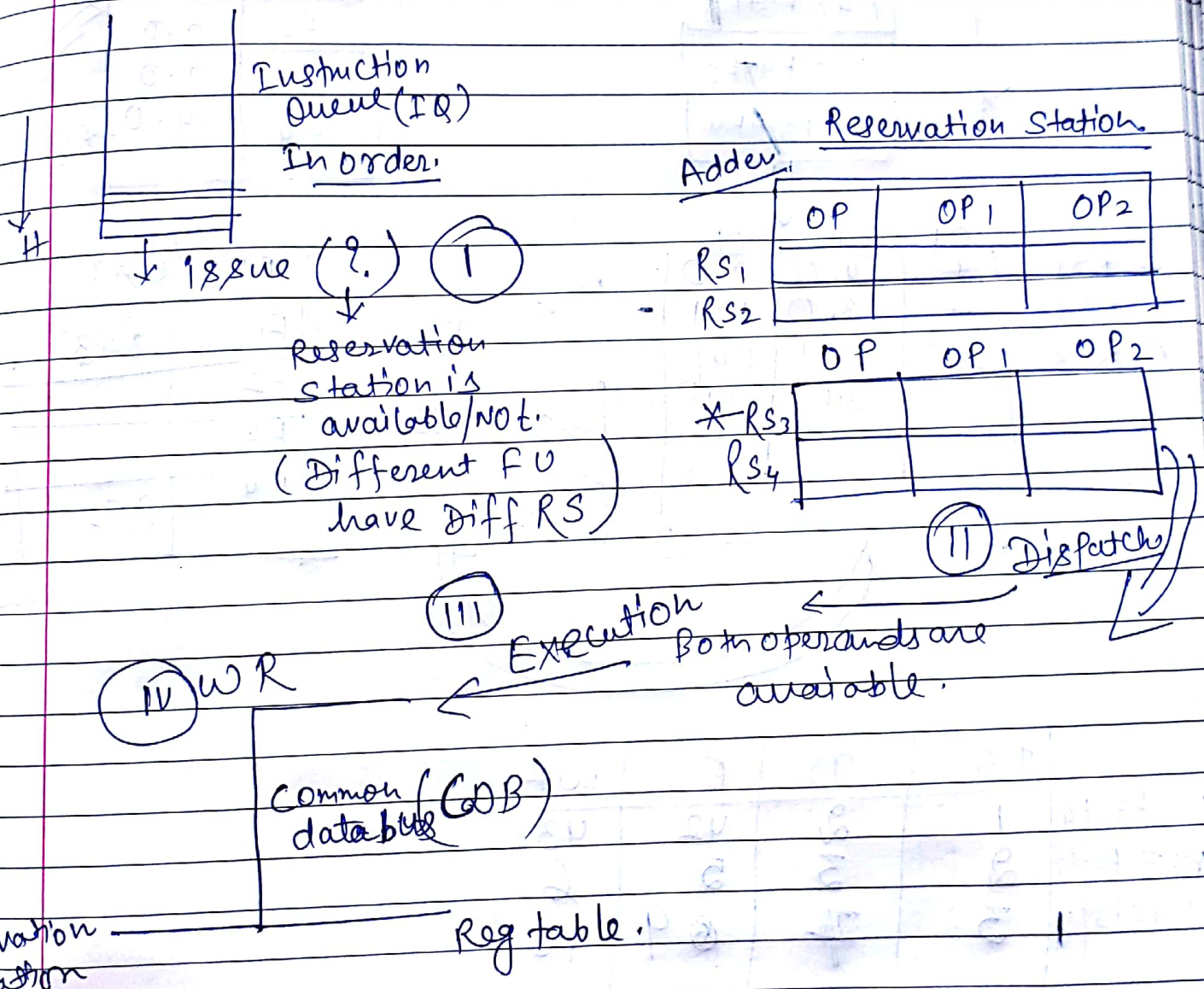
Drawbacks of Scoreboarding

- There is no forwarding.
- It is going to solve any hazards only put the stalls.

Scanned by CamScanner

Tomasulo's Algo.

It Solves WAR & WAW Hazard.



Scanned by CamScanner

WAW

Div → 40CC
Add → 2CC

Page No.:

F1 = F3 + F4

F4 = F3 + F1

F1 = F4 + F2

F1 = F3 / F4

Register table

F1	1.0
F2	2.0
F3	3.0
F4	4.0

→ RS1
→ RS2

	Op	Op1	Op2
+ RS1	+	4.0	2.0
+ RS2	+	3.0	RS1
/ RS3	/	F3 3.0	F4 4.0

RRT/RAT

F1	RS3
F2	
F3	
F4	RS2

	IS	D	E	WR
F1 = F3/F4	1	2	42	43
F1 = F4+F2	2	3	5	6
F4 = F3+F1	3	7	9	10

Scanned by CamScanner

WAR problem

~~F4 = F1 - F2~~ F3 = F4 + F3
~~F1 = F3 + F4~~ F4 = F2 + F3
~~F1 = F4 / F2~~ F1 = F4 / F2

RS1

	OP	OP1	OP2
+RS1	+	F2 (2:0)	F3 (3:0)
+RS2	+	RS1	3:0
/RS3	—	F4 (4:0)	F2 (2:0)

RS1

	IS	∞	E	WR
F1=F4/F2	1	2	4 2	4 3
F4=F2+F3	2	3	5	6
F3=F4+F3	3	7	9	10

Reg table.

F1	1:0 /RS3
F2	2:0
F3	3:0
F4	4:0 /RS1

RRT

F1 RS3
F2
F3
F4

Scanned by CamScanner

Page No.:

Book

(I) $f_1 = f_2 + f_3$

(II) $f_4 = f_1 - f_2$

(III) $f_1 = f_2 / f_3$

(IV) $f_2 = f_4 + f_1$

$f_1 = 1.0$
$f_2 = 2.0$
$f_3 = 3.0$
$f_4 = 4.0$

Add
div
W

	OP	OP_1	OP_2
+RS1	+	2.0	3.0
+RS2	−	RS1/5.0	2.0
/RS3	/	2.0	3.0

Reg.
f_1 1.0/RS
f_2 2.0
f_3 3.0
f_4 4.0/R

	IS	ID	EX	WR
$f_1 = f_2 + f_3$	1	2	~~5~~ 6	7
$f_4 = f_1 - f_2$	2	7	~~10~~ 11	12
$f_1 = f_2 / f_3$	~~8~~	9		
$f_2 = f_4 + f_1$	~~13~~	14	18	19

1	2	6	7
2	8	12	13
3	4	29	30
8	31	35	36

f_2 RS
f_2 1
f_3
f_4 RS

Scanned by CamScanner

$F1 = F2 + F3$	1	2	6	7
$F4 = F1 - F2$	2	8	12	13
$F1 = F2 / Fx$	20			
$F2 = F4 + F0$	8	14	18	19

	IS	D	EX	WR	C
$F1 = F2 + f3$	1	2	6	7	8
$F4 = F1 - F2$	2	8	12	13	14
$F1 = F2 / F3$	3	4	29	30	31
$F2 = F4 + F0$	9	10	14	15	32

	F0	1.0
	F1	1.0/ROB0/ROB2
	F2	2.0
	F3	0.0
	F4	3.0

	OP	OP1	OP2	ROB entry	
+RS1	+	2.0	0.0	ROB0	ROB1
+RS2	—	ROB0	2.0	ROB1	ROB2
/RS3	/	2.0	0.0	ROB2	ROB3

	T	D	V	C/NC
ROB0	+	F1	2.0	H
ROB1	—	F4		
ROB2	/	F1	—	

RRT	
F0	
F1	ROB0/ROB2
F2	
F3	
F4	ROB1

Scanned by CamScanner

Page No.:

$f_1 = f_3 + f_4$
$f_1 = f_4 / f_2$
$f_4 = f_3 - f_2$

$f_1 = 1.0$
$f_2 = 0.0$
$f_3 = 2.0$
$f_4 = 3.0$

	OP	OP_1	OP_2	ROB entry
RS 1	+	2.0	3.0	ROB 0
RS 2				
RS 3	/			

	IS	D	EX	WR	C
$f_1 = f_3 + f_4$	1	2	6	7	8
$f_1 = f_4 / f_2$	2	3	28	29	30
$f_4 = f_3 - f_2$	3	4	8	9	10

Scanned by CamScanner

Tomasulo' Alg + ROB

=> Branch hazards

=> Speculation.

```
(1)  MULTD   F2   F0   F1
     BNE     F2,  0,   Label
     ADD     F4   F3   F1
Label
     ADD     F3   F4   F2
```

Mult → 10CC

	IS	D	Ex	WR	C
1	1	2	12	13	14
2	2				
3	3	4			

	OP	OP1	OP2	ROB entry
RS1	*	0.0	1.0	ROB0
RS2	+	3.0	1.0	ROB2
RS3				

F0	0.0
F1	1.0
F2	2.0 ROB0
F3	3.0
F4	4.0

RRT

	T	D	V	C/NT
ROB0	*	F2	—	H
ROB1	BNE	F2	—	—
ROB2	+	F4	—	—
ROB3				

F0	
F1	
F2	ROB0
F3	
F4	

Scanned by CamScanner

Page No.:

			IS	D	Ex	WR	C
LD	R_1	0 (R_1)	1	2	3		
SD	R_2	0 (R_3)	2	3	4		
LD	R_4	0 (R_4)	3	4	5		
S∞	R5	0 (R0)					
LD	R5	0 (R_1)					

T	D	V	C/NC
L	R_1	—	H
S	R_2	—	

L/S	EA	V	C/NC
L	104		H
S	204	10	store to load forwarding
L	204	10	

After Calculation of 1st Instruction is 104.

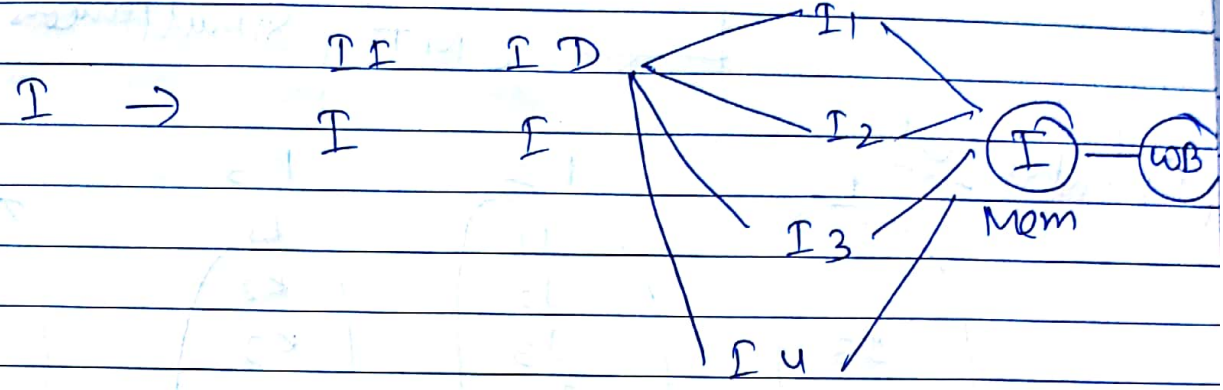
Scanned by CamScanner

V.V.I.

Multi issue Processor

VLIW

very long instruction word.



	IF	ID	EX	M	WB
I1	1				
I2	1				
I3	1				
I4	1				

5 issue Processor/

A 2 VIIW Can issue two load/store operations and two floating point operations and one decremental/branch instruction in one clock cycle.

LD1	SD1	1 CC	LD1	LD2	—	— —
LD2	SD2	2 CC	LD3	LD4	—	— —
---	---	3 CC	LD5	LD6	ADD1	ADD2 —
LD7	SD7	4 CC	LD7	—	ADD3	ADD4 —
ADD1	DADDUI	5 CC	—	—	ADD5	ADD6 —
ADD2	BNE2	6 CC	SD1	SD2	ADD7	— —
---		7 CC	SD3	SD4	—	DADDUI
		8 CC	SD5	SD6	—	— BNE2
ADD7		9 CC	SD7	—	—	— —
		10 CC				

Scanned by CamScanner

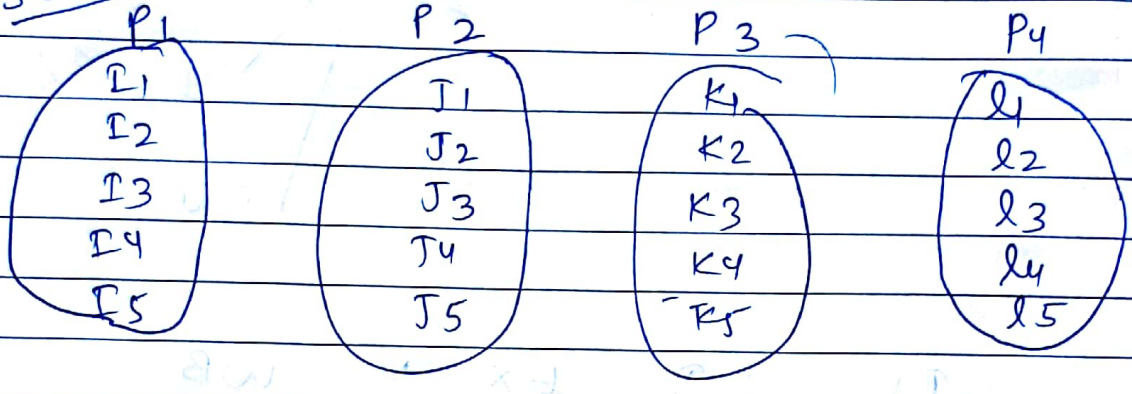
Page No.:

Simultaneous Multithreading.

4 Issue Processor (4 instructions in 1CC)

- Super scalar
- fine grained
- SMT (Simultaneous multithreading)

(1) Super Scalar



1CC	I_1	I_2
2CC	I_3	I_4
3CC		
4CC	I_5	I_6

When there is a big delay then move from one Processor to another for fetching instruction.

(2) Fine grained.

1CC	I_1	I_2
2CC	J_1	J_2
3CC	K_2	K_2
4CC	l_1	l_2 l_3 ...

(3) SMT

I_1	I_2	J_1	K_1
I_3	J_2	J_3	J_4

If there is a gap then jump to other processor

Scanned by CamScanner

~~05/03~~ 31/03/2018. PK Jana. Page No.: 1

Data level Parallelism.

Array Processors.

No. of Processors connected in parallel manner.

Single instruction stream multiple data stream.

SIMD computer organization:- (vector architecture)

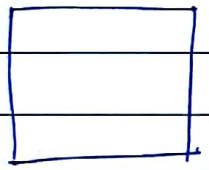
$$C_i = A_i + B_i * C_i$$

$i = 0$ to $n-1$

$A = (A_0, A_1, \ldots A_{n-1})$

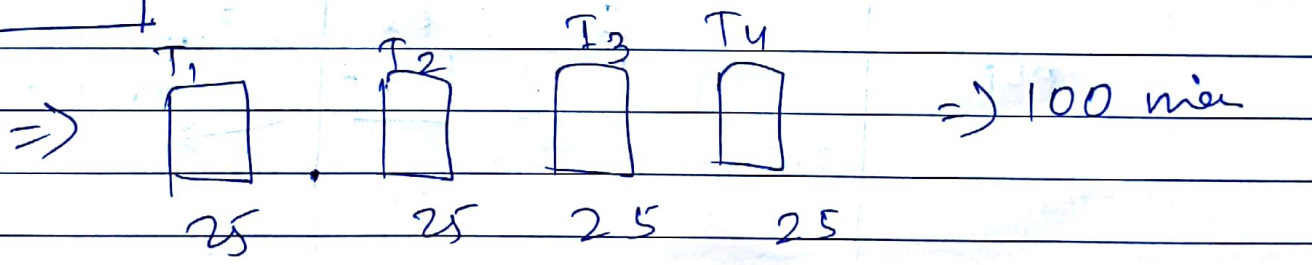
$B = (B_0, B_1, \ldots B_{n-1})$

$C = (C_0, C_1 \ldots C_{n-1})$

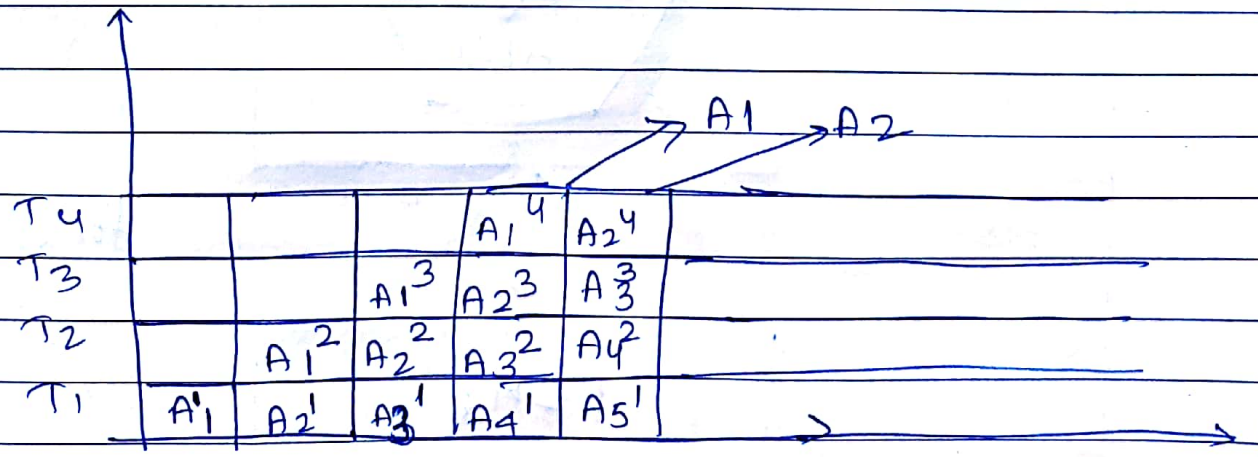


4 options
100 question

=> Total time = 400 min.



=> 100 min



$4 + 99 = 103 \approx 100$

→ This is Pipelining.

Scanned by CamScanner

$C_i = A_i + B_i * C_i$

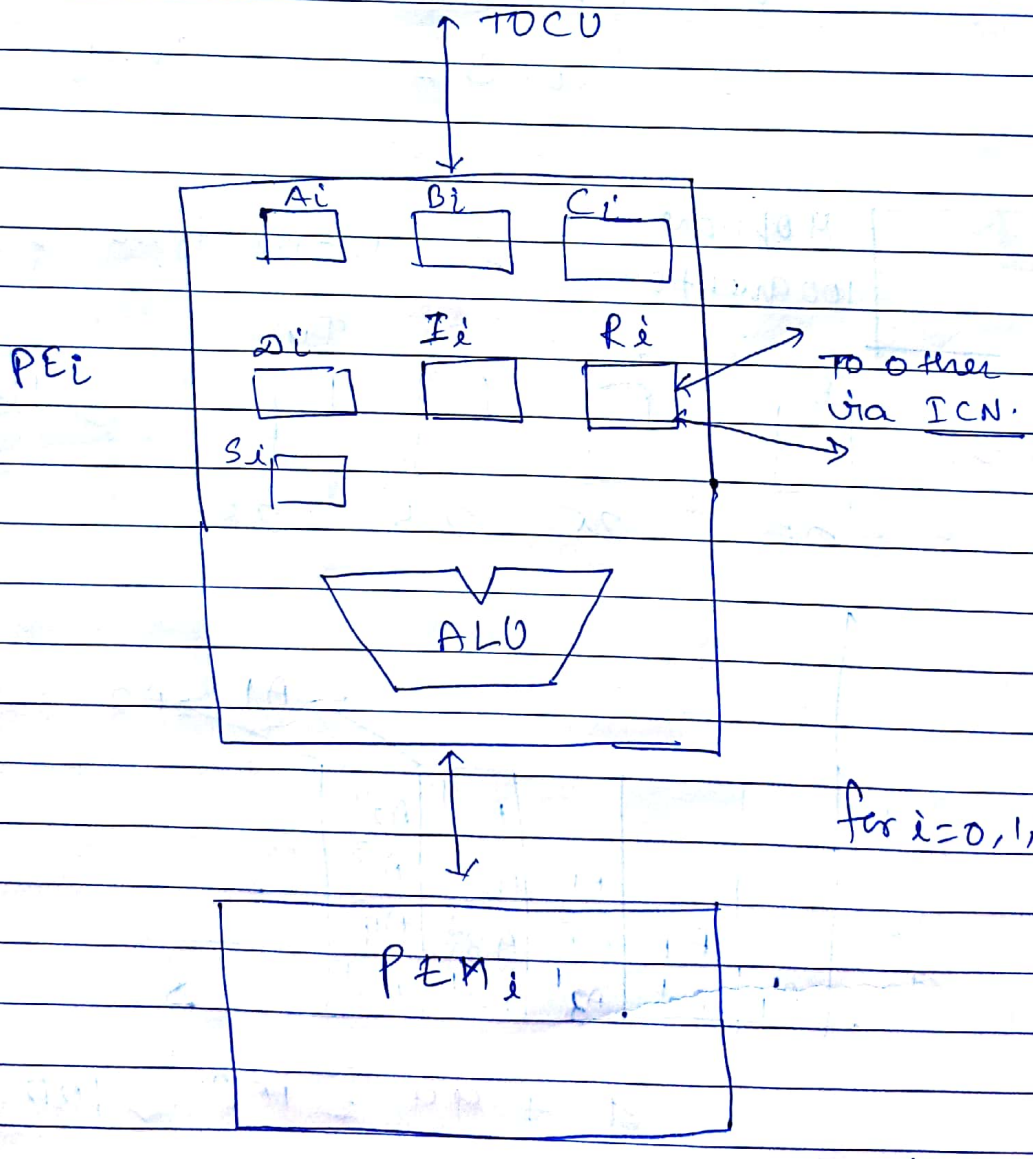
SIMD Computer $C = \langle N, F, I, M \rangle$

N → No. of processing element in the system

F → Set of ~~data routing~~ function

I → the set of machine instruction

M → the set of masking scheme.



for i=0,1,2,...,N-1

Components in a processing Element

Scanned by CamScanner

Illustration of Data Routing and Masking schemes:-

$A = A0, A1, ---, An-1$

we need to compute $S(K) = \sum_{i=0}^{K} Ai$ for $K = 0, 1, 2, --, n-1$

$S(0) = A0$

$S(1) = A0 + A1$

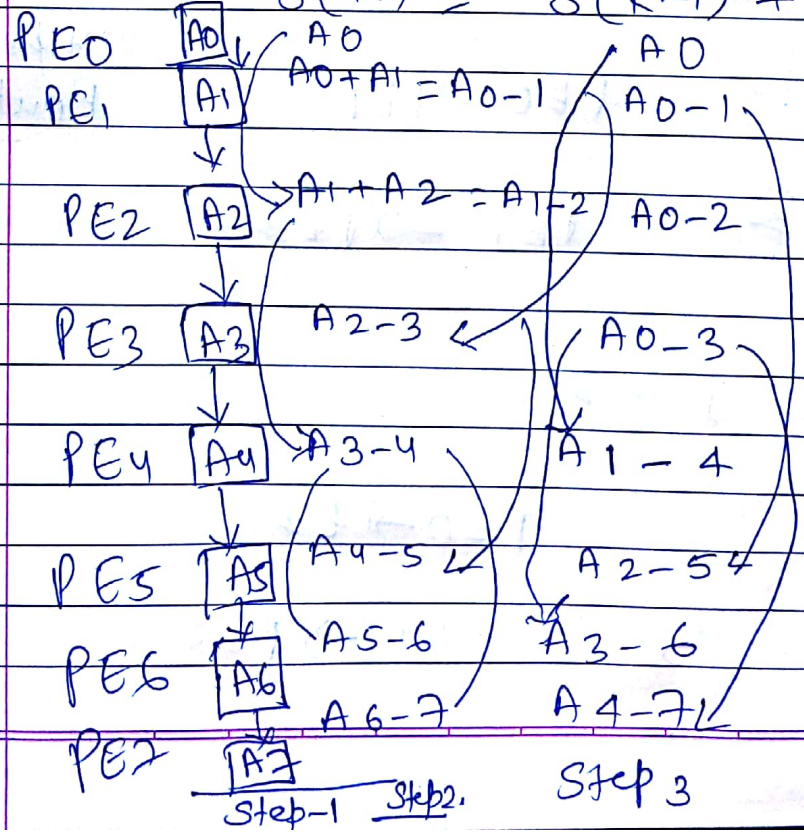
$S(2) = A0 + A1 + A2$

$S(n-1) = A0 + A1 + --- + An-1$

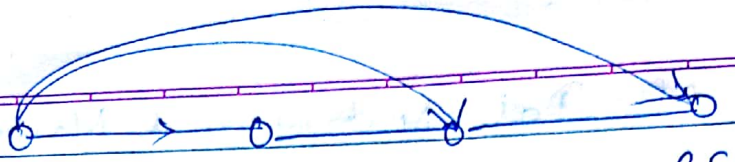
Time $\rightarrow O(n)$ with the help of recursion.

$S(0) = A0$

$S(K) = S(K-1) + AK$ for $K = 0, 1, --, n-1$

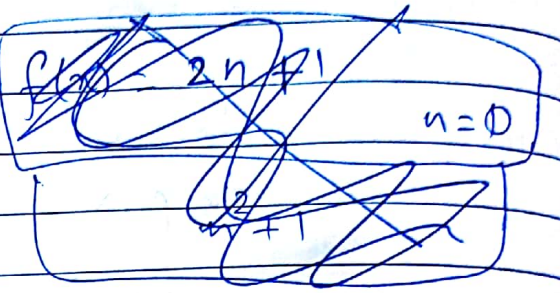


Scanned by CamScanner



Time complexity = $O(\log_2 n)$

PE1 PE3 PE5



$R_i(J) = J + 2^i$ $\quad 0 \le i \le \log(n)-1,$

$0 \le J \le n-1$

for n=8: $0 \le i \le 2$, $0 \le J \le 7$

for i=0 → 0, 1, 2 (3 steps of Routing.)

$R_0(J) = J+1$

$R_0(0) = 1$ PE0 → PE1

$R_0(1) = 2$ PE1 → PE2

for i=1

$R_1(J) = J+2$

$R_1(0) = 2$ PE0 → PE2

$R_1(1) = 3$ PE1 → PE3

Scanned by CamScanner

for i = 2.

$R_2(J) = J + 4$

$R_2(0) = 4 \quad P(E0) \rightarrow PE4$

Masking scheme. Step1 0, 1, 1, 1, 1, 1, ____

0 → when not perform addition.

1 → when addition ~~not~~ performs.

Step2.

Step3 → 0 0 0 0 1 1 1 1

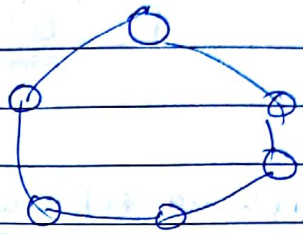
Scanned by CamScanner

Page No.:

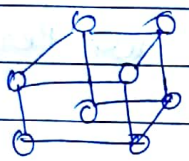
Types of Interconnection network.

Static vs Dynamic networks.

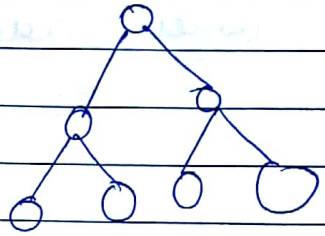
Linear



Ring



cube



Tree

↓

Of vertices of cube is replaced by ~~3 cube~~ 3 circles connected then cube circled.

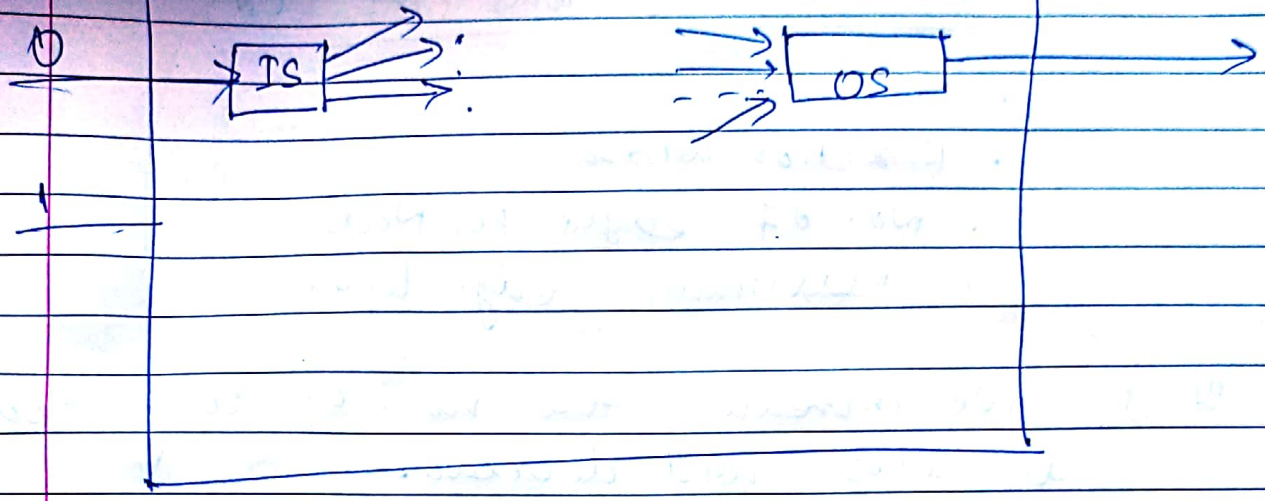
Two classes of dynamic network:—

- Single - Stage
- Multi stage.

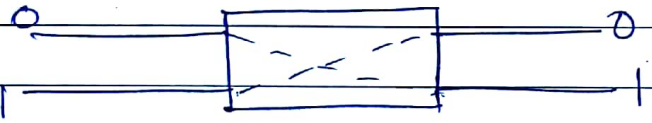
IS → Input selector ↓ 1XD demultiplexer

OS → output selector. DX1 multiplexer

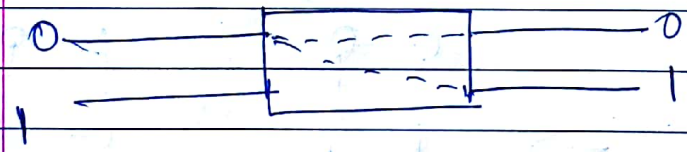
Scanned by CamScanner



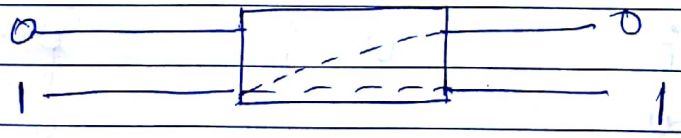
Straight



Exchange

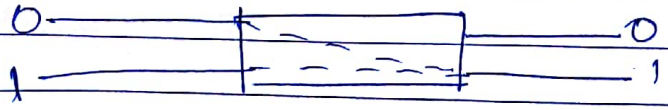


Lower broadcast.



Upper broadcast

Not Permitted.



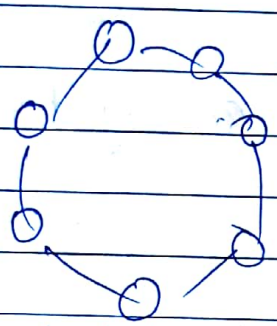
Not Permitted.

Scanned by CamScanner

Evaluation Criteria of ICN

- diameter
- Bisection width
- No. of edges per Node
- Maximum edge length.

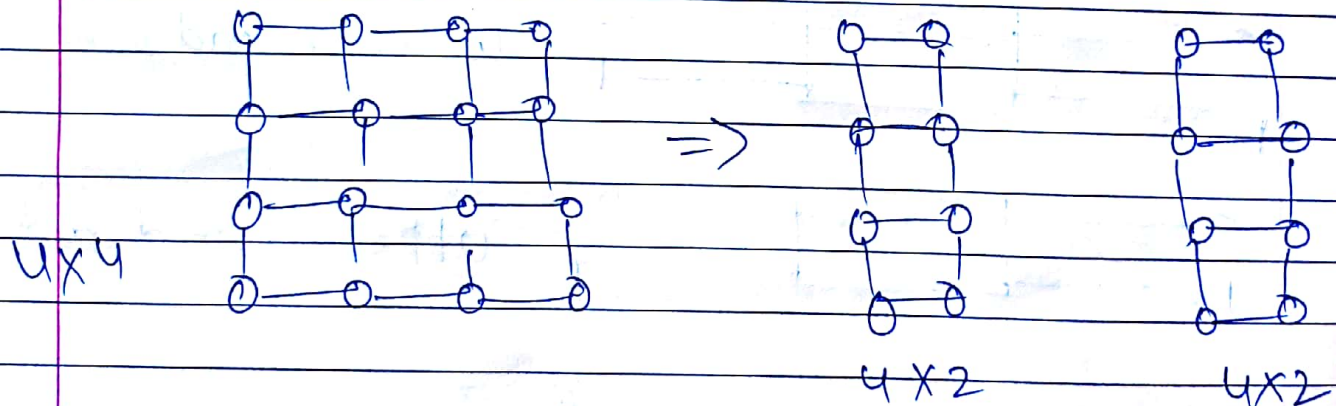
If we increase the no. of connections, diameter will decrease.



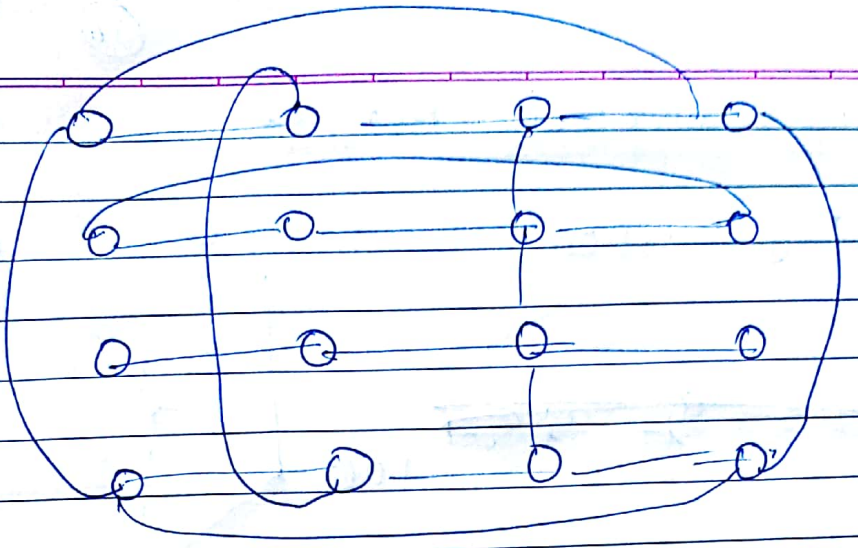
Unidirectional
$n-1$

If it changes to bidirections
$\left(\frac{n-1}{2}\right)$

- Bisection width



Scanned by CamScanner



4 X 4 2D - wrap around mesh

If no. of connections is more to reduce diameter, it will create hazards in the form of heat due to interference, overlapping of networks.

Dynamic networks. → Three classes:—

- Blocking
- Rearrangeable
- Non-blocking

↓

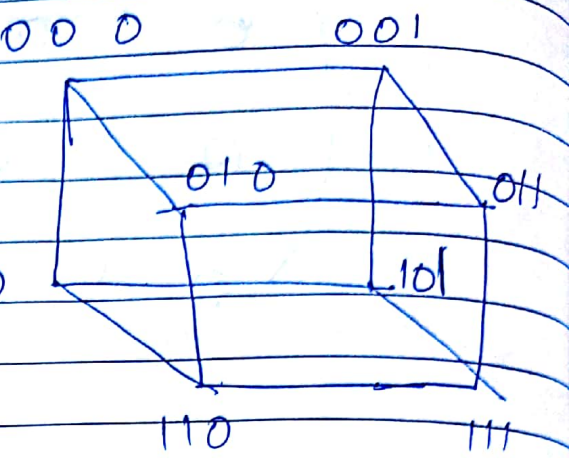
Simultaneous connected b/w source and Receiver.

Scanned by CamScanner

Cube interconnection Network.

Cube routing function

$C_i(a_{n-1} a_{n-2} \cdots a_{i+1} a_i a_{i-1} \cdots a_1 a_0)$



A 3 cube of 8 nodes

$C_i(a_{n-1} a_{n-2} \cdots a_{i+1} a_i a_{i-1} \cdots a_2 a_1 a_0)$

$= a_{n-1} a_{n-2} \cdots a_{i+1} \bar{a}_i a_{i-1} \cdots a_2 a_1 a_0$

$C_0(000) = 001$

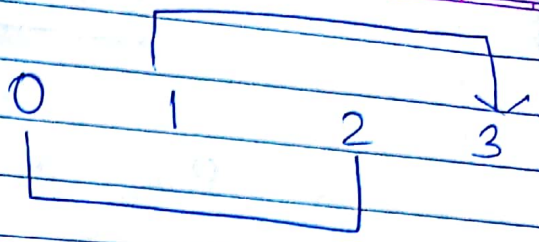
$C_0(001) = 000$

$C_0(101) = 100$

$0 \leftrightarrow 1 \quad 2 \leftrightarrow 3 \quad 4 \leftrightarrow 5 \quad 6 \leftrightarrow 7$ C_0

Scanned by CamScanner

$C_1(000) = 010$: diagonal connectivity



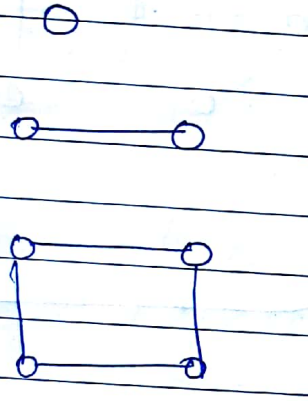
$C_2(000) = 100$

$C_2(111) = 011$

vertical connectivity.

Hypercube.

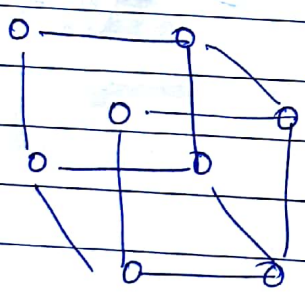
(2^n).



$2^0 = 1$

$2^1 = 2$

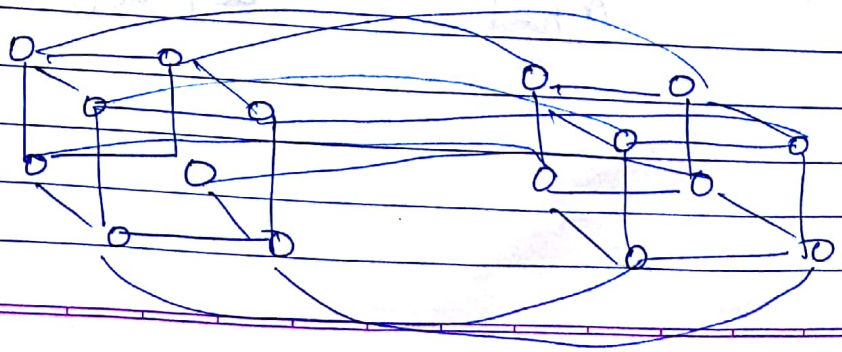
$2^2 = 4$



$2^3 = 8$

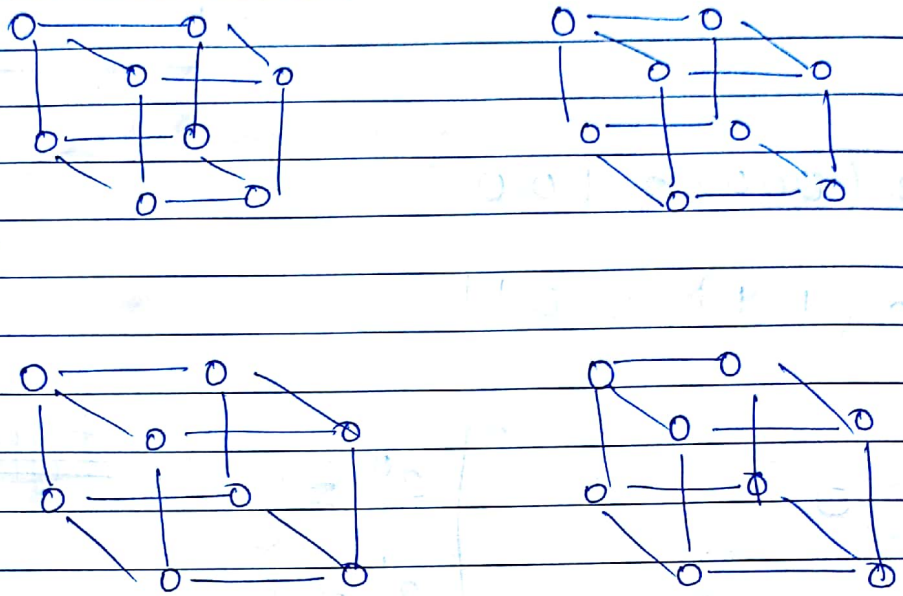
Horizontal connectivity

4D Hyper cube.



Scanned by CamScanner

5D Hyper Cube



Shuffle function

$S(a_{n-1}\, a_{n-2} \dots a_1\, a_0) = a_{n-2} \dots a_1 a_0 a_{n-1}$

$S(000) = 000$

$S(001) = 010$

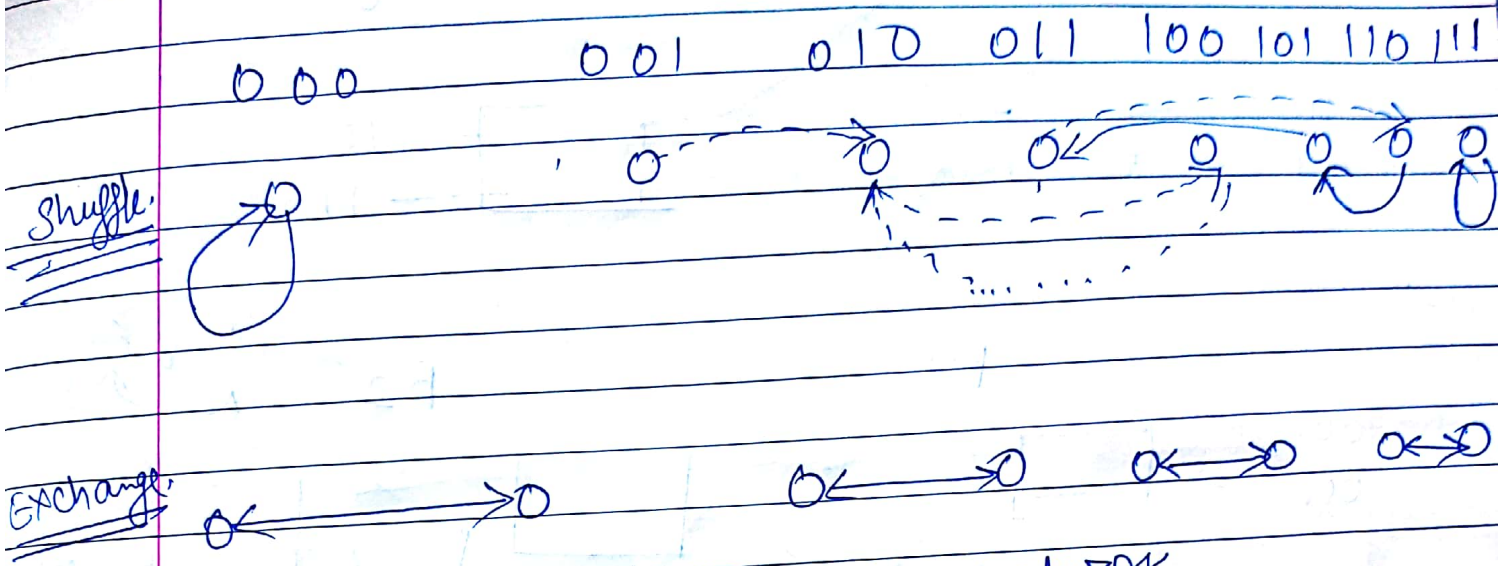
Inverse:

$a_{n-1} \dots a_1\, a_0 = a_{n-1} \dots a_1 a_0$

Scanned by CamScanner

Exchange routing function

$E(A) = CO(A)$



8 node - shuffle exchange network.

Task: Prove that the diameter of an n-processor (n p) Shuffle exchange network is $O(\log n)$.

Butterfly function

$B_K (b_{n-1}\ b_{n-2} \dots b_{K+1}\ b_K\ b_{K-1} \dots b_2 b_1 b_0)$

$= b_{n-1}\ b_{n-2} \dots b_{K+1}\ b_0 b_{K-1} \dots b_2 b_1 b_K$

$K \rightarrow 1$ to $n-1$

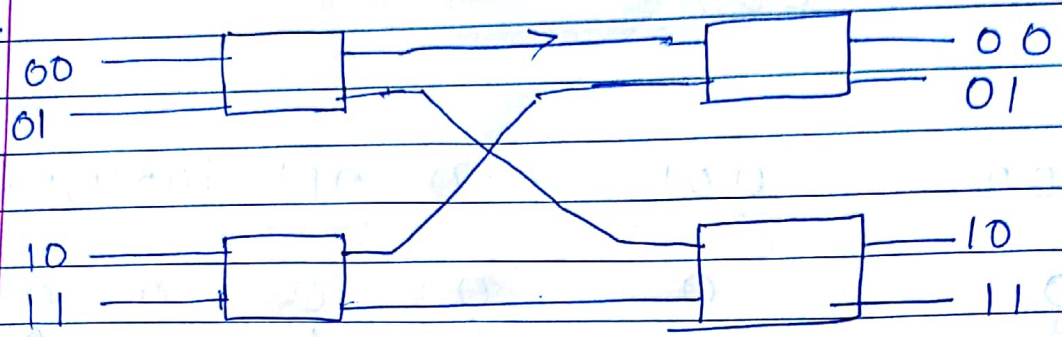
$]0, n-1]$

$0 < K \leq n-1$

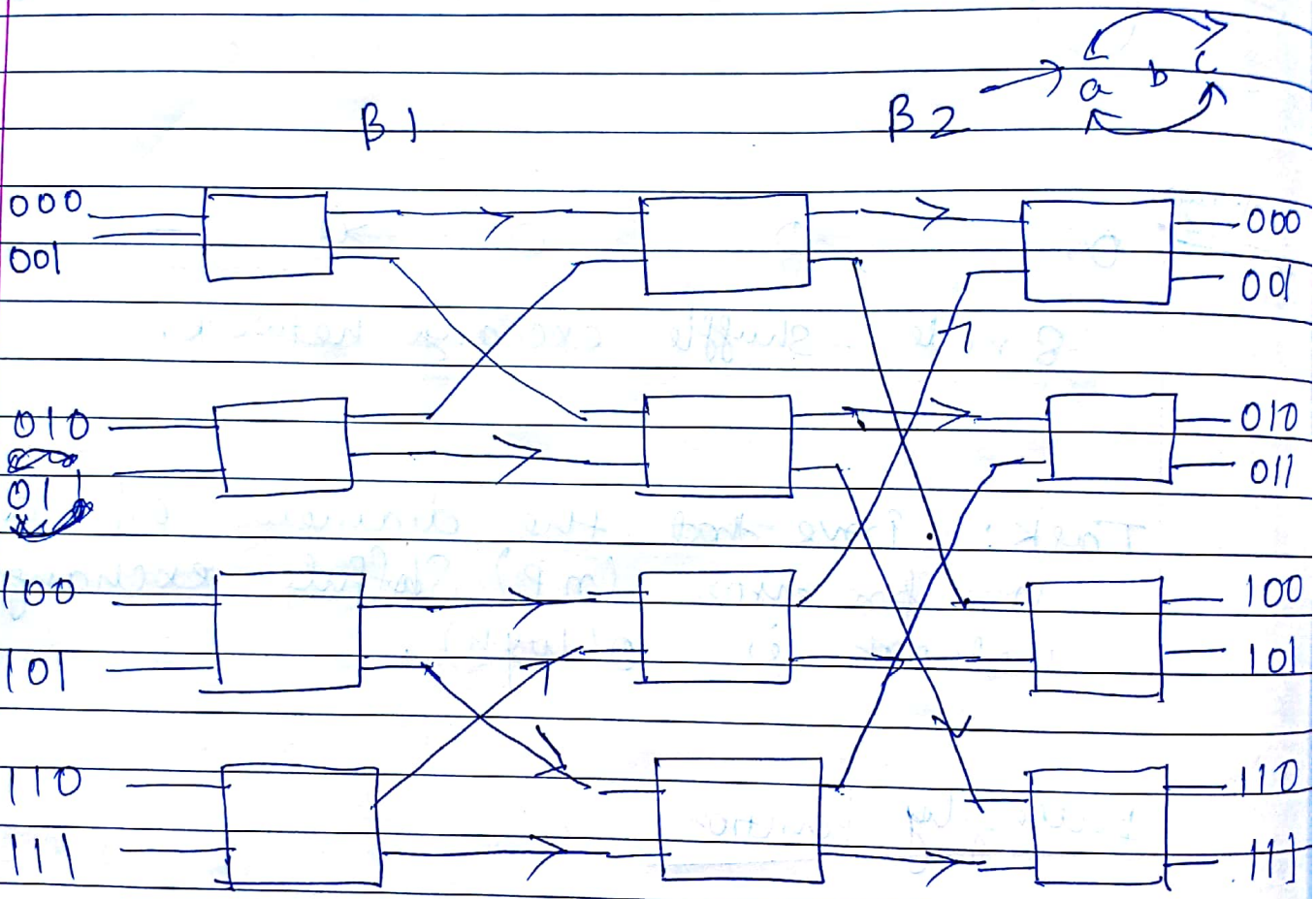
Scanned by CamScanner

Page No.:

ex-1

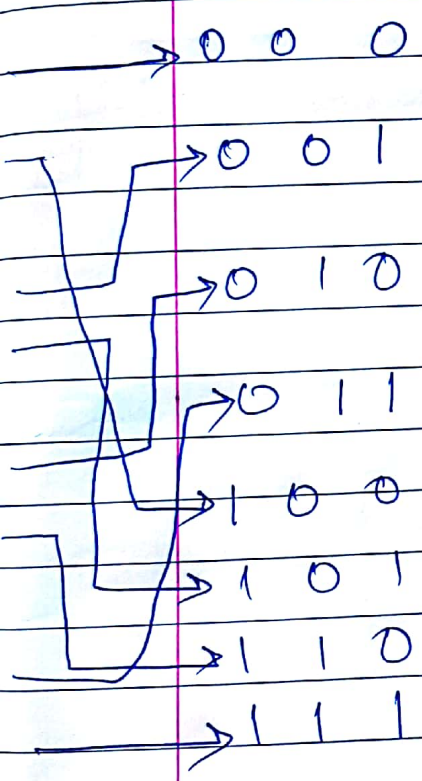


ex-2



Scanned by CamScanner

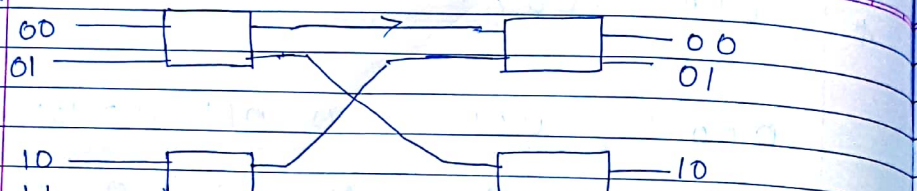
σ^{-1} (single bit right rotation)



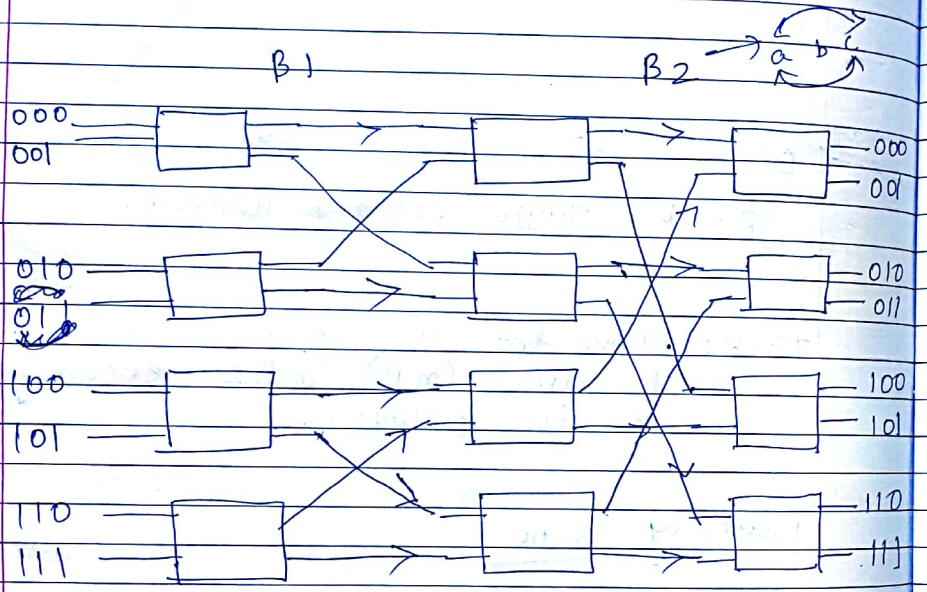
Scanned by CamScanner

Page No.:

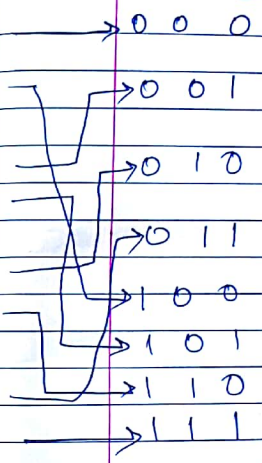
ex-1



ex-2

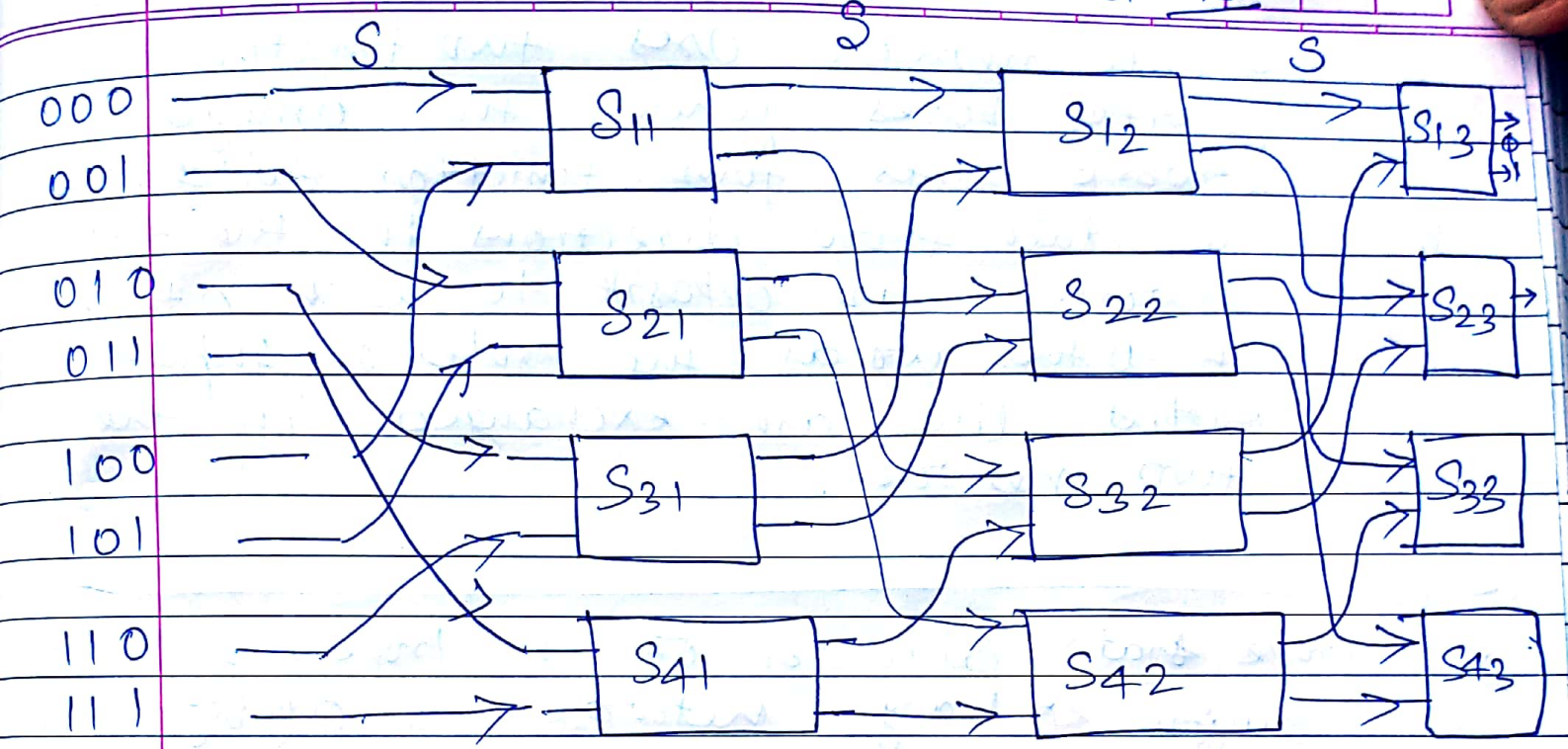


σ^{-1} (single bit right rotation)



Scanned by CamScanner

3 stage 8 x 8 Omega Switching network.



Reposition omega network. (changing positio

$S_{32} \leftrightarrow S_{22}$

Scanned by CamScanner

Page No.:

The cube network uses two function switch boxes whereas the omega network uses four function switch boxes.

The data flow directions in the two networks are opposite to each other. In other words, the roles of input-output lines are exchanged in the two network.

Prove that diameter of n-Processor Shuffle exchange network is $O(\log n)$

Scanned by CamScanner

Page No.:

S	E
000	000
001	001
~~010~~	
010	010
011	011
100	100
101	101
110	110
111	111

for $i = 1$ to $\log_2 n$

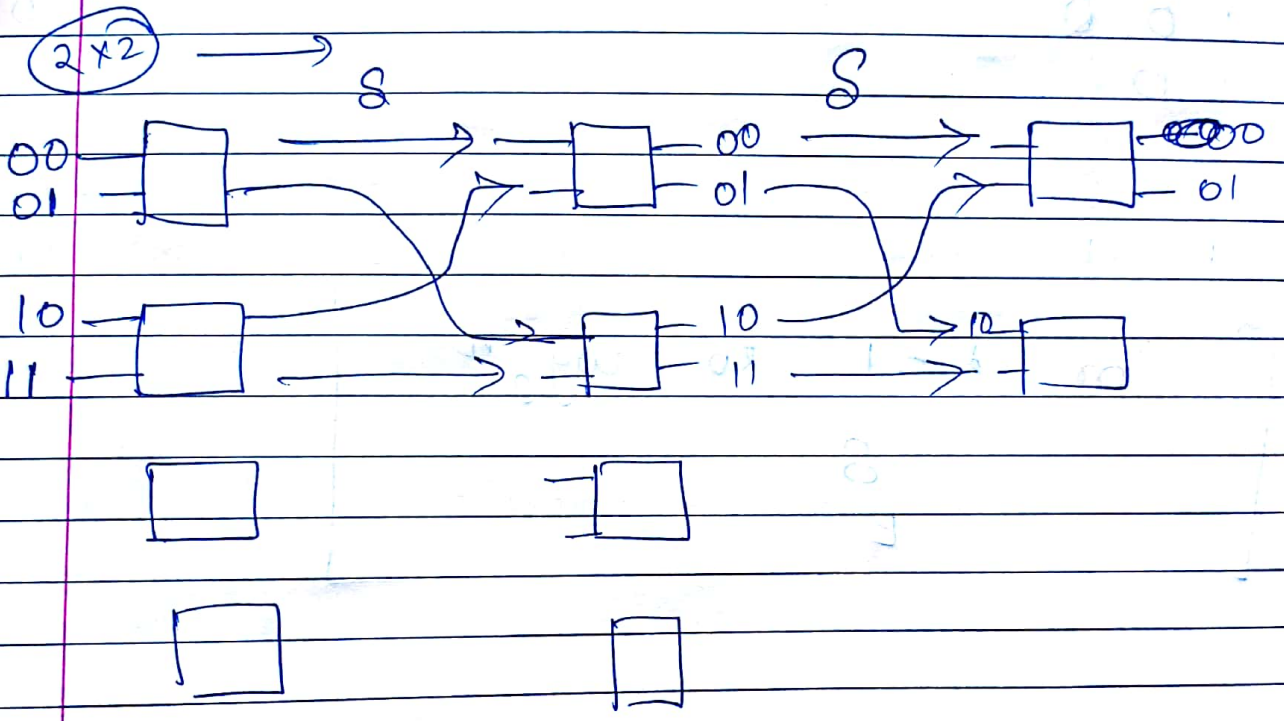
S

E

Scanned by CamScanner

Page No.:

Let us consider a matrix of size $2^m \times 2^m$. To prove the ~~Inverse~~ Transpose of matrix can be obtained by m-perfect shuffle. or m-Shuffle.



Scanned by CamScanner

$A^{-1} = adj$

$$\begin{bmatrix} \phi & 0 \\ 0 & \phi \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

2 × 2.

$$\begin{bmatrix} 1 & 0 & 0 \\ & 0 & \\ & 1 & \\ & & 1 \end{bmatrix}$$

$m = 2$. $\quad 2^2 \times 2^2 = 16$

$$\begin{pmatrix} 0000 & 0001 & 0010 & 0011 \\ 0100 & 0101 & 0110 & 0111 \\ 1000 & 1001 & 1010 & 1011 \\ 1100 & 1110 & 1110 & 1111 \end{pmatrix}$$

~~1000~~

row col.

1001

↓

0011

↓

$\bar{0}\,1\,\bar{1}\,0$

other row

Scanned by CamScanner

Page No.:

$2^m \times 2^m = 2^{2m}$ (2m) bits $= 2m$

i row | jth Colum

$a_{2m-1}\ a_{2m-2} \dots a_m$ | $a_{m-1}\ a_{m-2} \dots a_2 a_1 a_0$

m Shuffle.

$a_{m-1}\ a_{m-2} \dots a_2 a_1 a_0$ $a_{2m-1}\ a_{2m-2} \dots a_m$

jth column ith row.

Scanned by CamScanner

01/04/2018.

Page No.:

Iliac Routing function

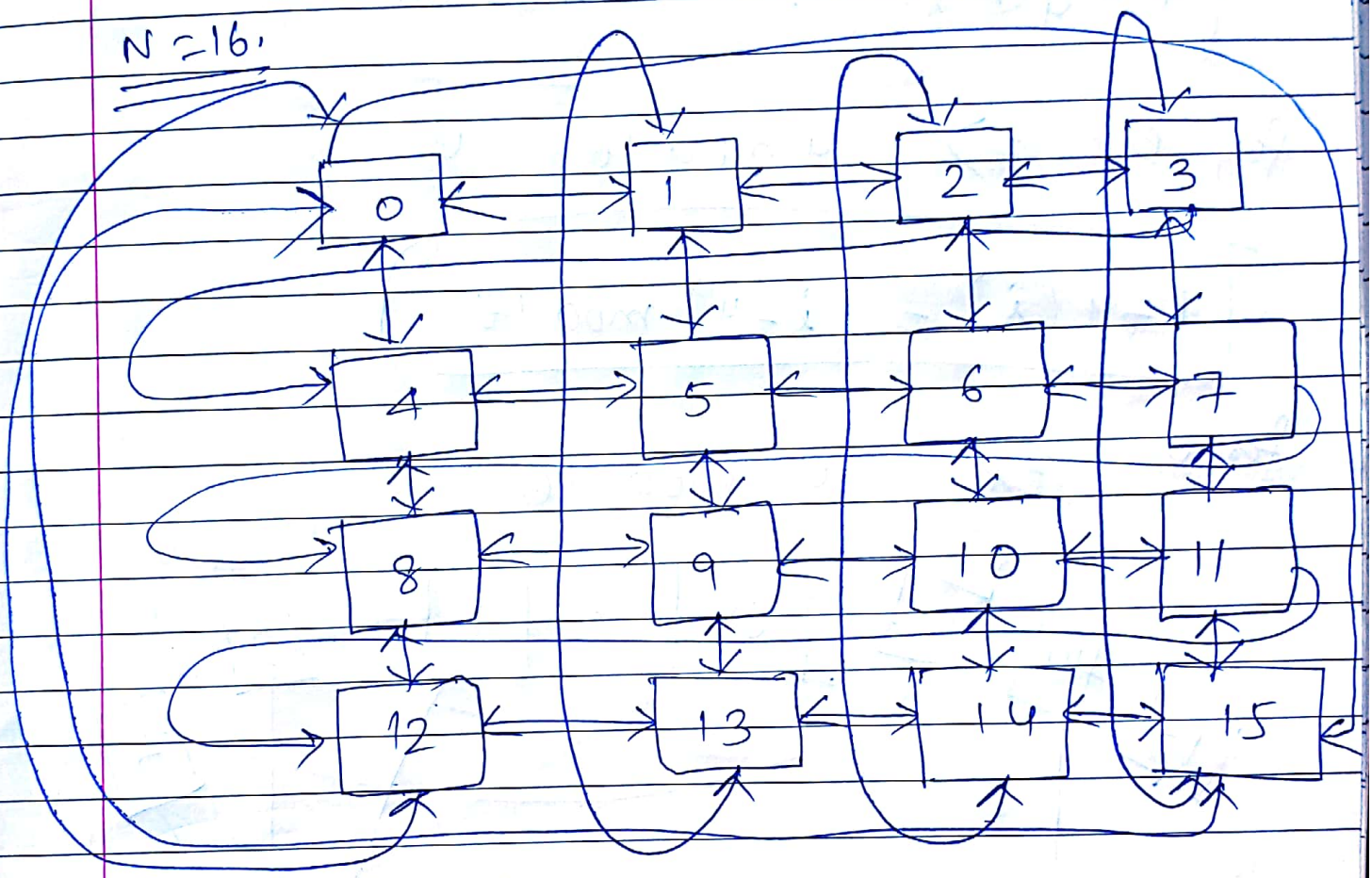
i → address of Processor

$$R_{\pm 1}(i) = (i \pm 1) \bmod N$$

$$R_{\pm r}(i) = (i \pm r) \bmod n$$

where $r = \sqrt{N}$, $0 \le i \le N-1$

N = 16.



$R_{+1}(i) = (i+1) \bmod 16$ $r = 4$

$0 \le i \le 15$

for 0 => 1 mod 16 = 1

for 1 => 2 mod 16 = 2

for 2 => 3 mod 16 = 3

for 3 => 4 mod 16 = 4

Scanned by CamScanner

$R-1(i) = (i-1) \mod 16$

for 0 $\Rightarrow -1 \mod 16 = 15$

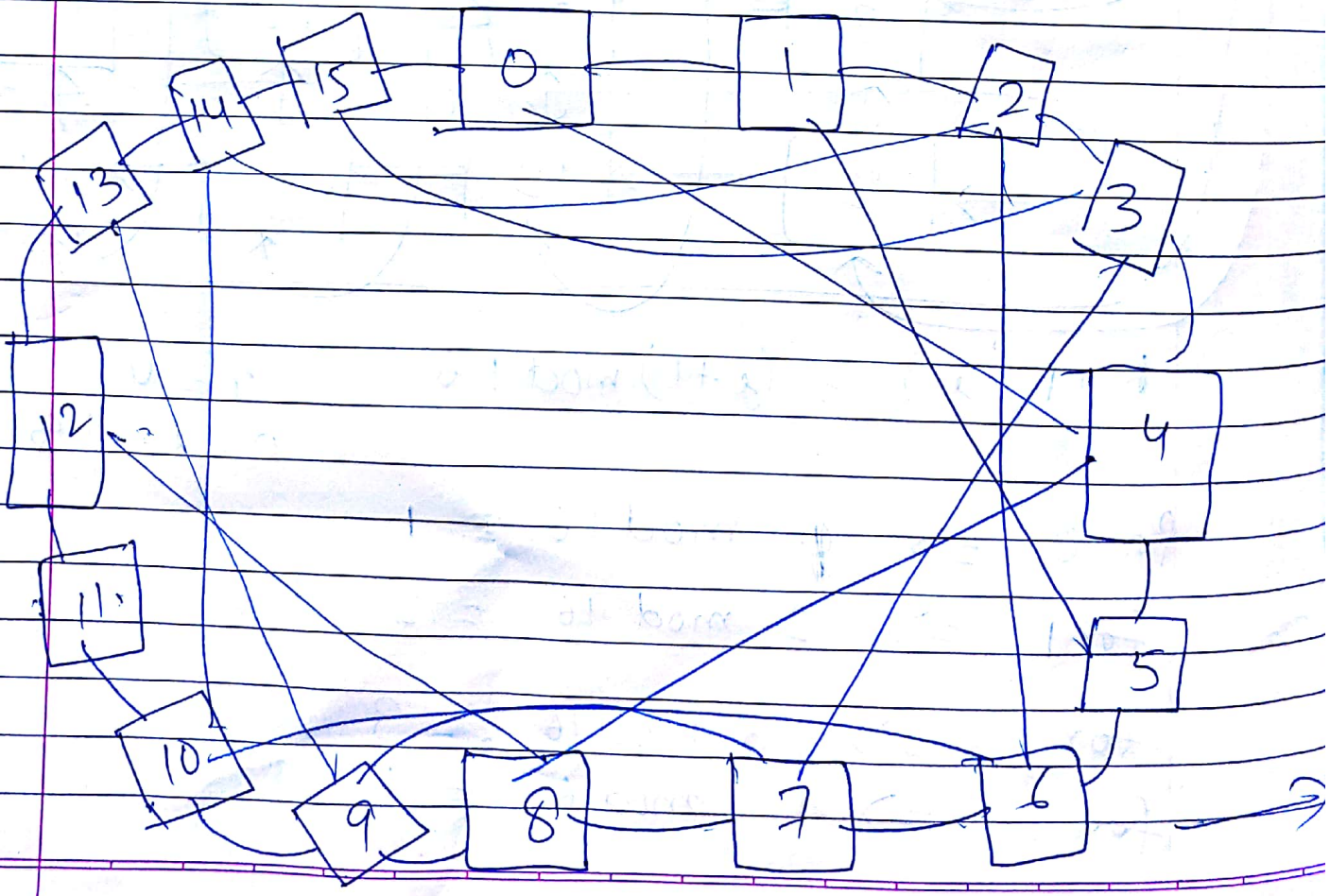
for 1 $\Rightarrow 0 \mod 16 = 0$

$$R+4(i) = (i+4) \mod 16$$

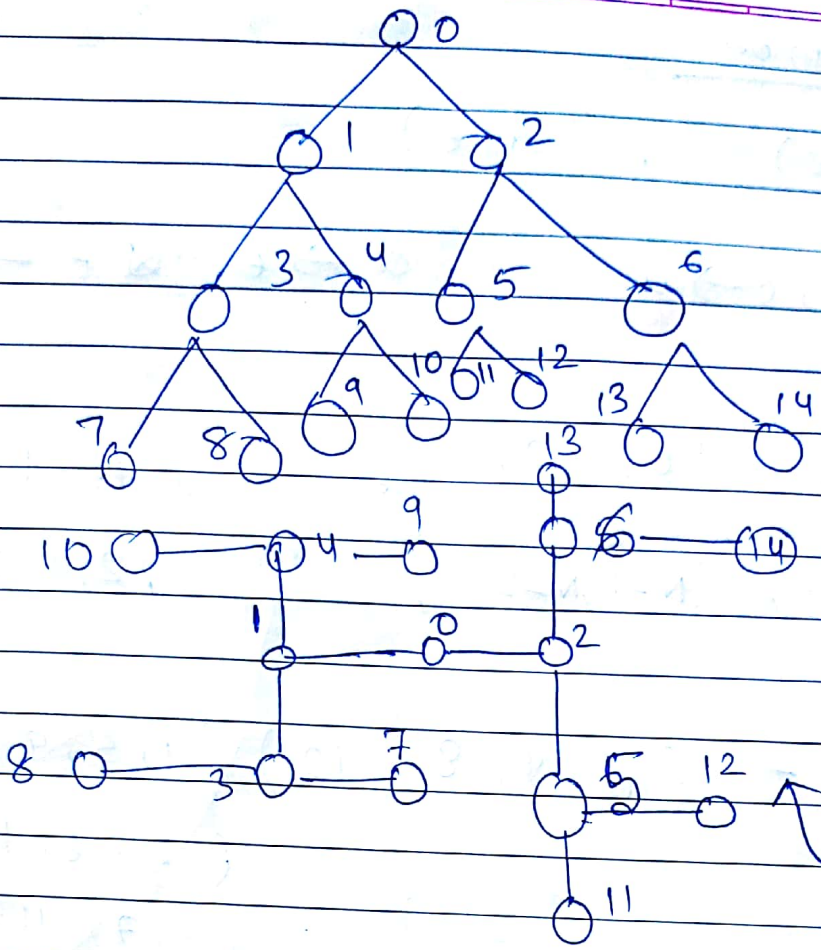
for 0 $\Rightarrow 4 \mod 16 = 4$

$$R-4(i) = (i-4) \mod 16$$

for 0 $\Rightarrow -4 \mod 16$



Scanned by CamScanner



$2n - 1$

$= 2 \times 8 - 1$

$= 15$

nodes

Space is not utilized completely.

If original Illiac Routing function is implemented for 64 PE (Processing element).

R+1

R-1

R+4

R-4

This is not suitable for fabrication

Scanned by CamScanner

Page No.:

Permutation:

(a, b, c) , (d, e)

$a \to b, b \to c, c \to a$ $\quad d \to e$ & $e \to d$

$R+1(i) = (0, 1, 2, \ldots, N-1)$

$R-1(i) = (N-1, N-2, \ldots, 2, 1, 0)$

$R+r(i) = (0, 4, 8, 12)(1, 5, 9, 13)$
$(2, 6, 10, 14)$
$(3, 7, 11, 15)$

$= \bigwedge_{i=0}^{r-1} (i, i+r, i+2r, \ldots, i+N-r)$

$R_r = \bigwedge_{i=0}^{r-1}$

$l = a + (n-1)d$

$= i + (r-1)r$

$= i + r^2 - r$

$= i + N - r$

$r, 2r, 3r$

$r + (n-1)r$ $\quad a_n = r(r^n - 1)$

Scanned by CamScanner

Barrel Shifter.

$$B+J(i) = (i + 2^J) \pmod N$$

$$B-J(i) = (i - 2^J) \pmod N$$

$0 \le i \le N-1$, $0 \le J \le n-1$ and $n = \log_2 N$

$B+0 = R+1$, $B-0 = R-1$, $B+n/2 = R+r$

$B-n/2 = R-r$

$B \le (\log_2 N)/2$

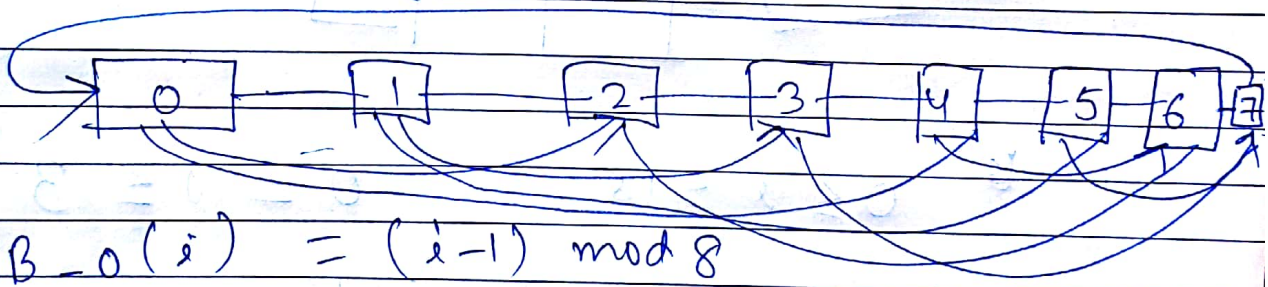
CIS.

$B \pm J(i) = (i \pm 2^J) \bmod N$

$N = 8$.

$0 \le i \le 7$, $0 \le J \le 2$

$B+0(i) = (i+1) \bmod 8$



$B-0(i) = (i-1) \bmod 8$

$B+1(i) = (i+2) \bmod 8$

$B+2(i) = (i+4) \bmod 8$

$B-2(i) = (i-4) \bmod 8$

$0, r, 2r, \ldots, n-1$

i, r

$i + nr - 2r$

$i + (n-1)r$

$r + (n-2)r$

$nr - r$

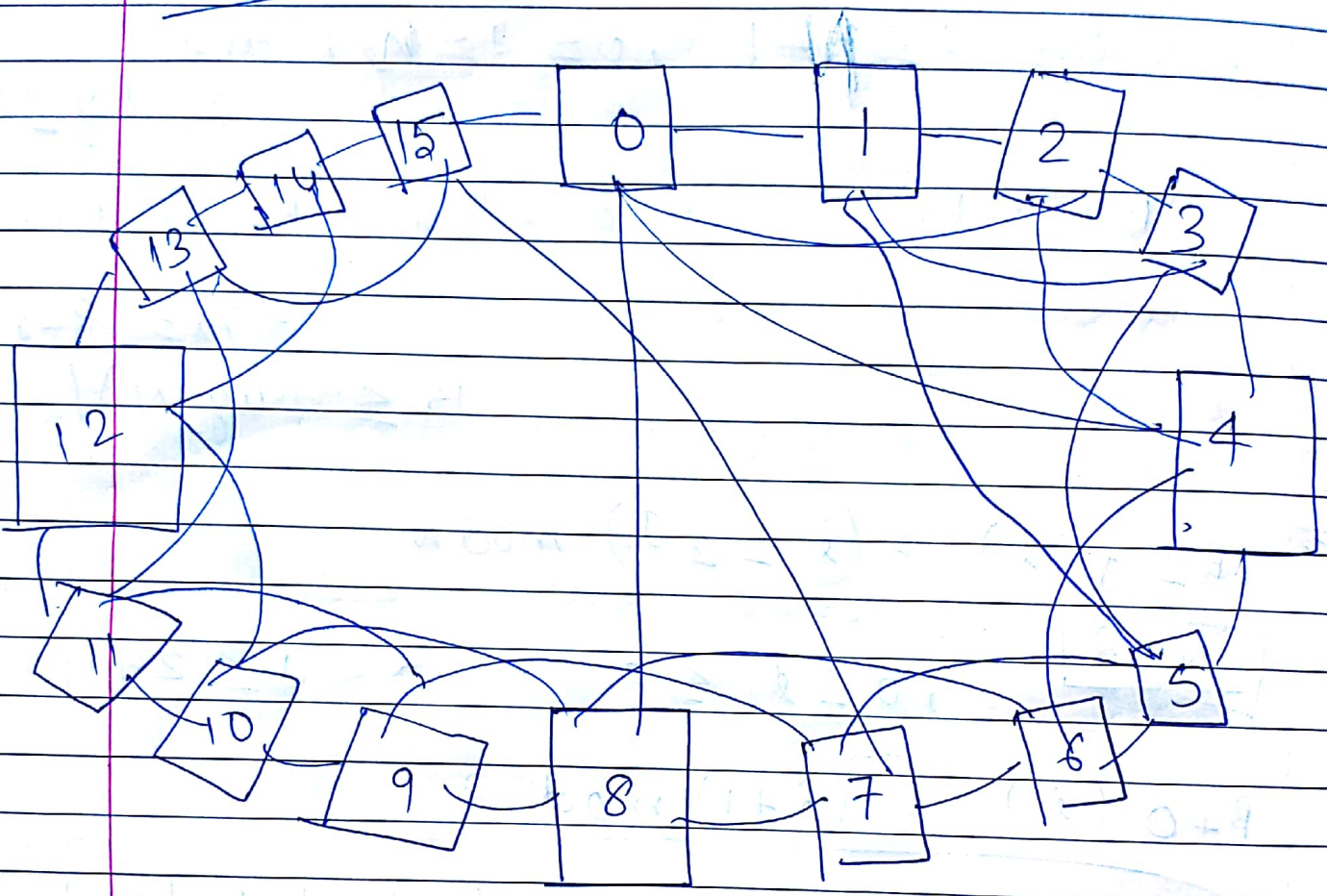
$(r-1)r$.

Scanned by CamScanner

Redraw for $N = 16$ in the form of a chordal ring.

$$B \pm J(i) = (i \pm 2^J) \bmod N$$

$N = 16$



$$0 \le i \le 15, \quad 0 \le J \le 3$$

$$B+0(i) = (i + 1) \bmod 16$$

$$B-0(i) = (i - 1) \bmod 16$$

$B \pm 0$

$B \pm 1$

$B \pm 2$

$B \pm 3$

Scanned by CamScanner

Page No.:

Prove that the set of Illiac routing functions is a subset of Barrel Shifter.

$R \pm r(i) = (i \pm r) \bmod N$

$r = \sqrt{N}$, $0 \le i \le N-1$

Barrel:

$B \pm J(i) = (i \pm 2^J) \bmod N$

$0 \le i \le N-1$,

$0 \le J \le n-1$

$n = \log_2 N$

Let $N = 4$

$r = 2$, $0 \le i \le 3$

$R \pm 2(i) = (i \pm 2) \bmod 4$

Let $N = 4$

$0 \le i \le 3$

$0 \le J \le 1$

$B \pm 1(i) = (i \pm 2^J) \bmod N$

$R \pm 1(i) = (i \pm 1) \bmod N$

$R \pm r(i) = (i \pm r) \bmod N$, $r = \sqrt{N}$

$B \pm J(i) = (i \pm 2^J) \bmod N$

$B \pm 0(i) = (i \pm 1) \bmod N = R \pm 1(i)$

$B \pm 1(i) = i \pm 2)$

Scanned by CamScanner

$n = \log_2 N$

$N = 2^n$

$\sqrt{N} = 2^{n/2}$

$$B_{\pm n/2}(i) = (i \pm 2^{n/2}) \bmod N$$

$$= (i \pm \sqrt{N}) \bmod N$$

$$= (i \pm r) \bmod N$$

$$= R_{\pm r}(i)$$

Task:

Expres the Barrel Shift Routing function in the form of Permutation.

Scanned by CamScanner

Page No.:

Q. Draw the Barrel Shifter with 256 PEs by showing ~~atleast~~ all the connections ~~of atleast~~ from at least from one P.E.

$$B \pm J(i) = (i \pm 2^J) \bmod N$$

$$0 \le i \le N-1, \quad 0 \le J \le n-1$$

$$0 \le i \le 255, \quad 0 \le J \le 7$$

$$n = \log_2 N$$

$$= \log_2 256$$

$$= \log_2 2^8$$

$$= 8$$

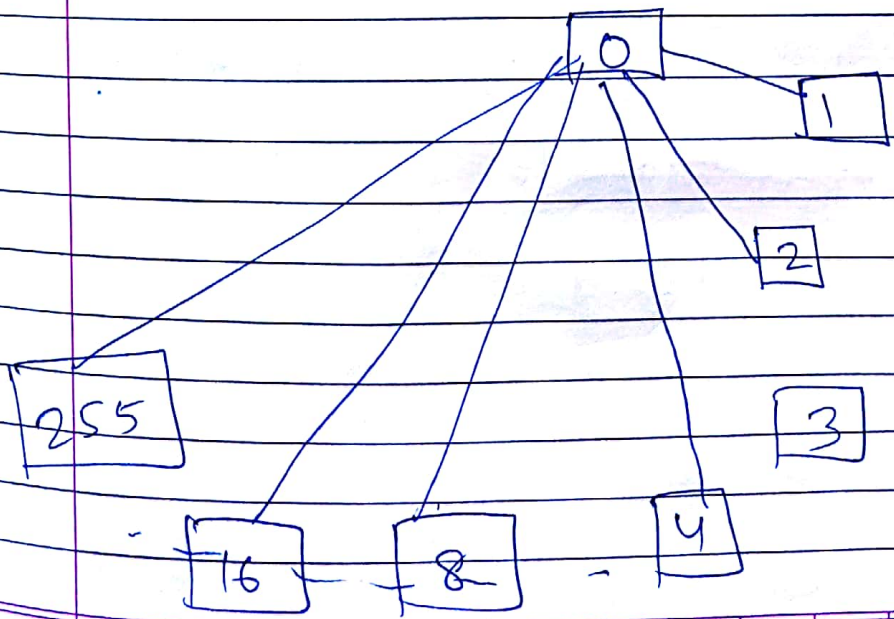
$$B \pm 0(i) = (i \pm 1) \bmod 256$$

for $i = 0 \Rightarrow (\pm 1) \bmod 256$

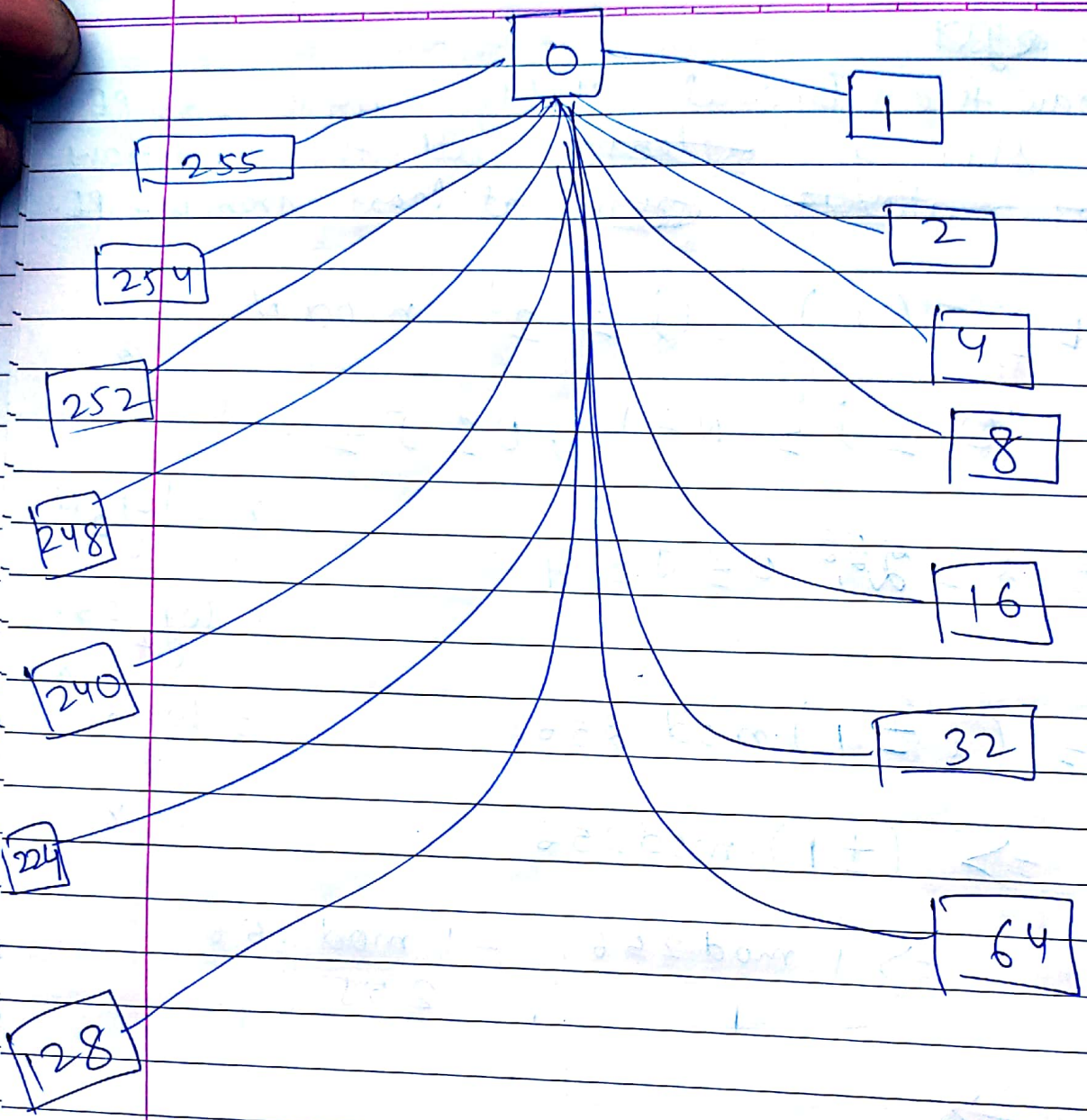
$\Rightarrow 1 \bmod 256, \quad -1 \bmod 256$

$1, \quad 255$

$i = 1$: $B \pm 1 \longrightarrow$



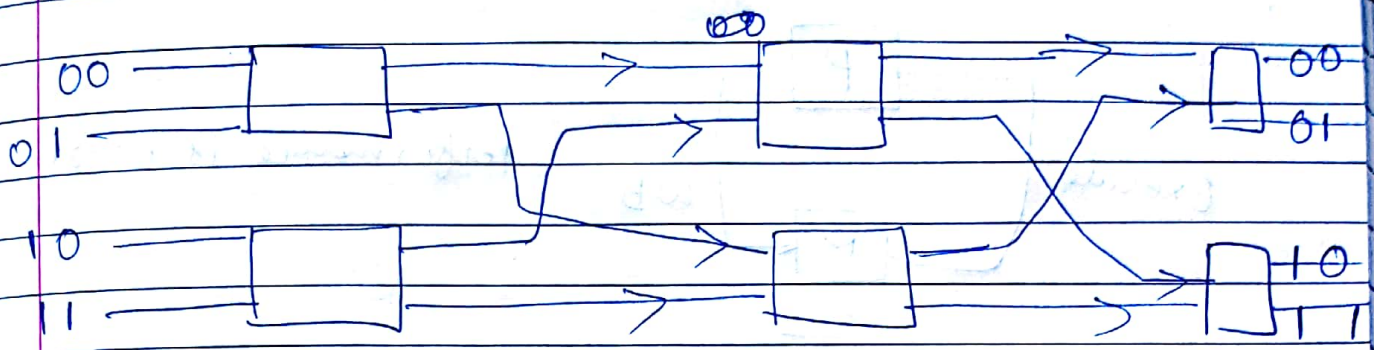
Scanned by CamScanner



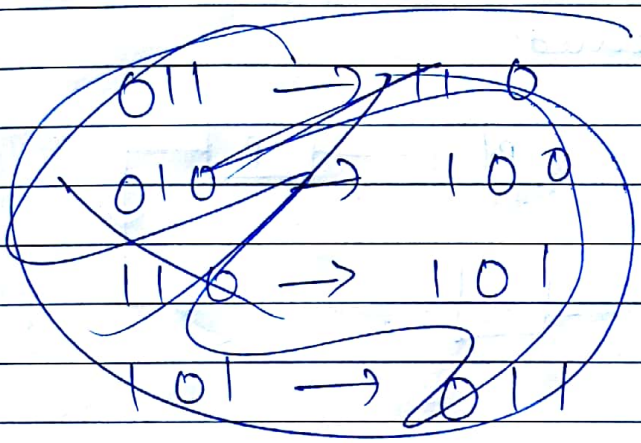
Scanned by CamScanner

Page No.:

Prove/Disprove that the multistage Omega network can't perform shift permutation.

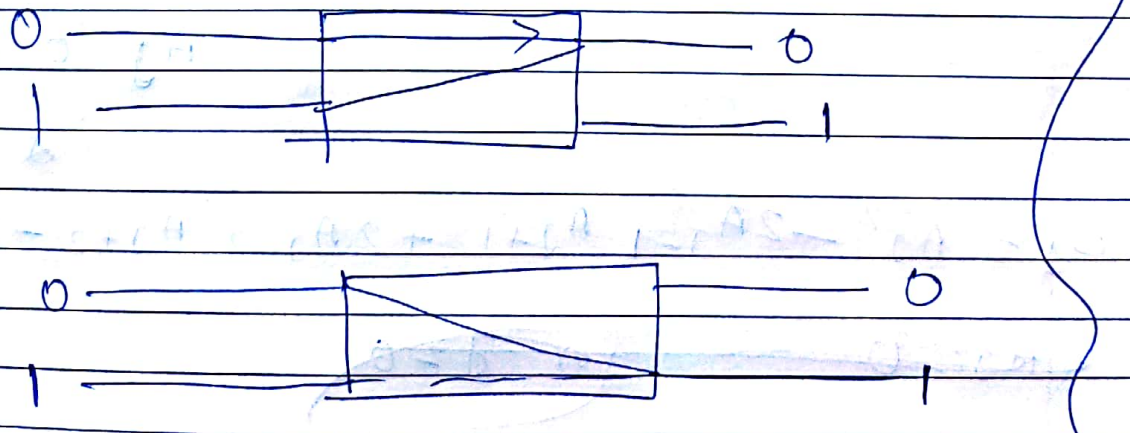


2 Stage 2X2 Omega N/w



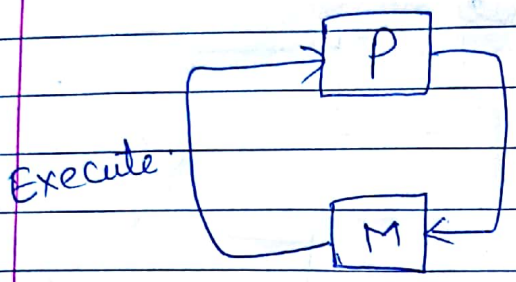
00 → 00
01 → 10
10 → 01
11 → 11

Not Permitted.



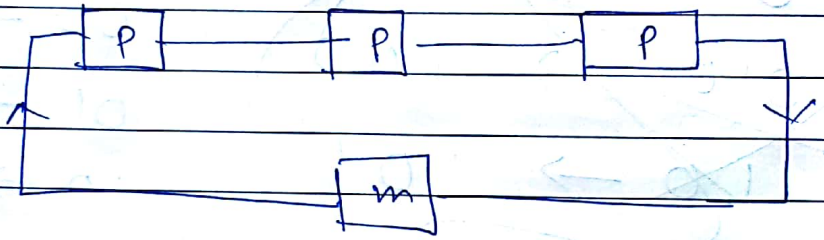
Scanned by CamScanner

Systolic System



Performance is poor.

By choosing collection of Processor, Performance can be increased.



Ultraeffe's Root squaring.

$A_J = 0$ for $J < 0$ & $J > n$

$C_J = A_J^2 - 2A_{J-1}A_{J+1} + 2A_{J-2}A_{J+2} - 2A_{J-3}A_{J+3}$

~~$A_J = 0$~~ ~~for $J = 0$~~ ~~$J = 1$~~

$C_0 = A_0^2$

$C_1 = A_1^2 - 2A_0A_2$

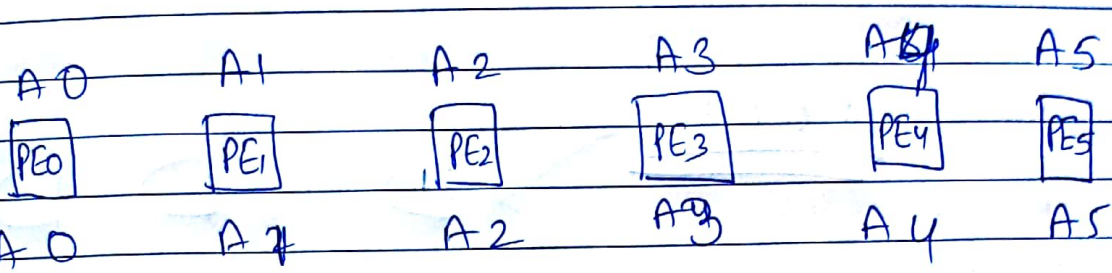
$C_2 = A_2^2 - 2A_1A_3 + 2A_0A_4$

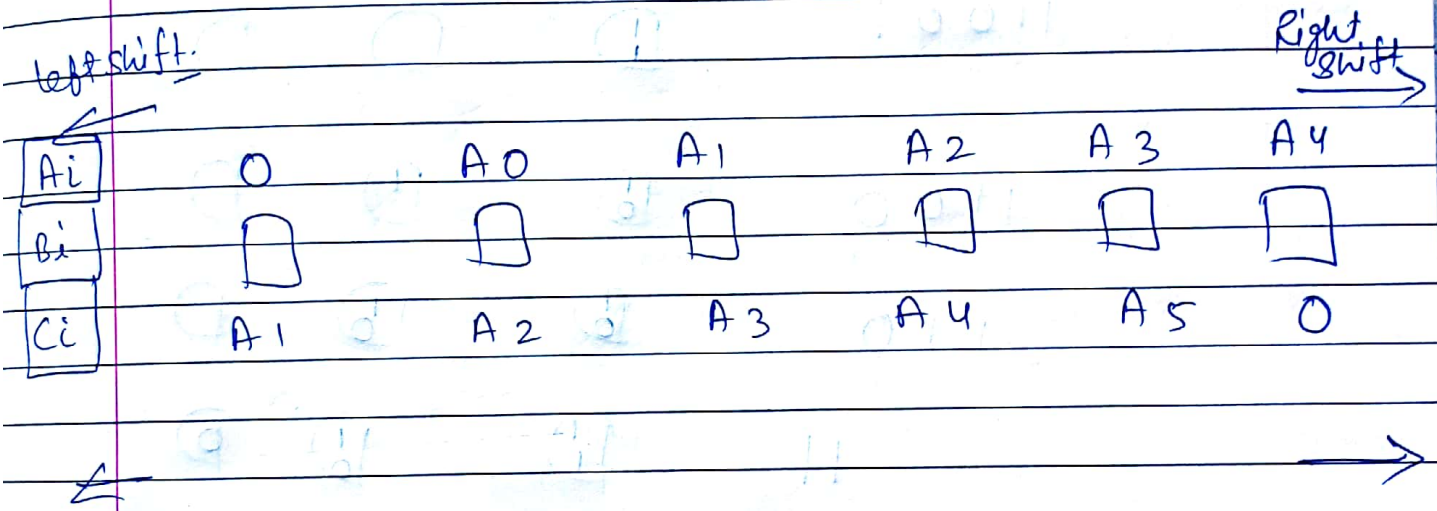
Scanned by CamScanner

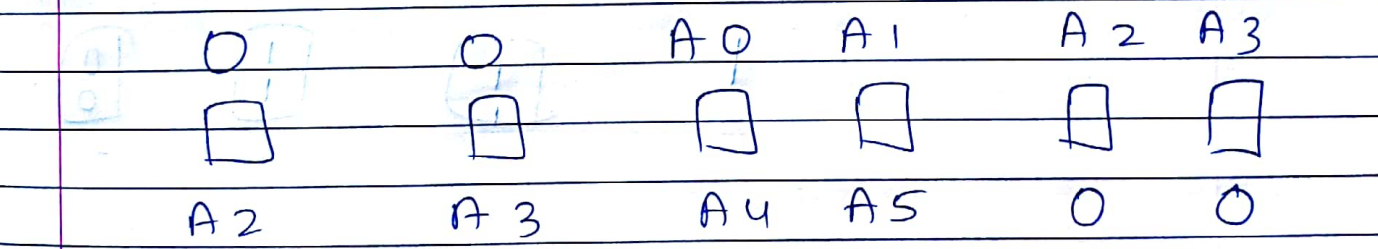
$C_3 = A_3^2 - 2A_2A_4 + 2A_1A_5 - 2A_0A_6$

$C_4 = A_4^2 - 2A_3A_5$

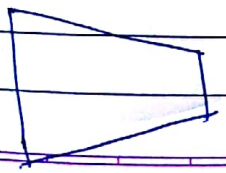
$C_5 = A_5^2$



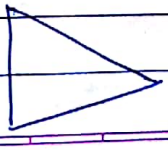




n=5



n=6



for odd value
blunt edge
for even value.
Sharp edge.

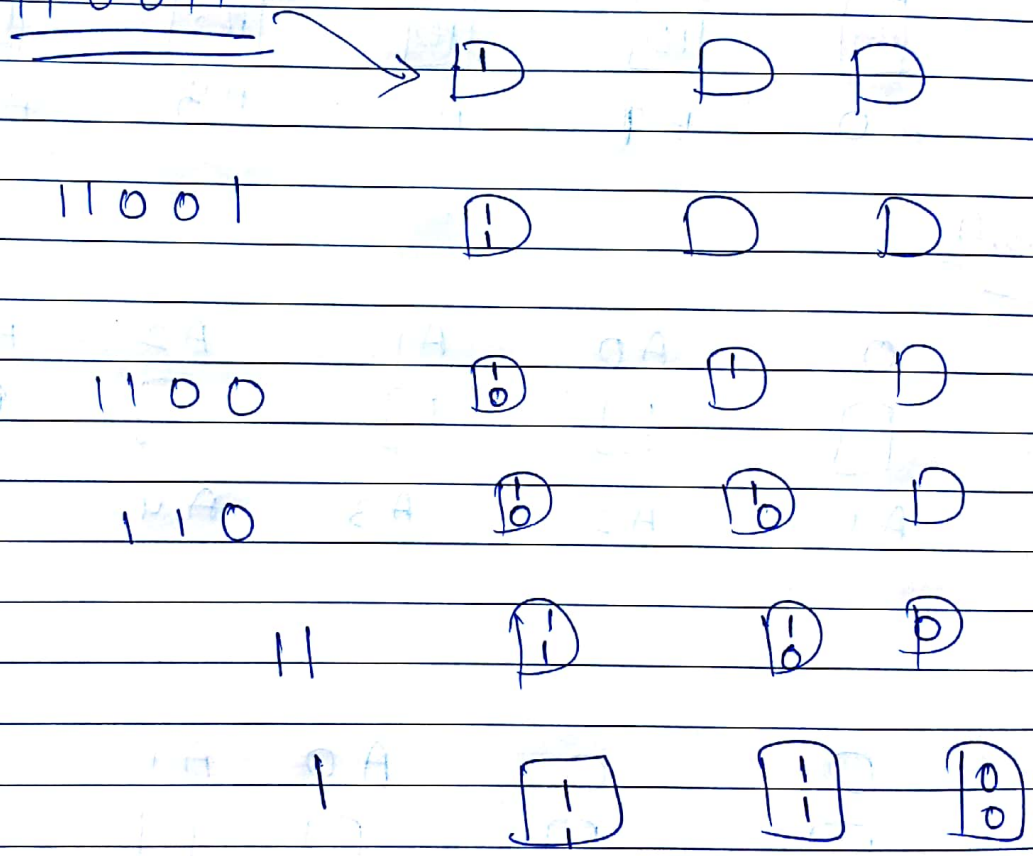
Scanned by CamScanner

Page No.:

Palindrome: $\longrightarrow$ form of ww^r

w^r is reverse of w.

11011 is a palindrome,



Scanned by CamScanner

another form: w c w^r 10001

Page No.:

Buffered reader . org. [SOC]
System on chip.

Diffrence b/w CPU & CPU.

Scanned by CamScanner