Module 4

- Message passing mechanism
  - Message routing schemes
  - Flow control strategies
  - Multicast routing algorithm
- Pipelining and superscalar technique
  - Linear pipeline processors
  - Nonlinear pipeline processors

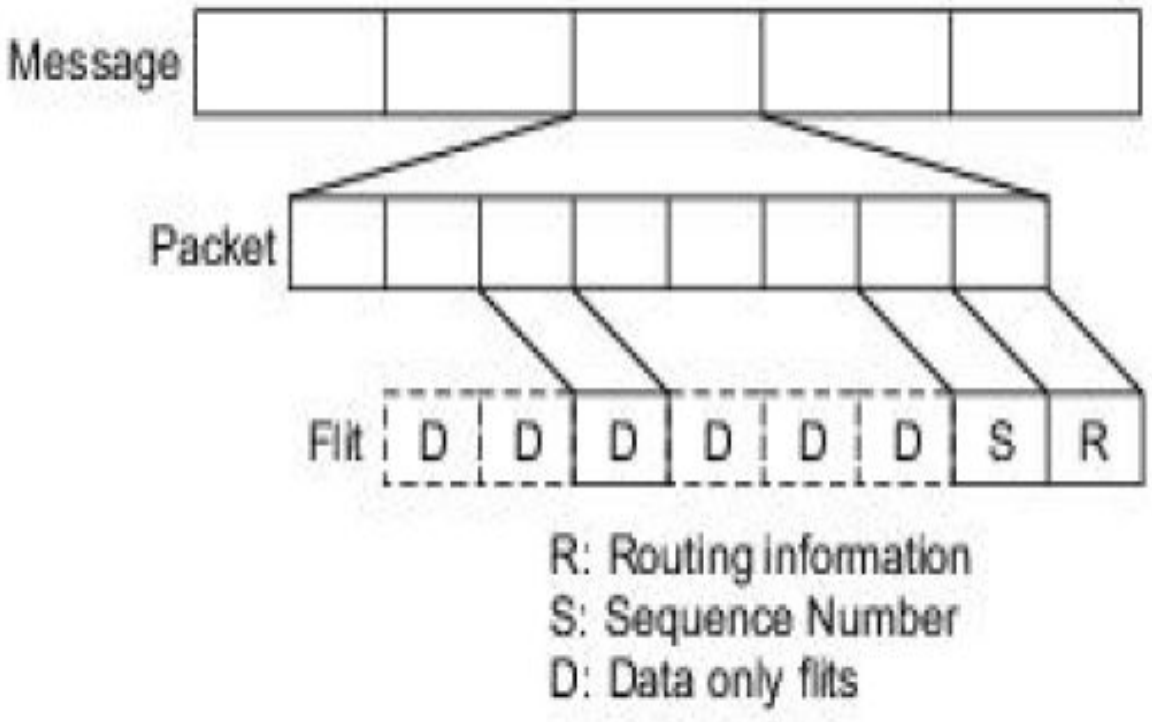
Message passing mechanism

- Message passing in a multicomputer network demands special hardware and software support.

- Message Routing schemes
- Flow control Strategies
- Multicast Routing Algorithms

Message-Routing Schemes(Cont..)

Massage Format



Massage Format

- A massage is the logical unit for internode communication.
- It is often assembled from an arbitrary number of fixed-length packets It may have a variable length

Massage Format

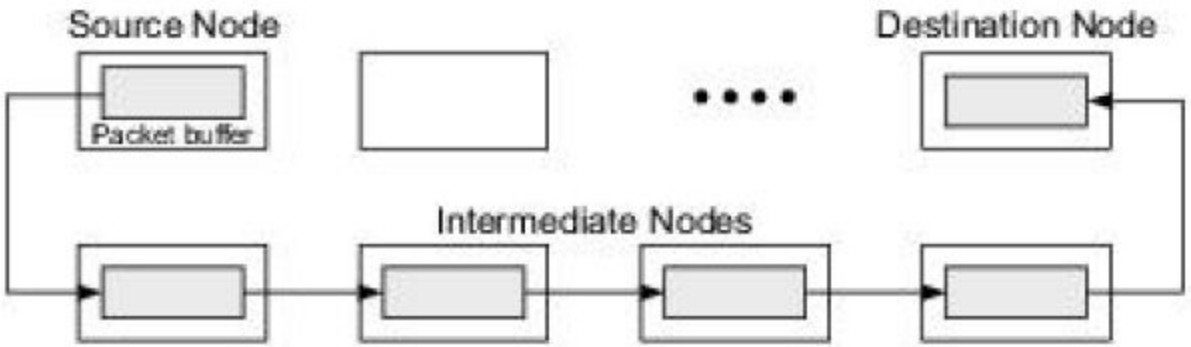
- A packer is the basic unit containing the destination address for routing purposes.
- Because different packets may arrive at the destination asynchronously, A sequence number is needed in each packet to allow reassembly of the message transmitted.
- A packet can be further divided into a number of fixed-length flits (flow control digits).
- Routing information (destination) and sequence number occupy the header flits.
- The remaining flits are the data elements of a packet.

Massage Format

- In multicomputers with store-and-forward routing,
- packets are the smallest unit of information transmission.
- In wormhole routed networks, packets are further subdivided into flits (flow control digits).
- The flit length is often affected by the network size.
- The packet length is determined by the routing scheme and network implementation.
- Typical packet lengths range from 64 to 512 bits.
- The sequence number may occupy one to two flits depending on the message length.
- Other factors affecting the choice of packet and flit sizes include channel bandwidth, router design, network traffic intensity, etc.

Store and Forward Routing

Packets are the basic unit of information flow in a store and forward network.



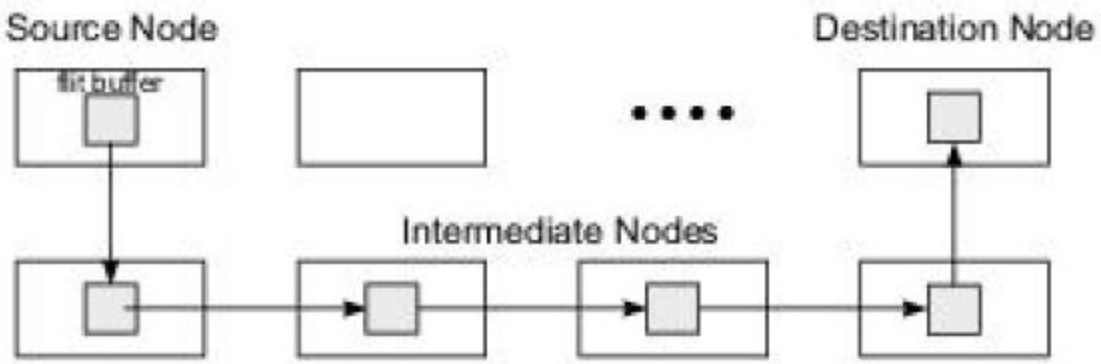
(a) Store-and-forward routing using packet buffers in successive nodes

Store-and-Forward Routing

- Each node is required to use a packet buffer.
- A packet is transmitted from a source node to a destination node through a sequence of intermediate nodes.
- When a packet reaches an intermediate node, it is first stored in the buffer.
- Then it is forwarded to the next node if the desired output channel and a packet buffer in the receiving node are both available.
- The latency in store-and-forward networks is directly proportional to the distance (the number of hops) between the source and the destination.
- This routing scheme was implemented in the first generation of multicomputers.

Wormhole Routing

sub dividing the packet into smaller flits, latter generations of multi computers implement wormhole routing scheme,



(b) Wormhole routing using flit buffers in successive routers

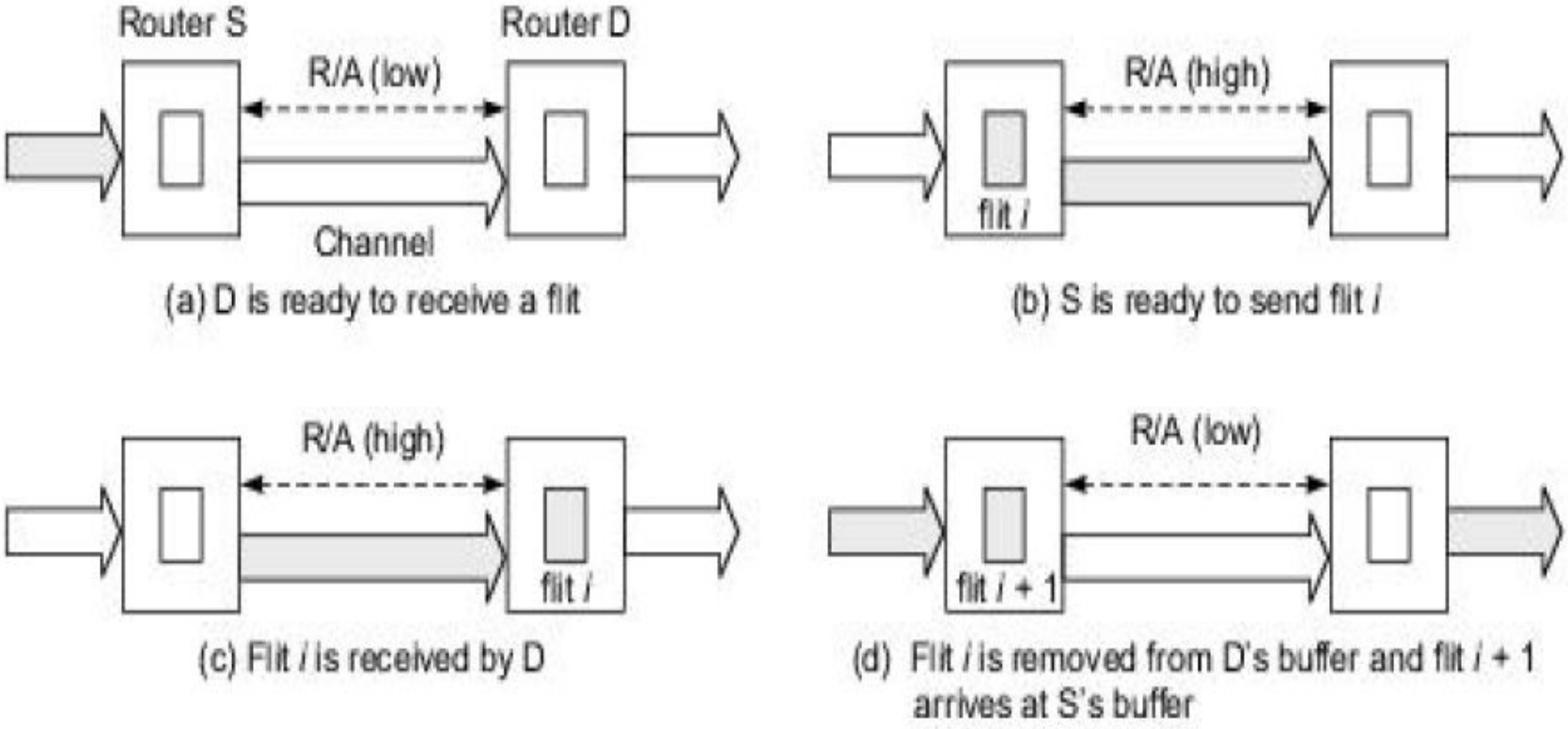
Wormhole Routing

- By subdividing the packet into smaller flits, latter generations of
- multicomputers implement the wormhole routing scheme.
- Flit buffers are used in the hardware routers attached to nodes.
- The transmission from the source node to the destination node is done through a sequence of routers.
- All the flits in the same packet are transmitted in order as inseparable companions in a pipelined fashion.
- The packet can be visualized as a railroad train with an engine car (the header flit) towing a long sequence of box cars (data flits).
- Only the header flit knows where the train (packet) is going.
- All the data flits (box cars) must follow the header flit.
- Different packets can be interleaved during transmission.
- However, the flits from different packets cannot be mixed up.
- Otherwise they may be towed to the wrong destinations.

Asynchronous Pipelining

The pipelining of successive flits in a packet is done asynchronously using a handshaking protocol.

handshaking protocol



handshaking protocol

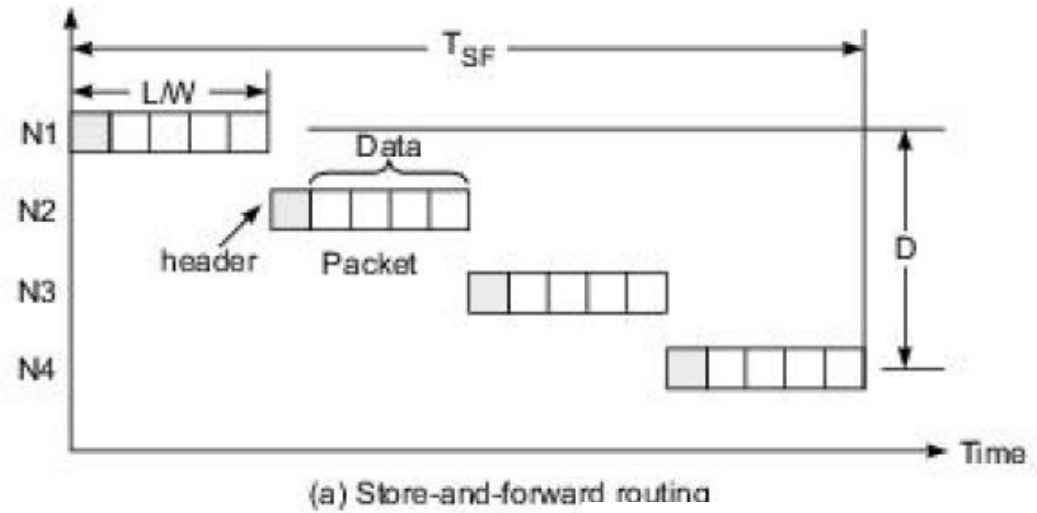
- A 1-bit ready/request (R/A) line is used between adjacent routers.
- When the receiving router (D) is ready to receive a flit
- it pulls the R/A line low
- When the sending router (S) is ready it raises the line high and transmits flit i through the channel.

handshaking protocol

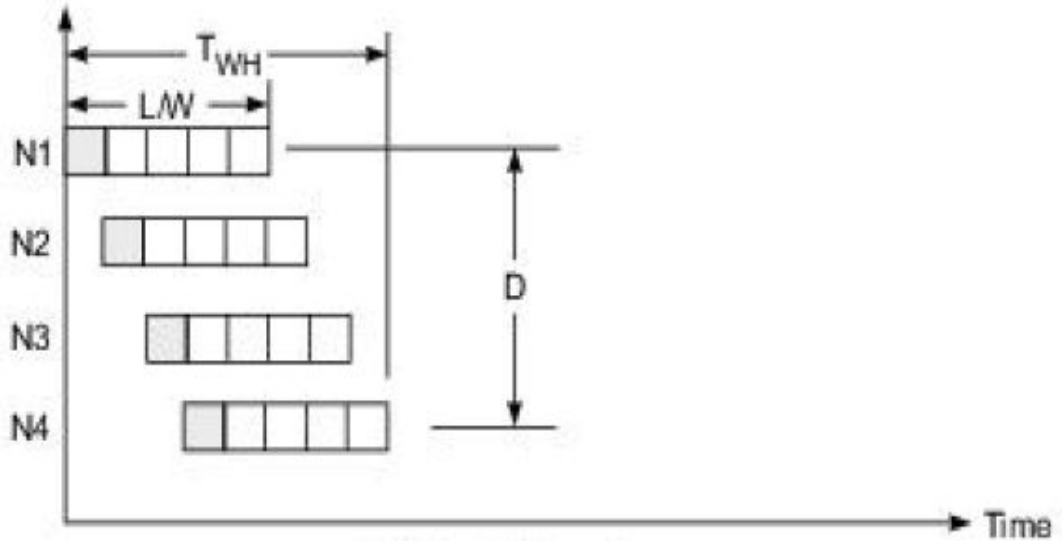
- While the flit is being received by D the RM line is kept high
- After flit I is removed from D's buffer (i.e. is transmitted to the next node)
- the cycle repeats itself for the transmission of the neat flit i + 1 until the entire packet is transmitted.

- Asynchronous pipelining can be very efficient,
- the clock used can be faster than that used in a synchronous pipeline
- The pipeline can be stalled if flit buffers or successive channels along the path are not available during certain cycles

Latency analysis



(a) Store-and-forward routing



(a) Wormhole routing

Latency analysis

- The communication latency T_{SF} for a store-and-forward network

$$T_{SF}=\frac{L}{W}(D+1)$$

- The latency T_{WH}, for a wormhole-routed network is

$$T_{WH}=\frac{L}{W}+\frac{F}{W}xD$$

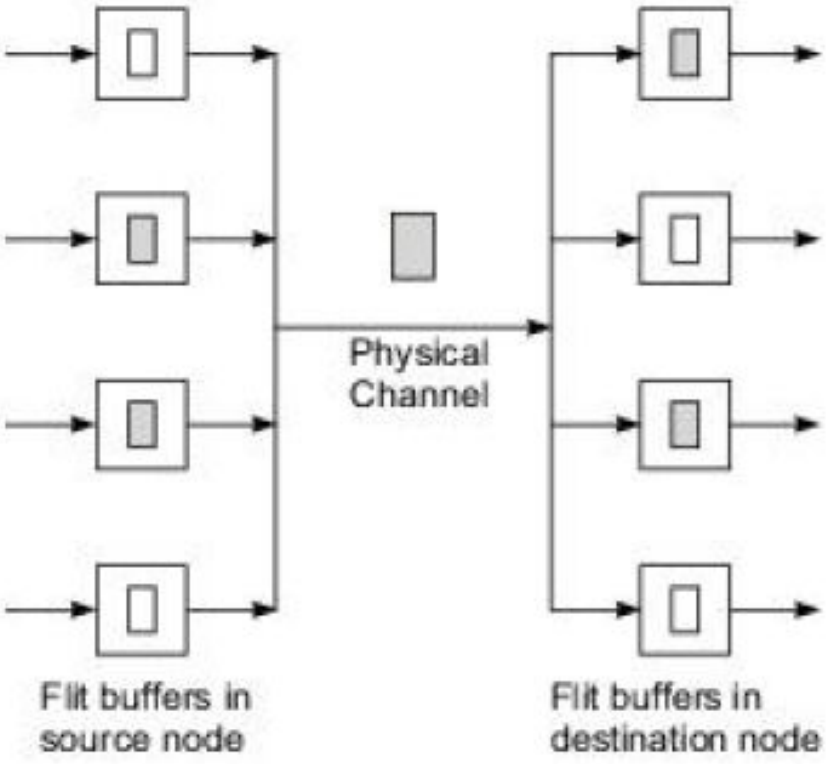
L be the packet length (in bits), W the channel bandwidth (in bits/s), D the distance (number of nodes traversed minus 1), and F the flit length (in bits).

wormhole routing would still have much lower latency than packet store-and-forward routing

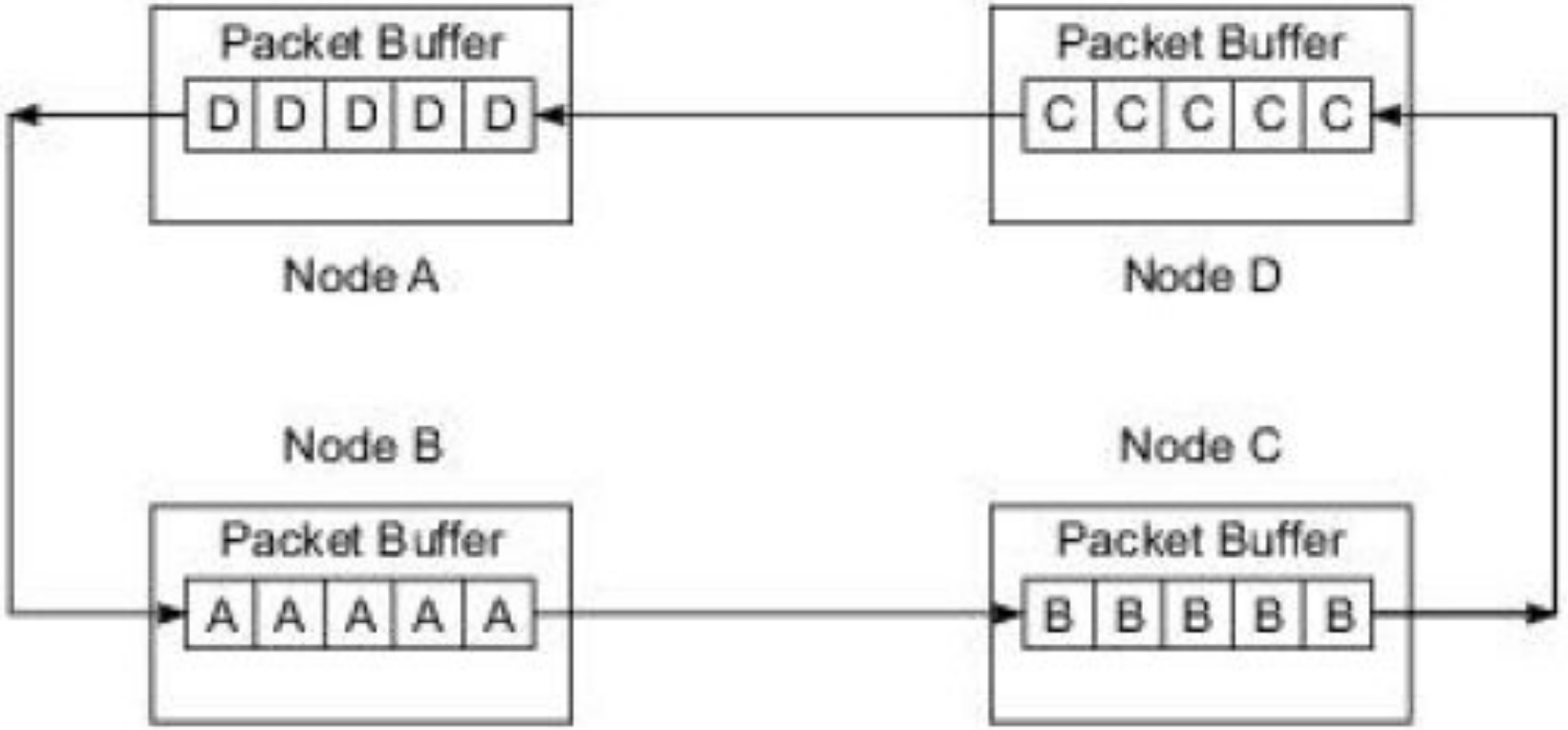
- Ignored the network startup latency and block time due to resource shortage (such as channels being busy or buffers being full, etc.)
- The channel propagation delay has also been ignored because it is much smaller than the terms in TSF or TWH·
- A typical first generation value of TSF is between 2000µs and 6000µs, while a typical value of TWH is 5µs or less.
- Current systems employ much faster processors, data links and routers.
- Both the latency figures above would therefore be smaller, but wormhole routing would still have much lower latency than packet store-and-forward routing.

Deadlock and Virtual Channels

- Virtual Channel: A virtual channel is a logical link between two nodes.

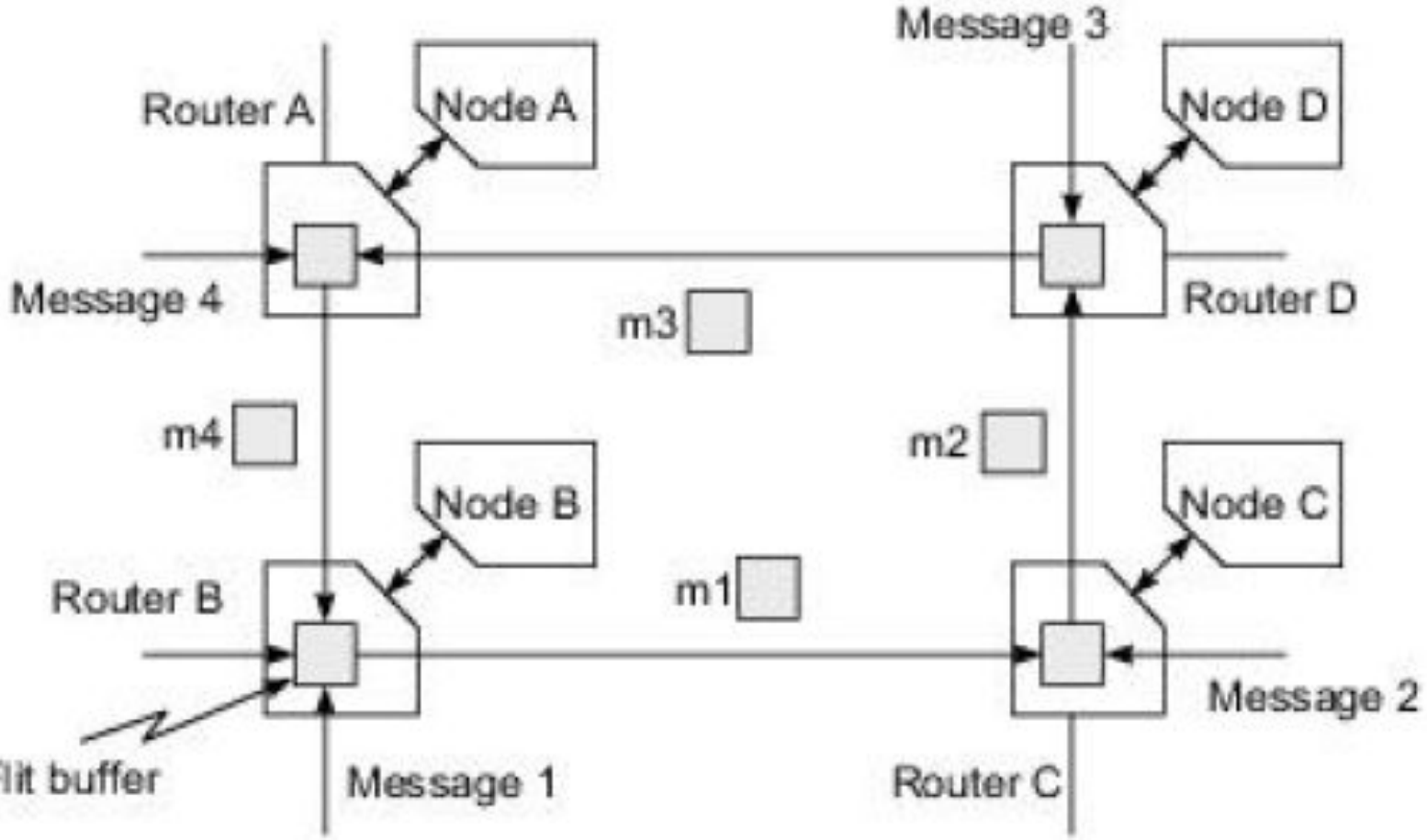


The deadlock situations caused by circular waits at buffers or at channels

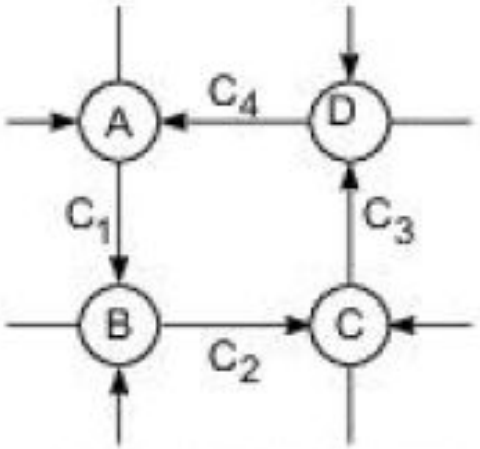


(a) Buffer deadlock among four nodes with store-and-forward routing

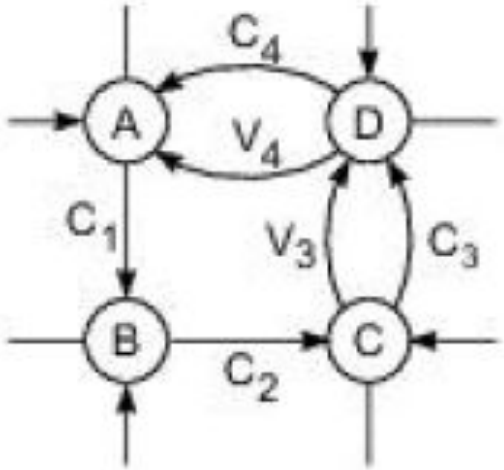
The deadlock situations caused by circular waits at buffers or at channels



Channel deadlock among four nodes with wormhole routing; shaded boxes are flit buffers



(a) Channel deadlock



(c) Adding two virtual channels (V_3, V_4)

Flow Control Strategies

- To control smooth network traffic flow without causing congestion or deadlock situations.

1. Pocket Collision Resolution
2. Dimension-Order Routing
3. E-cube Routing on Hypercube
4. X-Y Routing on a 2D Mesh
5. Adaptive Routing

1. Pocket Collision Resolution

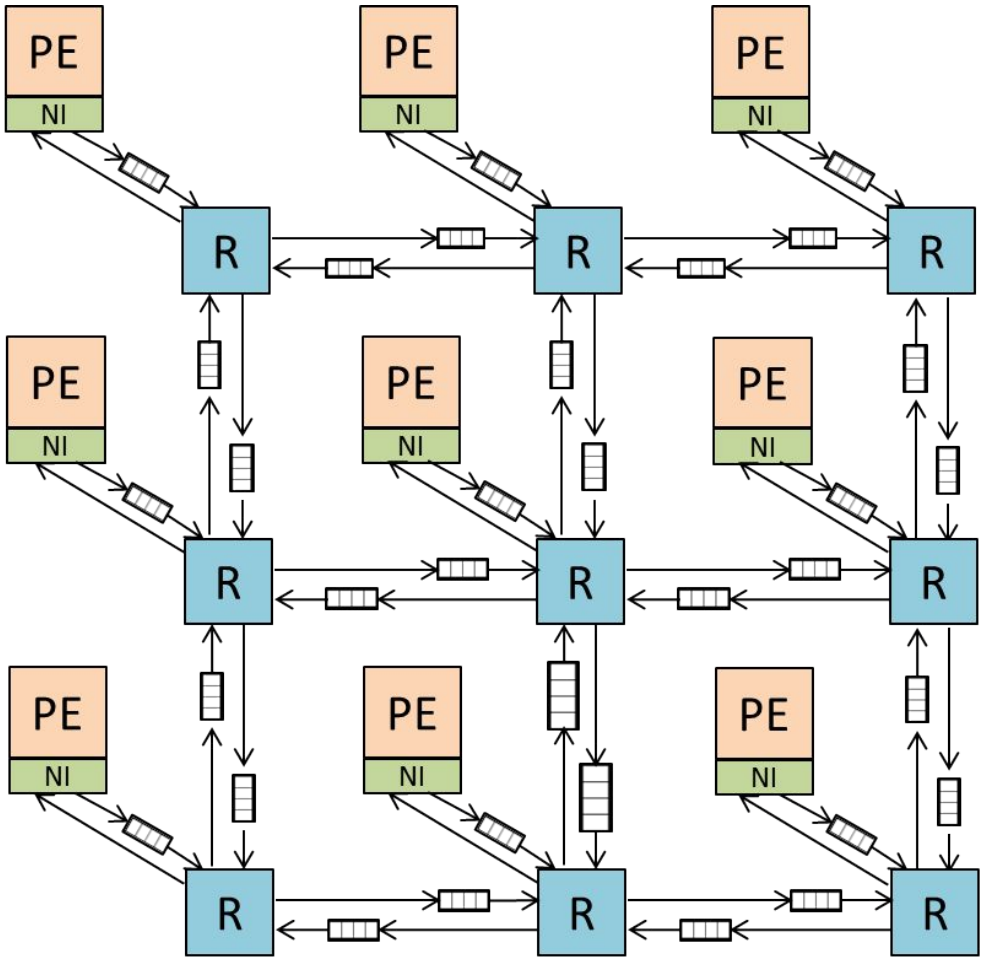
move a flit between adjacent nodes in a pipeline of channels

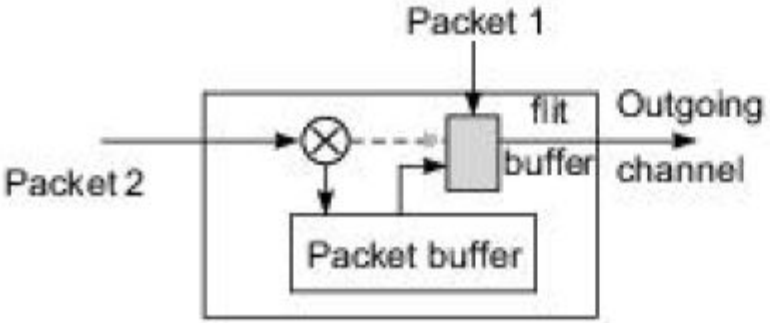
- Three elements
  1. The source buffer holding the flit
  2. The channel being allocated
  3. The receiver buffer accepting the flit

- When two packets reach the same node,
- They may request the same receiver buffer or the same outgoing channel.
- Two arbitration decisions must be made

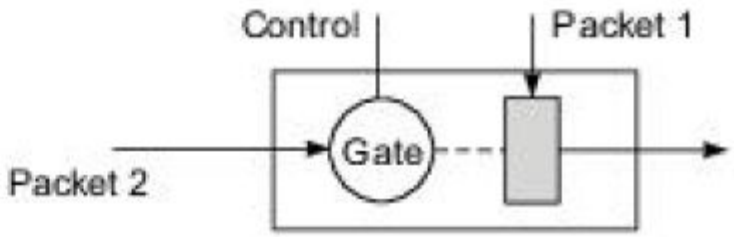
1. Which packet will be allocated the channel?
2. What will he done with the packet being denied the channel?

Flow Control Strategies

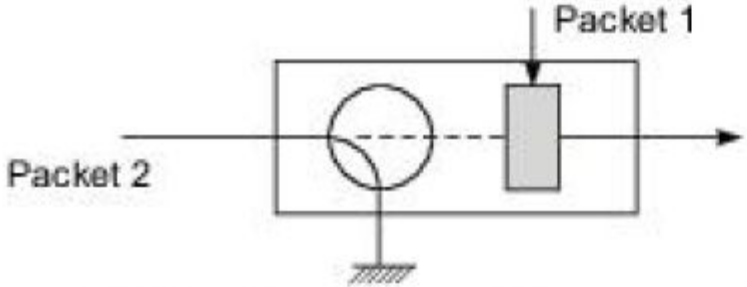




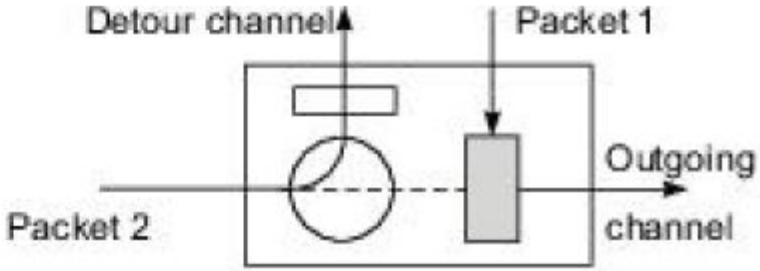
(a) Buffering in virtual cut-through routing



(b) Blocking flow control



(c) Discard and retransmission



(d) Detour after being blocked

- Packet l is being allocated the channel, and packet 2 being denied.
- Packet 2 is temporarily stored in a packet buffer.
- When the channel becomes available later, it will be transmitted .
- The packet buffer however may cause significant storage delay.

1. Virtual cut-through routing

- A buffering method has been proposed with the virtual cut-through routing scheme.
- Packet 2 is temporarily stored in a packet buffer.
- When the channel becomes available later, it will be transmitted then.
- This buffering approach has the advantage of not wasting the resources already allocated.
- However, it requires the use of a large buffer to hold the entire packet.
- The packet buffers along the communication path should not form a cycle.
- The packet buffer however may cause significant storage delay.
- The virtual cut through method offers a compromise by combining the store-and-forward and wormhole routing schemes.
- When collisions do not occur, the scheme should perform as well as wormhole routing.
- In the worst case, it will behave like a store-and-forward network

2.Blocking and 3.Discard policy

- Pure wormhole routing uses a blocking policy in case of packet collision. The second packet is being blocked from advancing; however, it is not being abandoned. The blocking policy is economical to implement but may result in the idling of resources allocated to the blocked packet

- The discard policy, which simply drops the packet being blocked from passing through. The discard policy may result in a severe waste of resources, and it demands packet retransmission and acknowledgment. Otherwise, a packet may be lost after discarding. This policy is rarely used now because of its unstable packet delivery rate. The BBN Butterfly network had used this discard policy.

4. Detour policy

- The fourth policy is called detour policy.
- The blocked packet is routed to a detour channel.
- The blocking policy is economical to implement but may result in the idling of resources allocated to the blocked packet.
- Detour routing offers more flexibility in packet routing.
- However, the detour may waste more channel resources than necessary to reach the destination.
- Furthermore, a re-routed packet may enter a cycle of **livelock**, which wastes network resources.
- Both the Connection Machine and the Denelcor HEP used this detour policy.

Dimension-Order Routing

- Packet routing can be conducted deterministically or adaptively.
- In deterministic routing, the communication path is completely determined by the source and destination addresses.
- In other words, the routing path is uniquely predetermined in advance, independent of network condition.
- Adaptive routing may depend on network conditions, and alternate paths are possible.
- In both types of routing, deadlock free algorithms are desired.
- Two such deterministic routing algorithms are given below, based on a concept called dimension order routing.

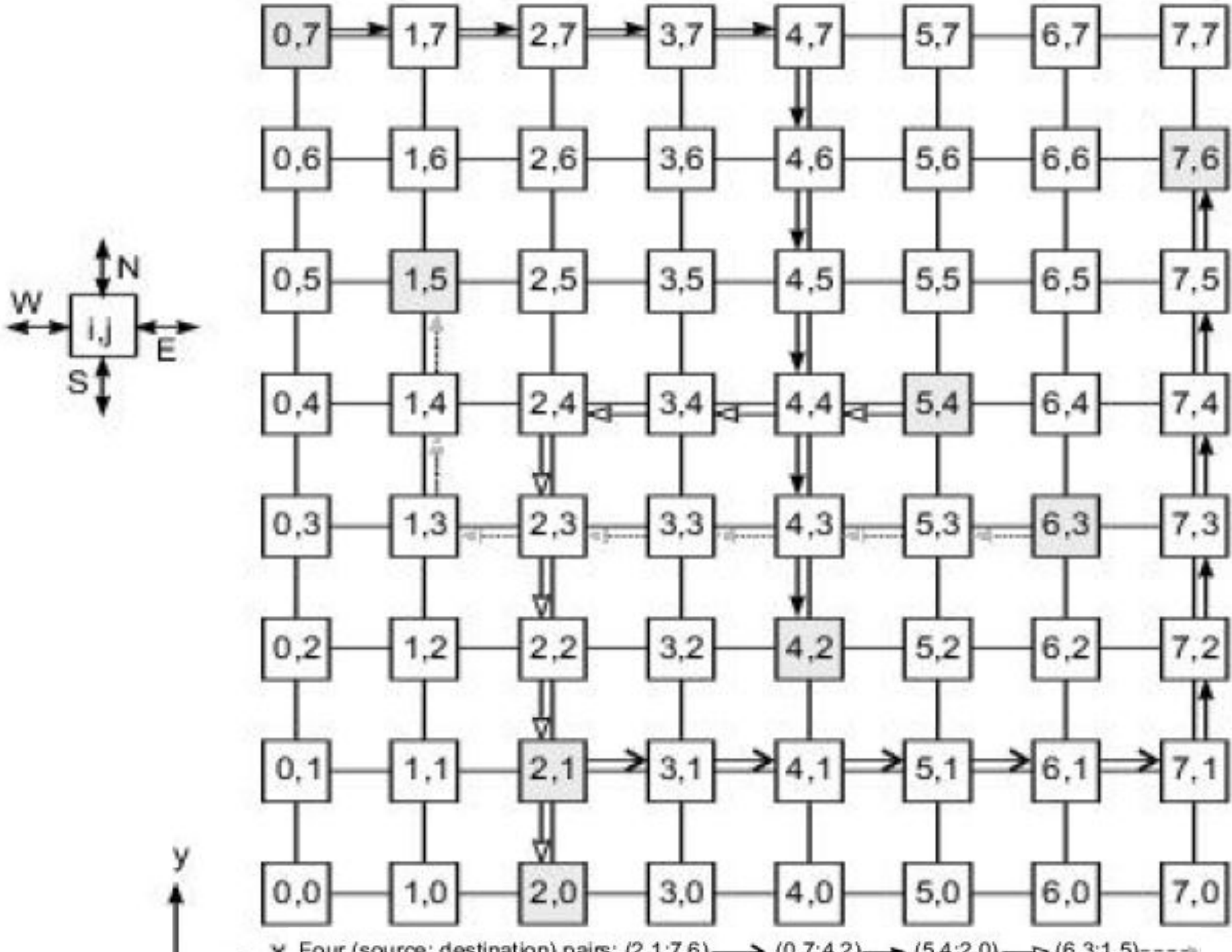
Dimension-Order Routing cont.

- Dimension-order routing requires the selection of successive channels to follow a specific order based on the dimensions of a multidimensional network.
- In the case of a two-dimensional mesh network, the scheme is called X-Y routing because a routing path along the X-dimension is decided first before choosing a path along the Y-dimension.
- For hypercube (or n-cube) networks, the scheme is called E-cube routing.

- Dimension-order routing requires the selection of successive channels to follow a specific order based on the dimensions of a multidimensional network.
- In the case of a two-dimensional mesh network,
- The scheme is called X-Y routing

3. X-Y Routing on a 2D Mesh

There are four possible X-Y routing patterns corresponding to the east-north, east-south, west—north, and west-south paths chosen.


N
W
i,j
E
S
0,7
1,7
2,7
3,7
4,7
5,7
6,7
7,7
0,6
1,6
2,6
3,6
4,6
5,6
6,6
7,6
0,5
1,5
2,5
3,5
4,5
5,5
6,5
7,5
0,4
1,4
2,4
3,4
4,4
5,4
6,4
7,4
0,3
1,3
2,3
3,3
4,3
5,3
6,3
7,3
0,2
1,2
2,2
3,2
4,2
5,2
6,2
7,2
0,1
1,1
2,1
3,1
4,1
5,1
6,1
7,1
0,0
1,0
2,0
3,0
4,0
5,0
6,0
7,0
y
x Four (source; destination) pairs: (2,1;7,6), (0,7;4,2), (5,4;2,0), (6,3;1,5)

E-cube Routing on Hypercube

- Consider an n-cube with N=2^n
- Each node b is binary-coded as

$$b=b_{n-1}b_{n-2}\ldots\ldots b_1 b_0$$

- source node is s = $s_{n-1}\ldots.s_1 s_0$
- destination node is d = dn-1 . . . d1d0.
- determine a route from s to if with a minimum number of steps.
- Let v= $v_{n-1}\ldots\ldots v_1 v_0$ be any node along the route.

Algorithm

i. Compute the direction bit $r_i = s_{i-1} \oplus d_{i-1}$ for all n dimensions (i= 1, ..., n). Start the following with dimension i=1 and v=s .

ii. Route from the current node v to the next node v $\oplus$ 2^{i-1} if r_i=1. Skip this step if r_i, = 0.

iii. Move to dimension i+ 1 (i.e. i<— i+ 1). If i<=n, goto step 2,else done.

E-cube routing on a four-dimensional hypercube

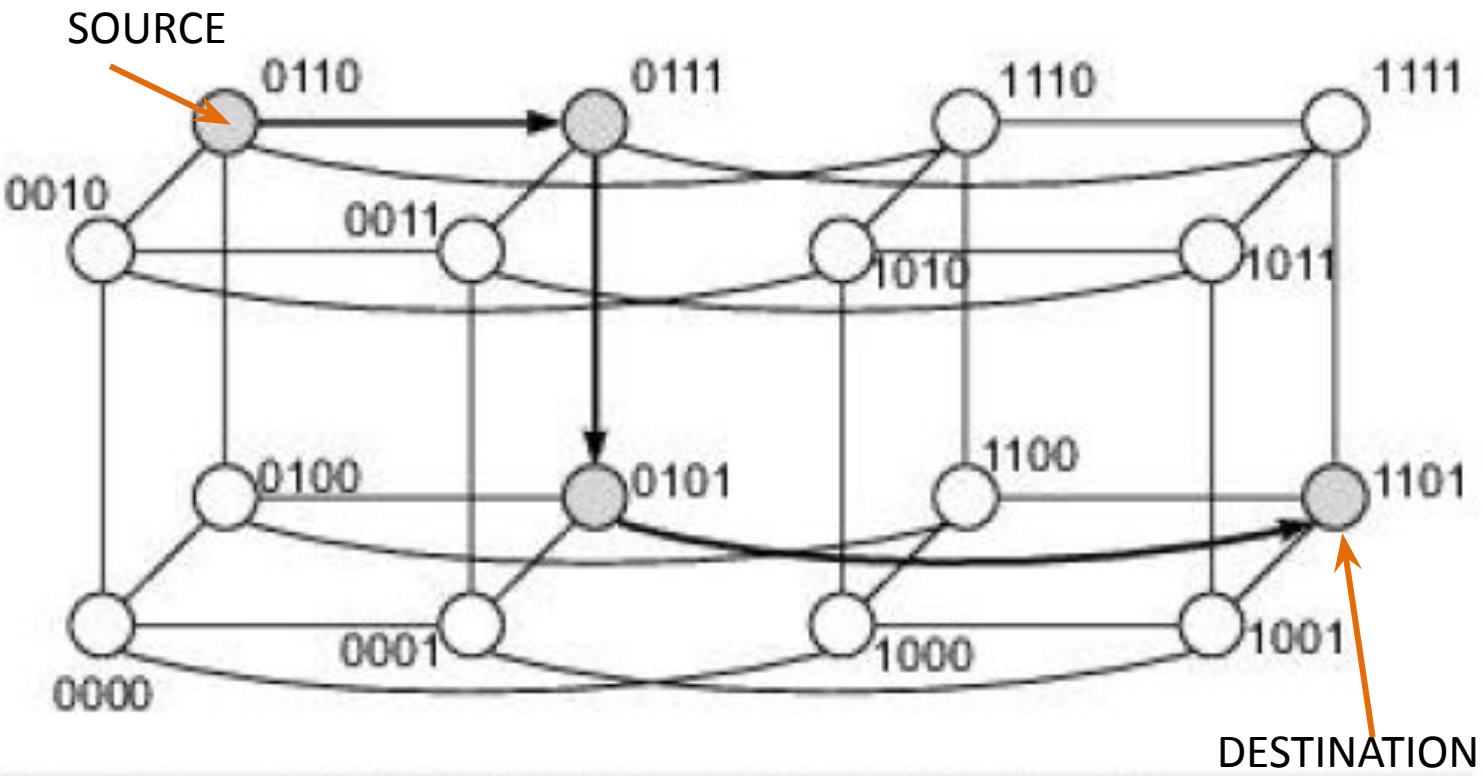


Fig. 7.34 E-cube routing on a hypercube computer with 16 nodes

ALGORITHM

- Determine a route from s to d with a minimum number of steps.
- source node is **S = s3s2s1s0 = 0110**
- destination node is **D = d3d2d1d0 = 1101**
- **ri= si-1 xor di-1**
- **vi = vi-1 xor 2^{i-1}**
- Start from **S = 0110**
- r1= s0 xor d0 = 0 xor 1 = 1 (so take node in that direction)
- v1= v0 xor 2^{1-1} = 0110 xor 0001 = **0111 (v1)**
- r2 = s1 xor d1 = 1 xor 0 = 1 (so take node in that direction)
- v2= v1 xor 2^{2-1} = 0111 xor 0010 = **0101 (v2)**
- r3 = s2 xor d2 = 1 xor 1 = 0 (so don't take node in that direction)
- v3 not taken
- r4 = s3 xor d3 = 0 xor 1 = 1 (so take node in that direction)
- v4= v2 xor 2^{4-1} = 0101 xor 1000 = **1101 (v4)**
- **0110 □ 0111 □0101 □1101**

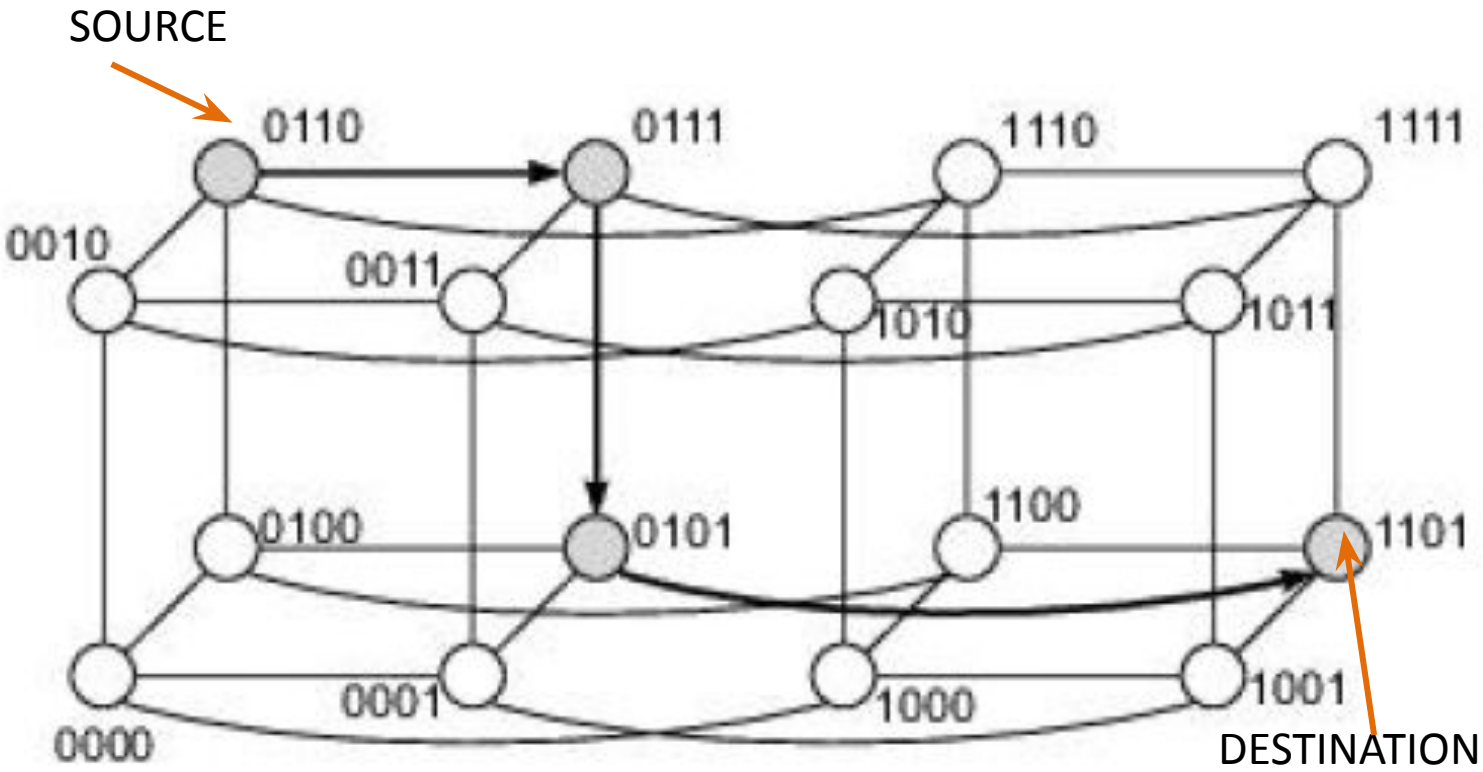
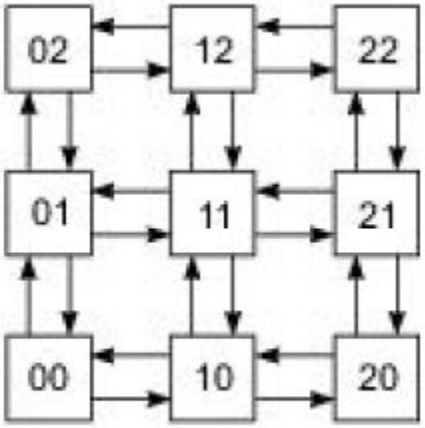


Fig. 7.34 E-cube routing on a hypercube computer with 16 nodes

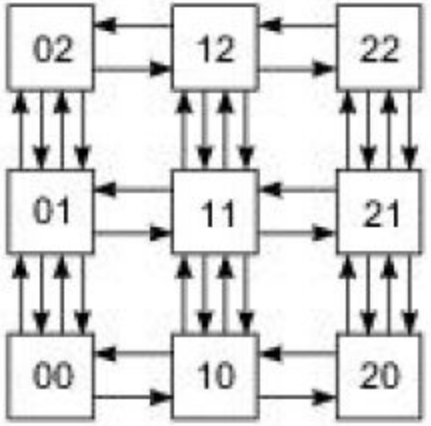
0110 □ 0111 □0101 □1101

Adaptive Routing

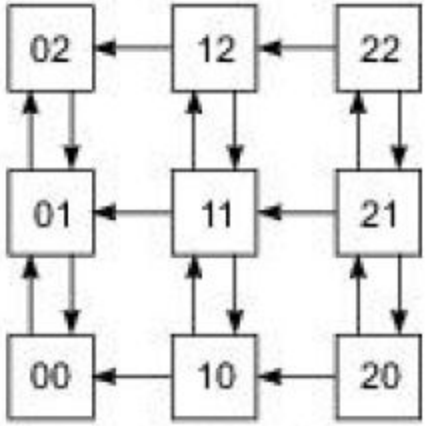
main purpose-achieve efficiency and avoid deadlock.



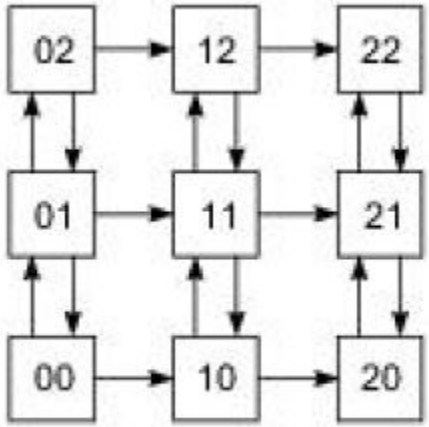
(a) Original mesh without virtual channel



(b) Two pairs of virtual channels in Y-dimension



(c) For a westbound message



(d) For an eastbound message

Multicast Routing Algorithms

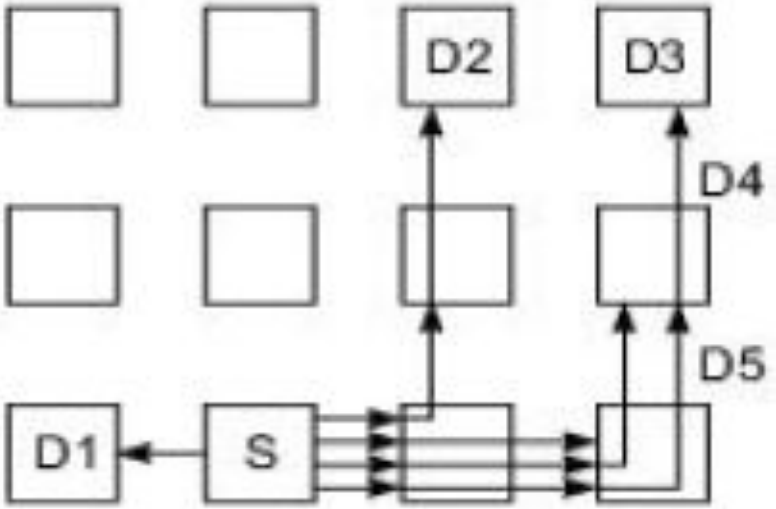
1. Communication Pattern
2. Routing Efficiency
3. Virtual Network
4. Network Partitioning

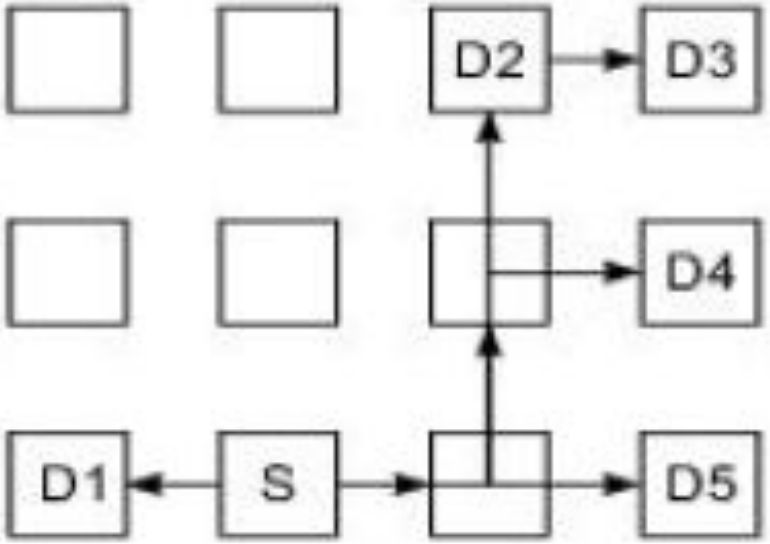
Communication Pattern

Four types: one-to-one unicast pattern , multicast pattern, broadcast pattern

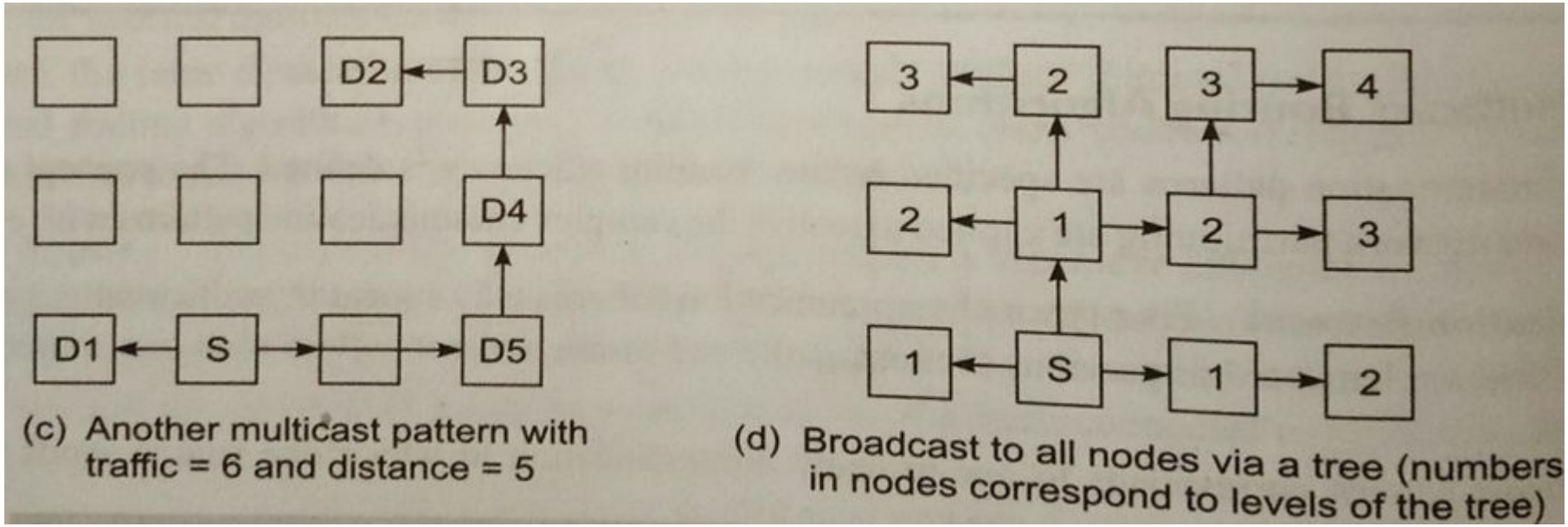
- A multicast pattern corresponds to one-to-many communication in which one source sends the same
- message to multiple destinations.
- A broadcast pattern corresponds to the case of one-to-all communication
- The most generalized pattern is the many-to-many communication

multicast pattern





multicast pattern



(c) Another multicast pattern with traffic = 6 and distance = 5

(d) Broadcast to all nodes via a tree (numbers in nodes correspond to levels of the tree)

Routing Efficiency

2. Two commonly used:
    1. channel bandwidth
    2. communication latency

channel bandwidth

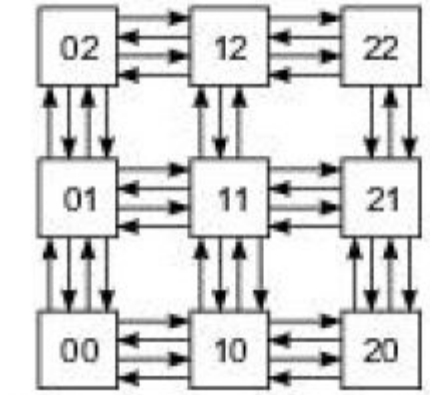
At any time instant (or during any time period) indicates the effective data transmission rate achieved to deliver the messages.

communication latency

The latency is indicated by the packet transmission delay involved.

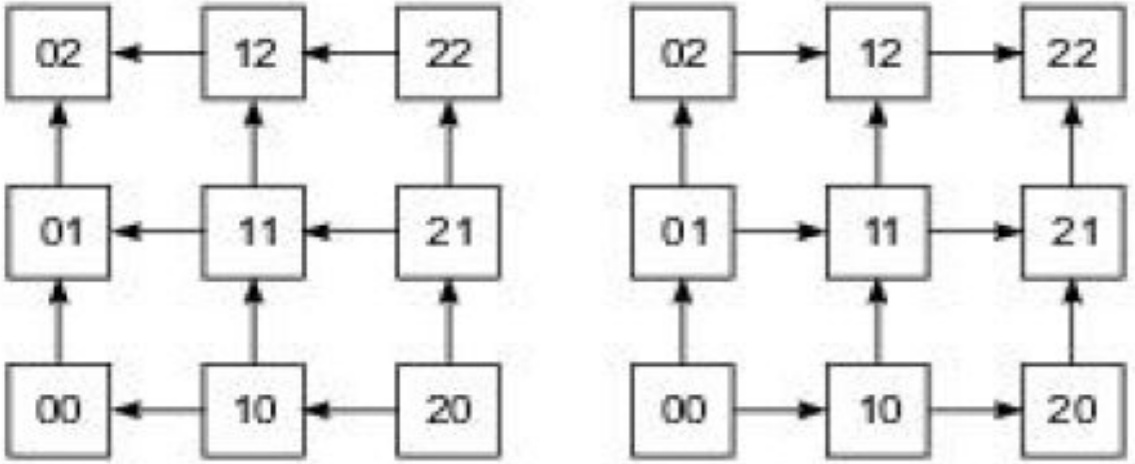
Virtual Network

These virtual channels can be used to generate four possible virtual networks



(a) A dual-channel 3 × 3 mesh

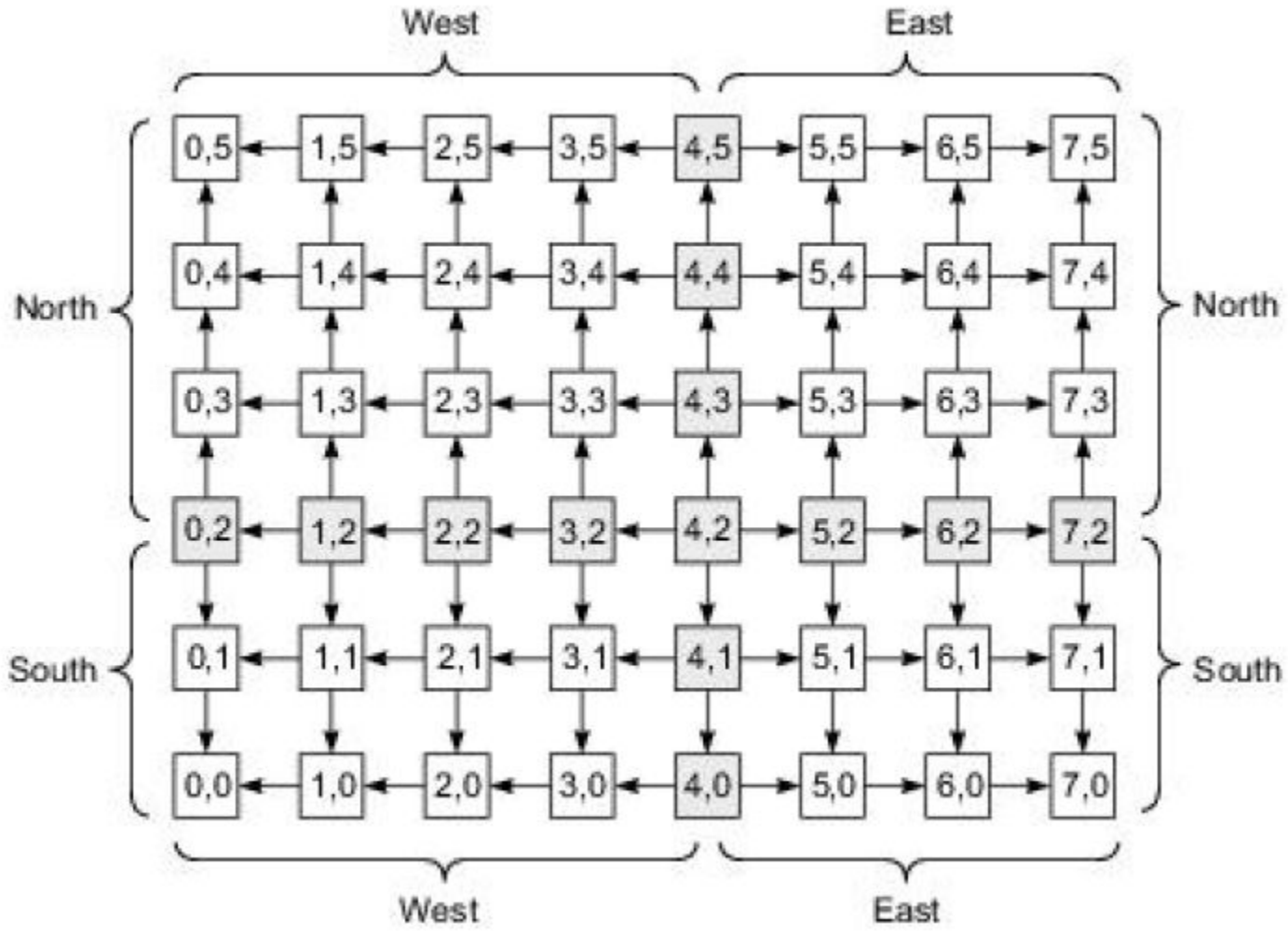
- For west-north traffic the virtual network



Network Partitioning

The concept of virtual networks is partitioning of a given physical network into logical subnet works for multicast communications.

- Partitioning of a 6 x 8 mesh into four subnets for a multicast from source node

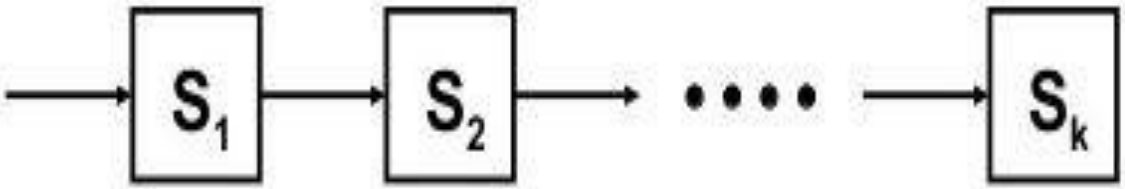

West
East
North
North
South
South
0,5
1,5
2,5
3,5
4,5
5,5
6,5
7,5
0,4
1,4
2,4
3,4
4,4
5,4
6,4
7,4
0,3
1,3
2,3
3,3
4,3
5,3
6,3
7,3
0,2
1,2
2,2
3,2
4,2
5,2
6,2
7,2
0,1
1,1
2,1
3,1
4,1
5,1
6,1
7,1
0,0
1,0
2,0
3,0
4,0
5,0
6,0
7,0
West
East

Pipelining and Superscalar Techniques

1. Linear pipeline processors
   a) Asynchronous and Synchronous Models
   b) Clocking and Timing Control
   c) Speedup, Efficiency, and Throughput
2. Nonlinear pipeline processors
   a) Reservation and Latency Analysis
   b) Collision-Free Scheduling
   c) Pipeline Schedule Optimization

Linear Pipeline processor

- linear pipeline processor is a cascade of processing stages which are linearly connected to perform a fixed function on a stream of data flowing from one end to the other.
- A linear pipeline processor is constructed with k processing stages.

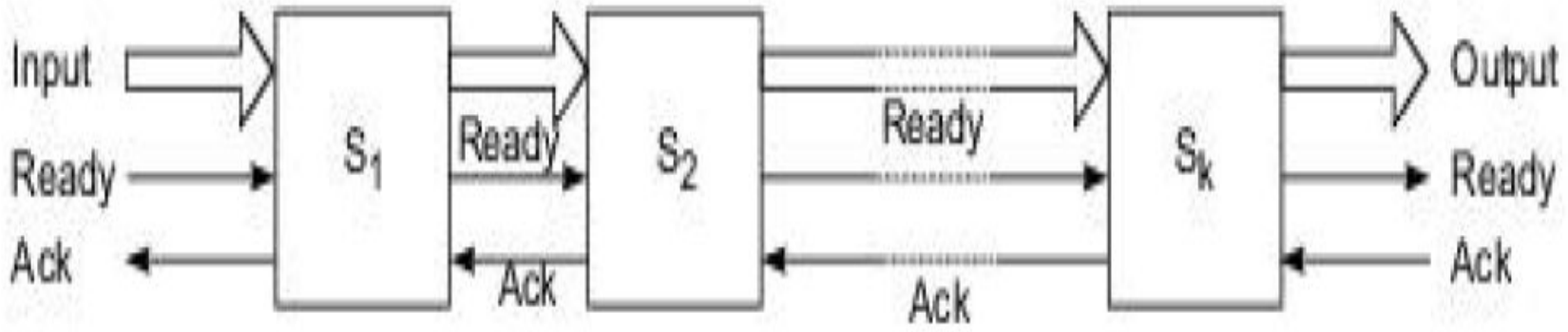


Linear pipeline processors

Asynchronous Models

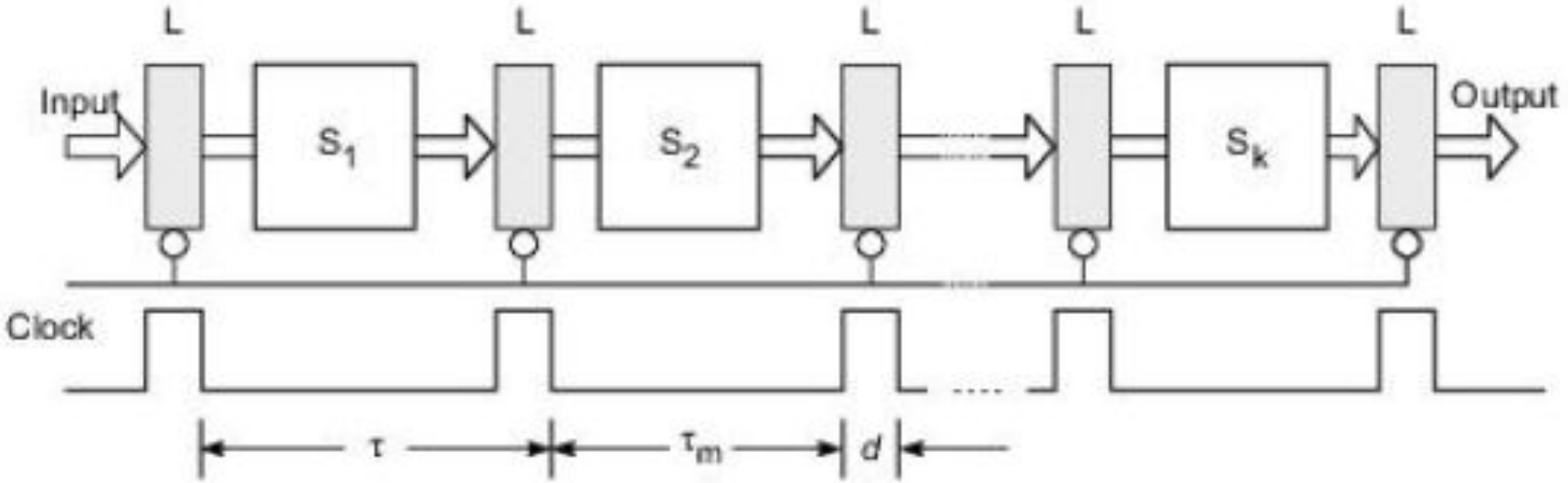
- Asynchronous pipelines may have a variable throughput rate
- Different amounts of delay may be experienced in different stages.

Asynchronous Models



(a) An asynchronous pipeline model

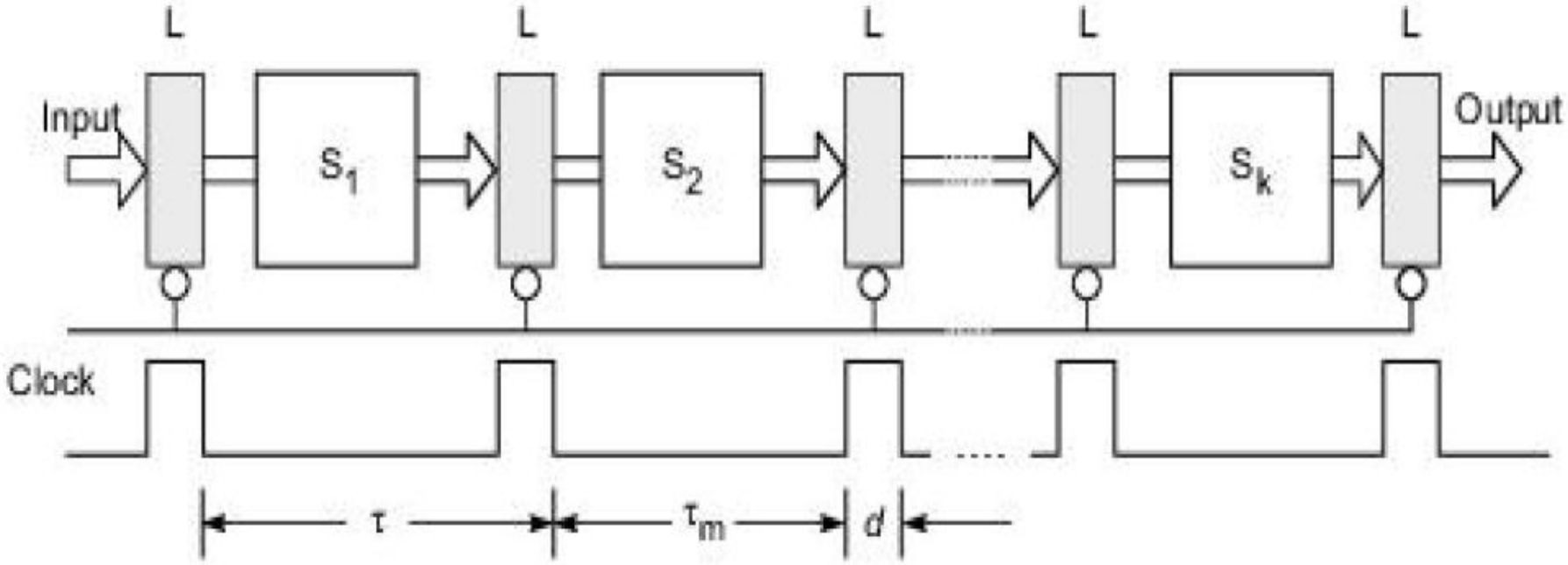
synchronous Models



Synchronous Models

- Clocked latches are used to interface between stages
- isolate inputs from outputs
- Upon the arrival of a clock pulse, all latches transfer data to the next stage simultaneously.

(a) An asynchronous pipeline model



(b) A synchronous pipeline model

τ = Clock period

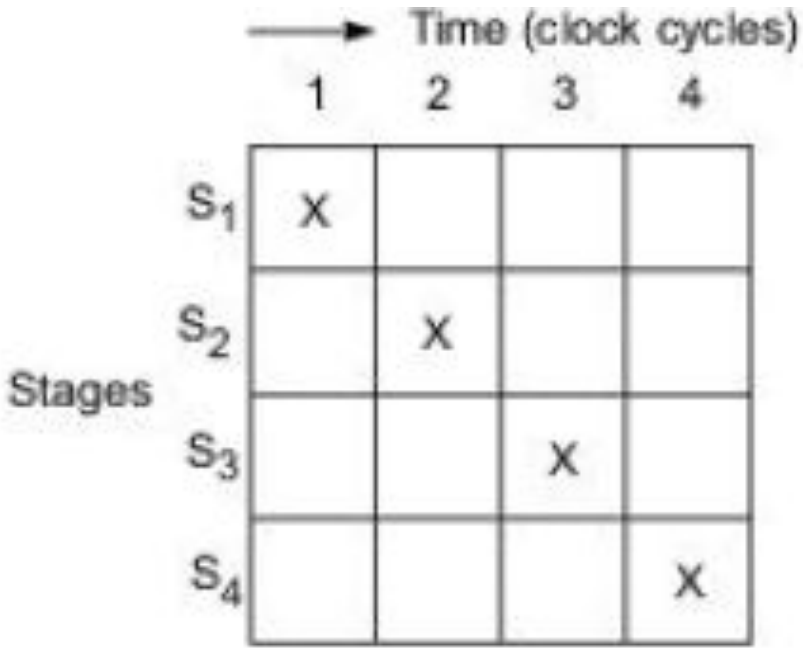
τ_m = Maximum stage delay

d = Latch delay

Synchronous Models

- equal delays in all stages
- These delays determine the clock period and thus the speed of the pipeline
- synchronous pipeline is specified by a reservation table

Reservation Table



- Successive tasks are one per cycle to enter the pipeline.
- Once the pipeline is filled up, one result emerges from the pipeline
- This throughput is sustained only if the successive tasks are independent

- Clock Period or Clock cycle (T)

  = max $\{t_i\}_1^k$ + d = τ_m + d

- ti: time delay of the circuitry in stage Si.
- d: time delay of latch.
- τ_m : Maximum stage delay

$$\tau = \max_i \{\tau_i\}_1^k + d = \tau_m + d$$

Clocking and Timing Control

- Pipeline Frequency (f)
- Inverse of clock period
- **f = 1/T**

Problem

4 stages

S1=60ns, s2=50ns, s3=90 ns and s4= 80ns,

d=10ns

Find clock time of this pipeline.

- The pipeline frequency is defined as the inverse of the clock period:

  $f = \frac{1}{\tau}$

- F represents the maximum throughput of the

Pipeline

- Depending on the initiation rate of successive tasks entering the pipeline
- The actual throughput of the pipeline may be lower than f

Clock Skewing

- The same clock pulse may arrive at different stages with a time offset of s.

Clocking and Timing Control

Speedup, Efficiency, and Throughput

- a linear pipeline of k stages can process n tasks in k + (n — 1) clock cycles.
- total time $T_k = [k + (n-1)]\tau$
- τ is the clock period

Clocking and Timing Control

- Speedup Factor

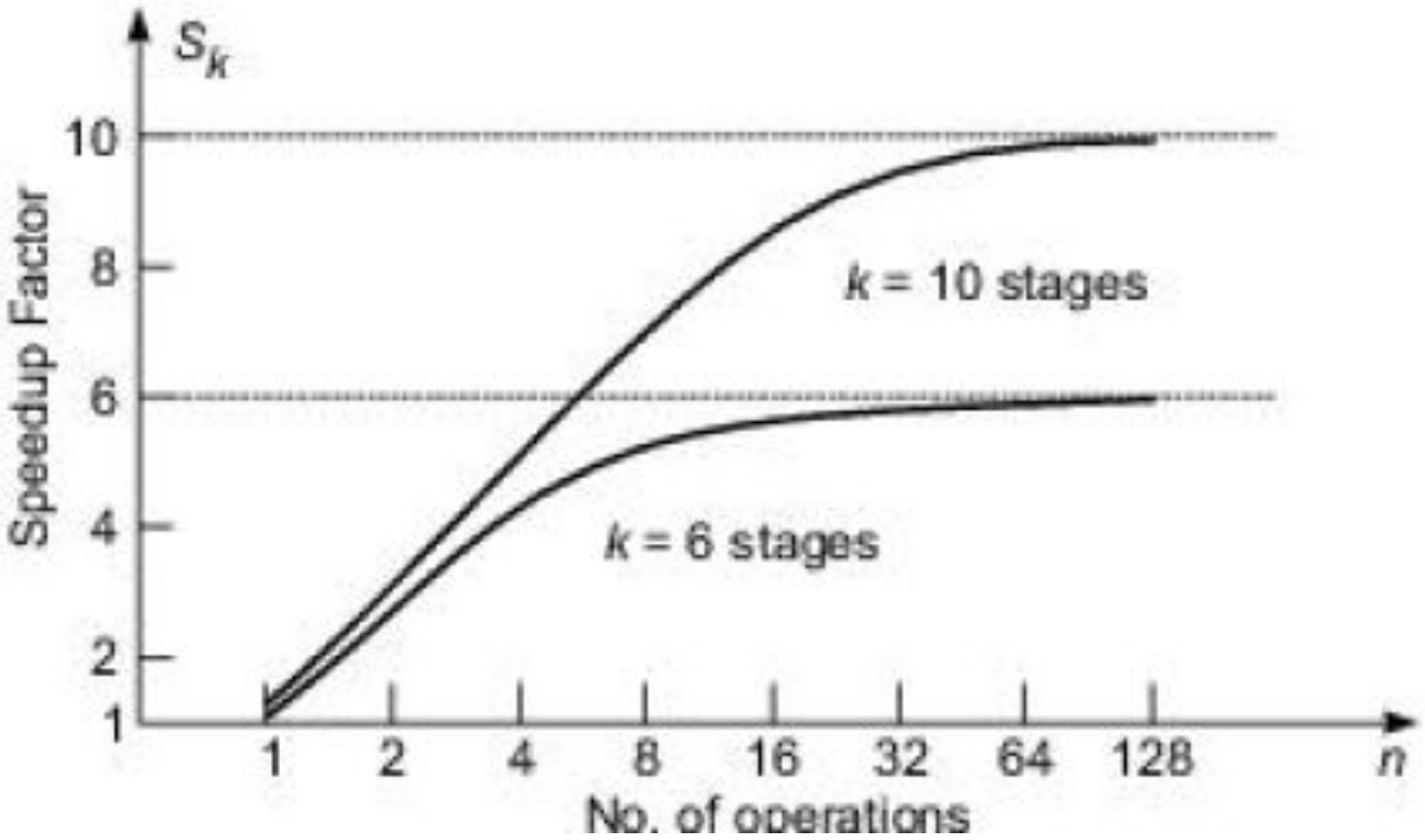
The speedup factor of a k-stage pipeline over an equivalent non pipelined processor is defined as

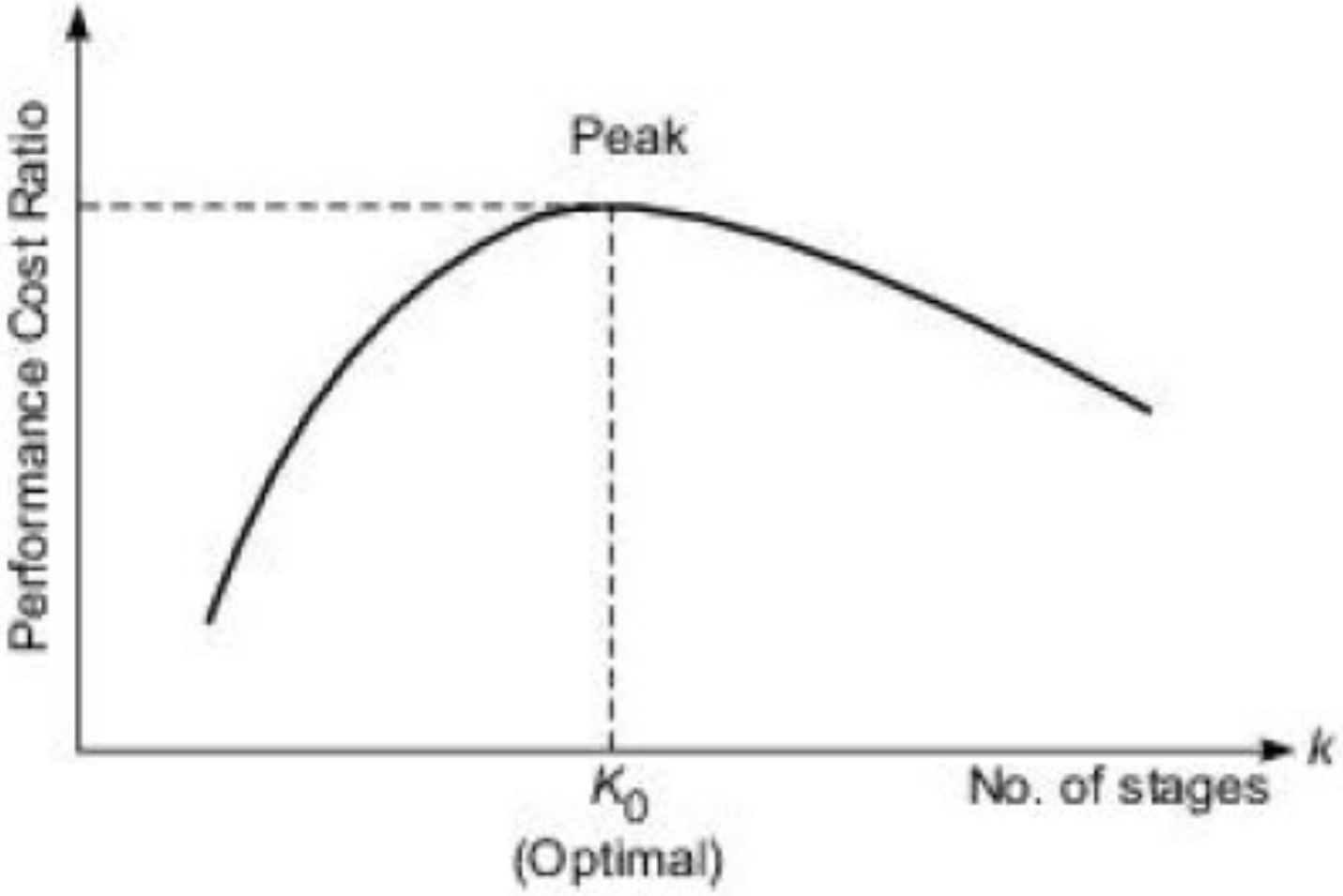
$$S_k = \frac{T_1}{T_k} = \frac{nk\tau}{k\tau + (n-1)\tau} = \frac{nk}{k + (n-1)}$$

Clocking and Timing Control

- Optimal Number of Stage

  Number of pipeline stages should be able to maximize the performance/cost ratio for the target processing load


S_k
10
8
6
4
2
1
Speedup Factor
k = 10 stages
k = 6 stages
1
2
4
8
16
32
64
128
n
No. of operations


Performance Cost Ratio
Peak
k
No. of stages
K_0
(Optimal)

Efficiency and Throughput

- Efficiency

The efficiency E_k of a linear It-stage pipeline is defined as

$$E_k = \frac{S_k}{k} = \frac{n}{k + (n - 1)}$$

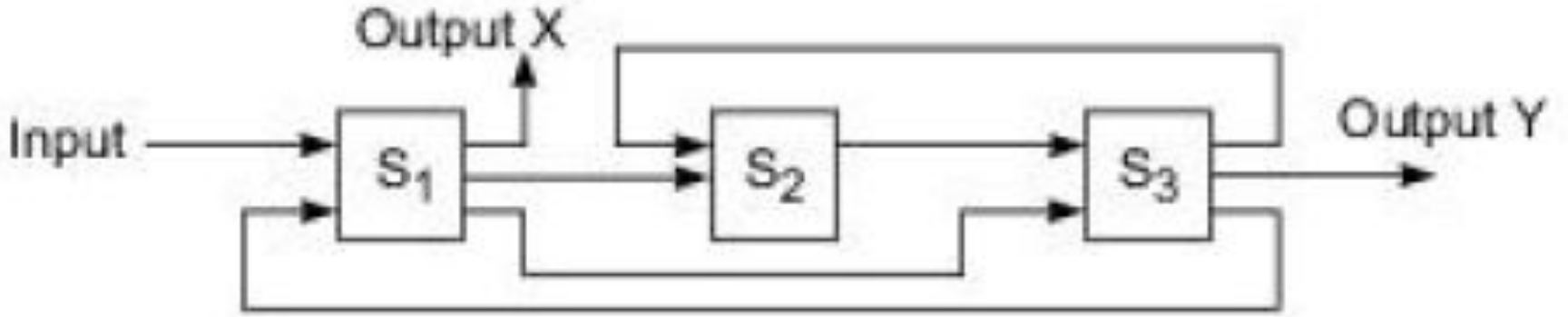
throughput

$$H_k = \frac{n}{[k + (n - 1)]\tau} = \frac{nf}{k + (n - 1)}$$

Nonlinear pipeline processors

- A dynamic pipeline can be reconfigured to perform variable functions at different times.
- It allows feed-forward and feedback connections in addition to the streamline connection.

Reservation and Latency Analysis



(a) A three-stage pipeline

- interconnection structures and data flow patterns with either feed back or feed forward connections.
- A sample pipeline is given with both feedback and feed forward connections.

•Three stages are used, S1, S2 and S3.

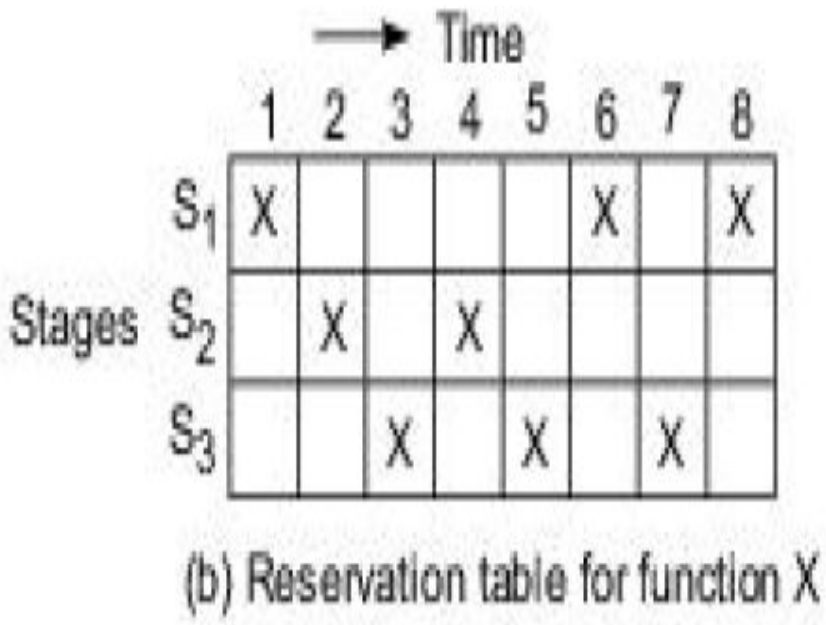
•Feed forward Connection – Connects Si to Sj, Feed back connection - Connects Si to Sj, j<=i.

Reservation Table

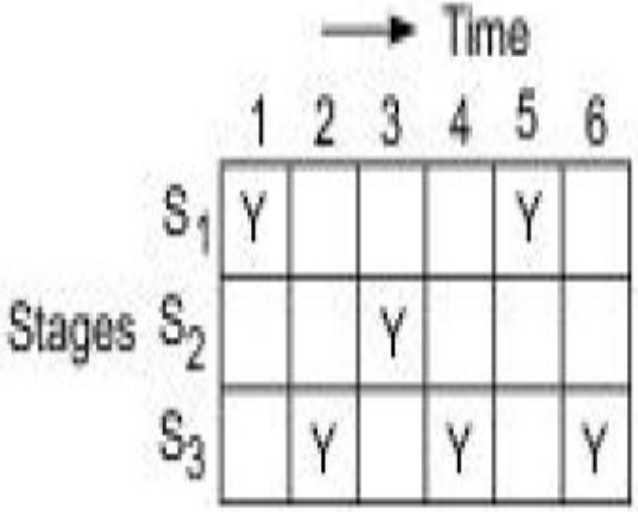
- Each functional evaluation can be represented using a **space-time diagram** of a pipeline called **Reservation Table (RT).**
- X axis – time units Y axis – stages

Reservation Table

•



(b) Reservation table for function X



(c) Reservation table for function Y

Reservation Table

For first sequence Sa, Sb, Sc, Sb, Sc, Sa called function A, we have

Time →

Stage ↓

	1	2	3	4	5	6
Sa	A					A
Sb		A		A		
Sc			A		A	

Reservation Table

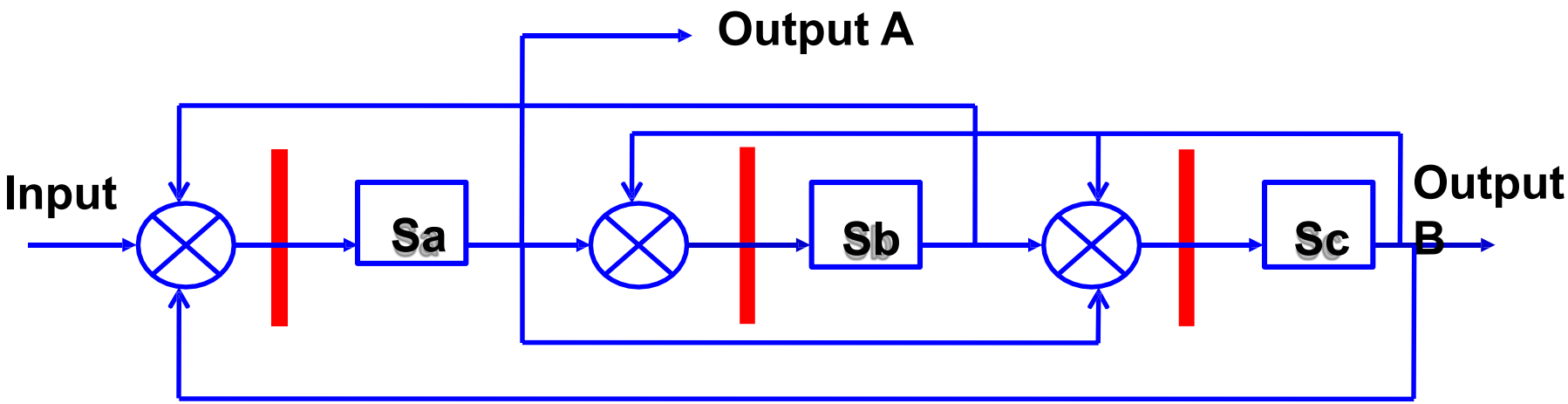
For second sequence Sa, Sc , Sb, Sa, Sb, Sc called function B, we have

Time →

Stage ↓

	1	2	3	4	5	6
Sa	B			B		
Sb			B		B	
Sc		B				B

3-Stage Non-Linear Pipeline



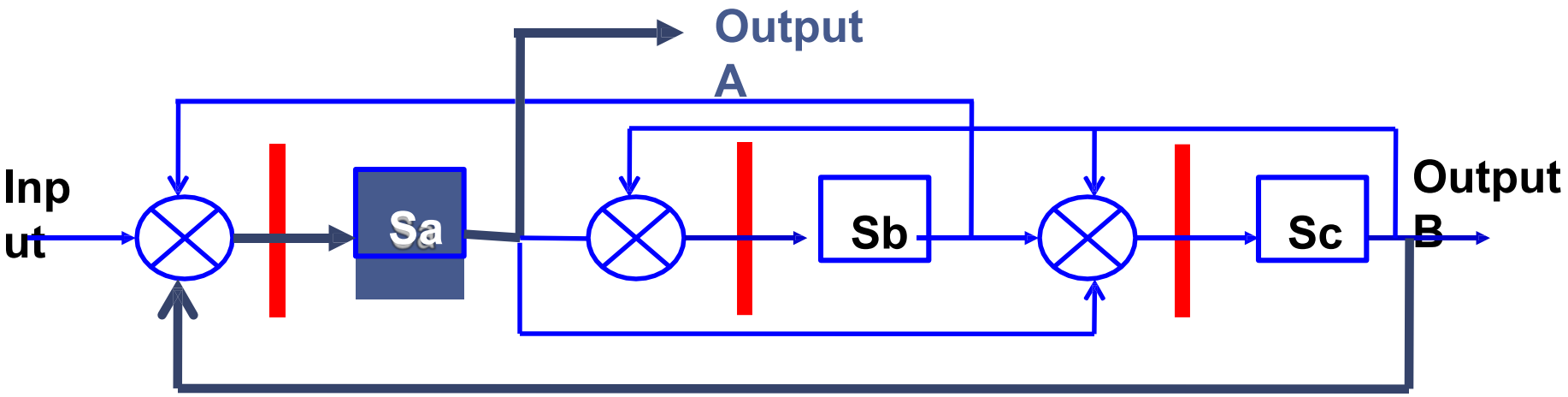
Reservation Table

Time →

Stage ↓

	1	2	3	4	5	6
Sa						
Sb						
Sc						

Function A-3-Stage Pipeline : Sa, Sb, Sc, Sb, Sc, Sa



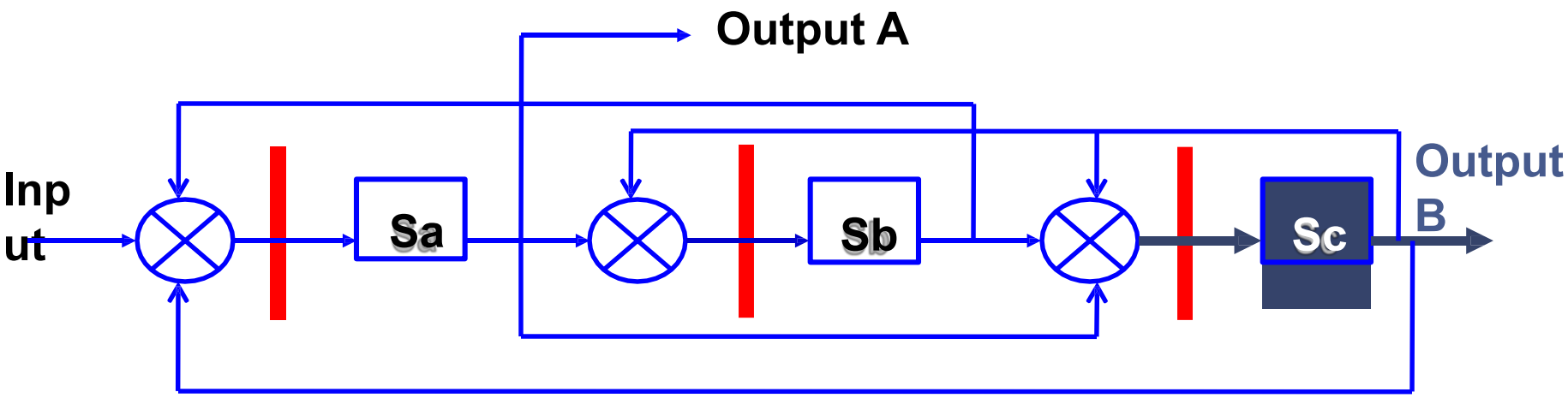
Reservation Table

Time →

↓ Stage

	1	2	3	4	5	6
Sa	A					A
Sb		A		A		
Sc			A		A	

Function B -3-Stage Pipeline : Sa, Sc, Sb, Sa, Sb, Sc



Reservation Table

Time →

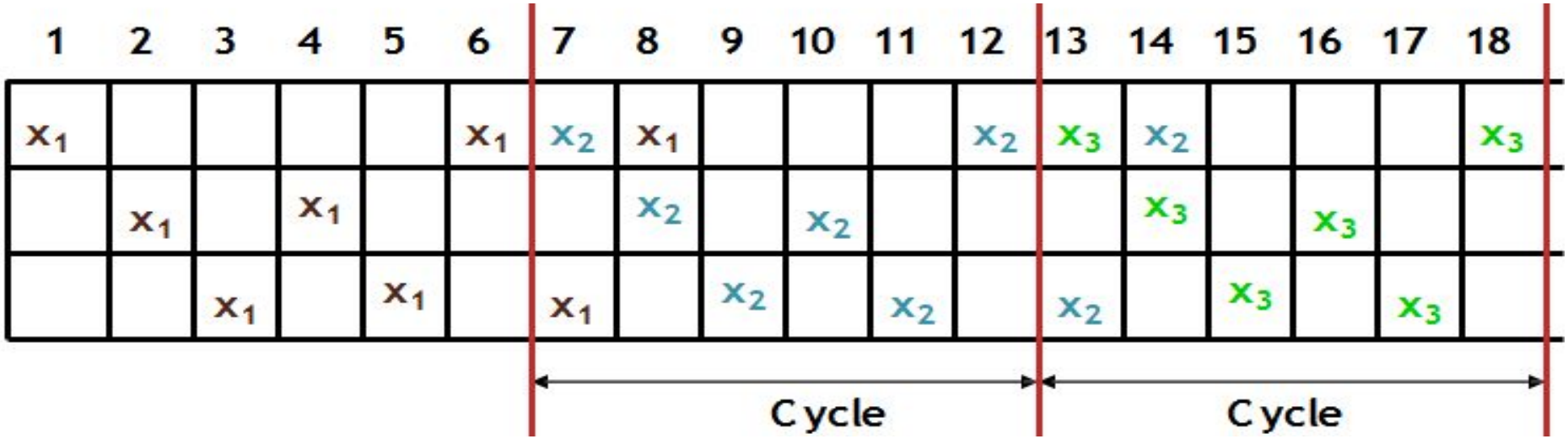
↓ Stage

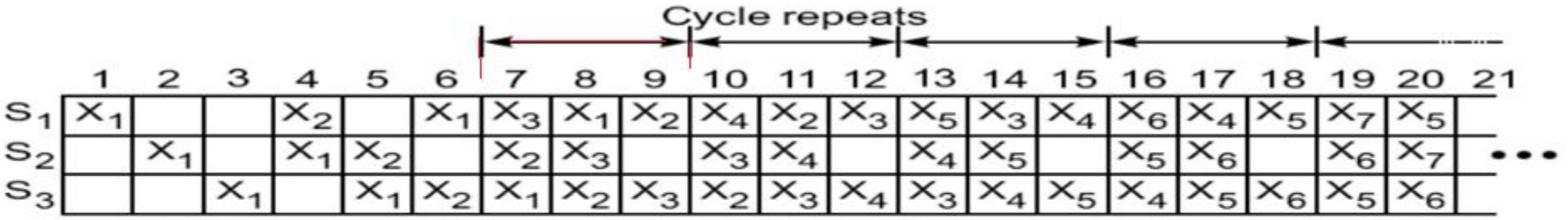
	1	2	3	4	5	6
Sa	B			B		
Sb			B		B	
Sc		B				B

Latency Analysis

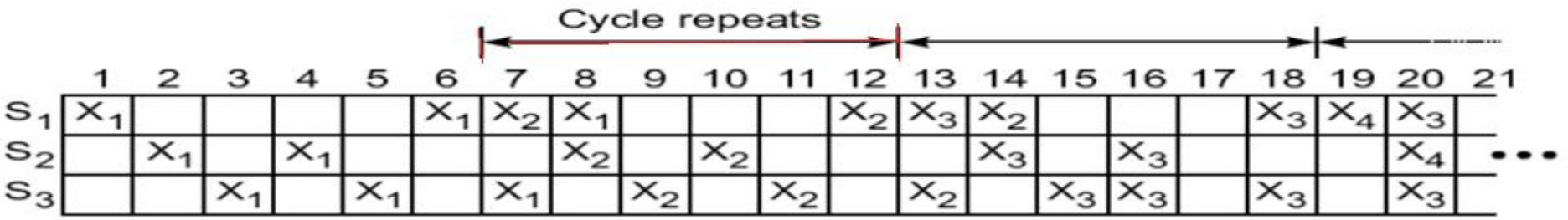
- **Latency:** The number of time units (clock cycles) between two initiations of a pipeline is the latency between them
- **Collision:** two or more initiations to use the same pipeline stage at the same time
- **Forbidden Latencies** □ latencies that will cause collision
- **Permissible Latencies**□ latencies that will not cause collision

A ***latency cycle*** - a latency sequence which repeats the same sub-sequence (cycle) indefinitely.





(b) Latency cycle (3) = 3, 3, 3, 3, ..., with an average latency of 3



(c) Latency cycle (6) = 6, 6, 6, 6, ..., with an average latency of 6

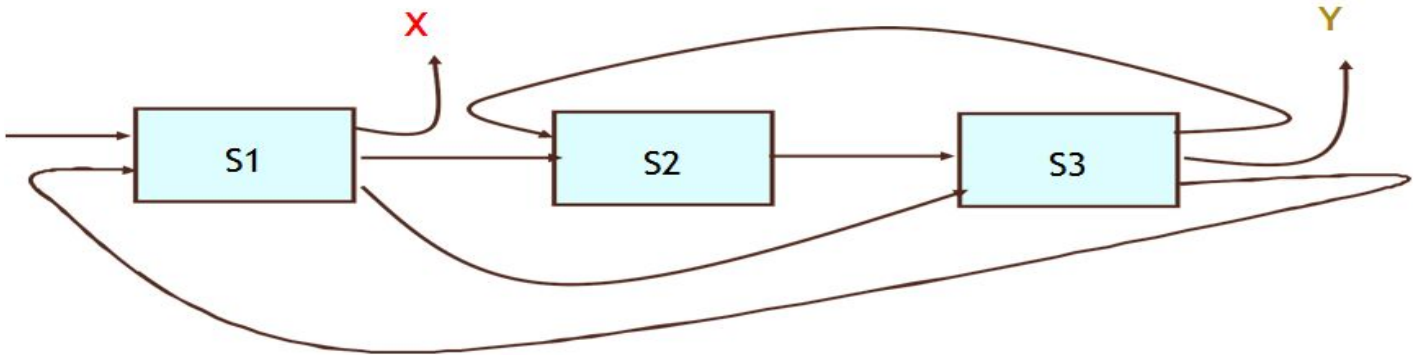
Three valid latency cycles for the evaluation of function X

Collision-Free Scheduling

- shortest average latency between initiations without causing collisions

1. collision vectors
2. state diagrams
3. Simple cycle
4. Greedy cycle
5. Minimal average latency {MAL}

Example -Reservation Tables for X & Y

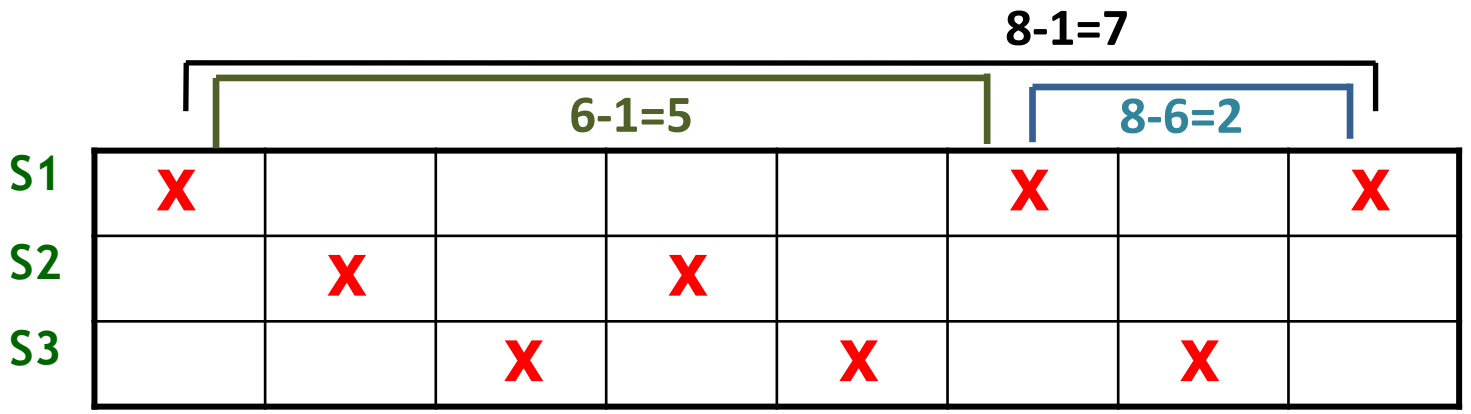


S1	X					X		X
S2		X		X				
S3			X		X		X	

Forbidden Latencies

- To detect the forbidden latencies, check the distance between two checkmarks in the same row of the reservation table.

Example -Reservation Table



Distances between X's in first row are 2, 5, 7
Distances between X's in second row is 2
Distances between X's in third row are 2, 4
So 2,4, 5, 7 are forbidden latencies

- Combined set of permissible and forbidden latencies.
- Forbidden Latencies: 2, 4, 5, 7
- Collision vector
  - C = (C_m, C_{m-1}, …, C_2, C_1), m <= n-1
  - n = number of column in reservation table
  - Ci = 1 if latency i is forbidden(causes a collision)
  - Ci = 0 if latency i is permissible.
- Collision Vector = 1 0 1 1 0 1 0

State Diagram

- State diagrams can be constructed to specify the permissible transitions among successive initiations.

- The collision vector, corresponding to the initial state of pipeline at time 1, is called the initial collision vector (ICV).

State diagram

1. Start with the **ICV (Initial Collision Vector)**
2. For each unprocessed state,

   For each bit i in the **CV_i which is 0**, do the following:

a. **Shift CV_i right by i bits**

b. **Drop i rightmost bits**

c. **Append zeros** to left

d. **Logically OR with ICV**

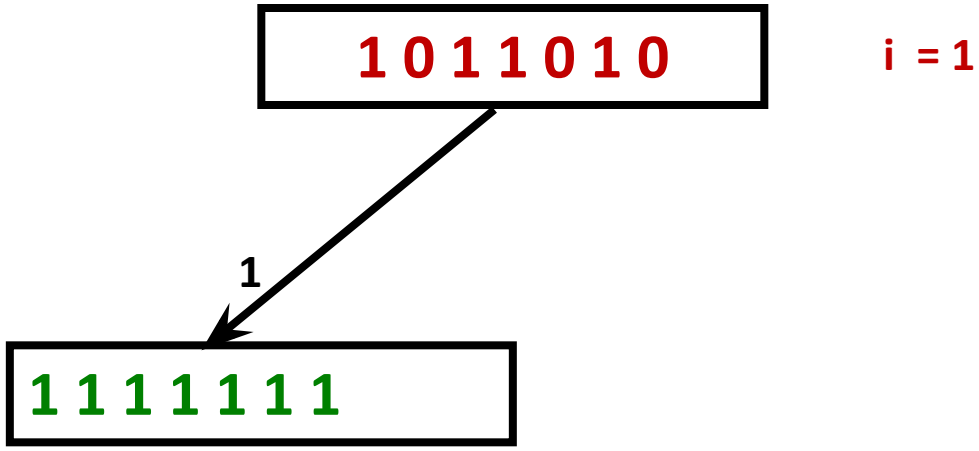
e. If step(d) results in a **new state then form a new node** for this state and **join it with node of CV_i by an arc** with a marking i.

- This shifting process needs to continue **until no more new states** can be generated.

State Diagram

1 0 1 1 0 1 0

State Diagram

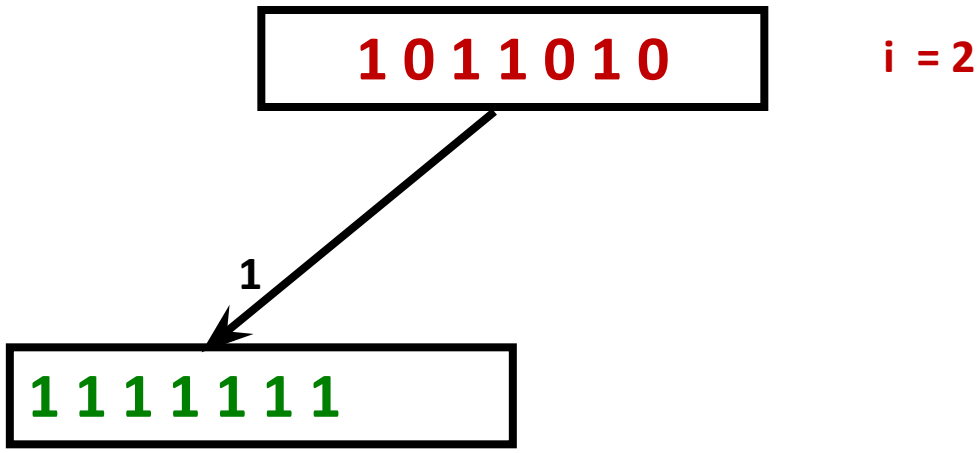


ICV – 1 0 1 1 0 1 0 OR

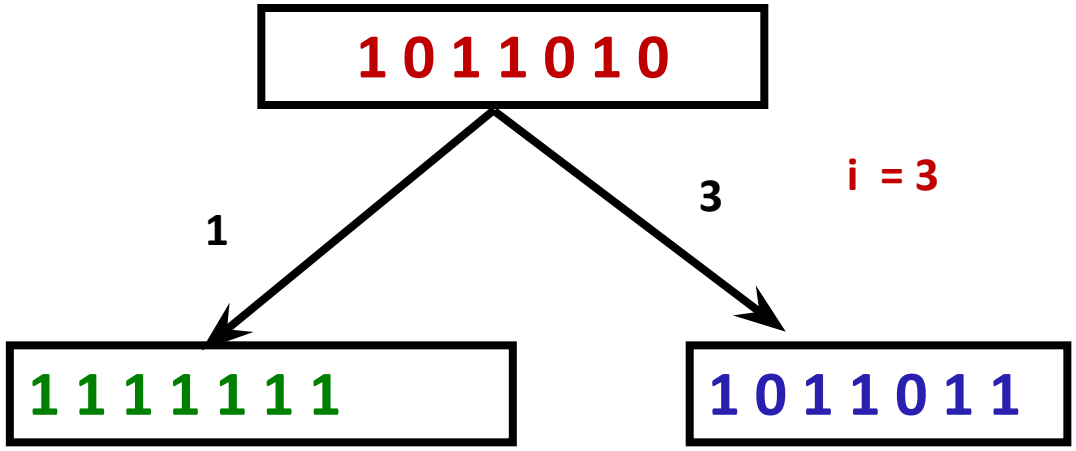
CV_i – 0 1 0 1 1 0 1 (shift right 1011010 by 1 bit)

CV* 1 1 1 1 1 1 1

State Diagram



State Diagram

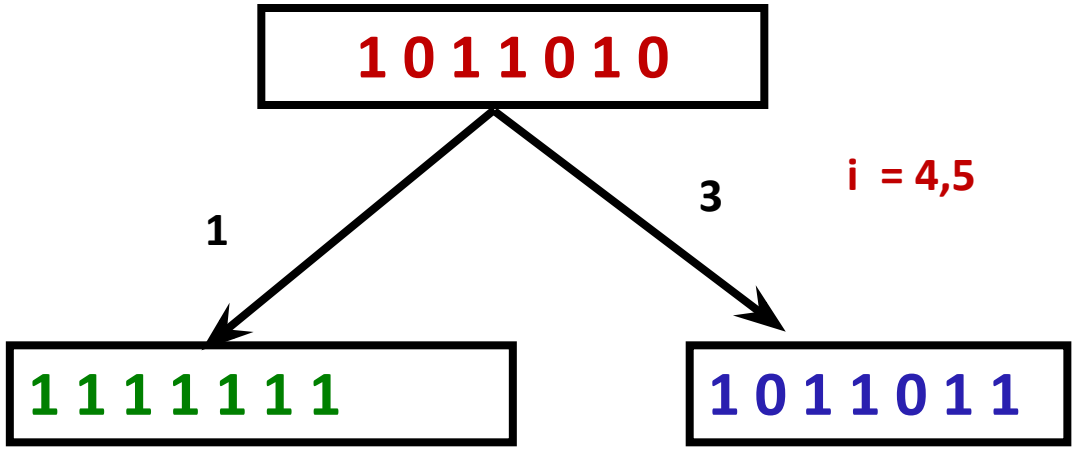


ICV – 1 0 1 1 0 1 0 OR

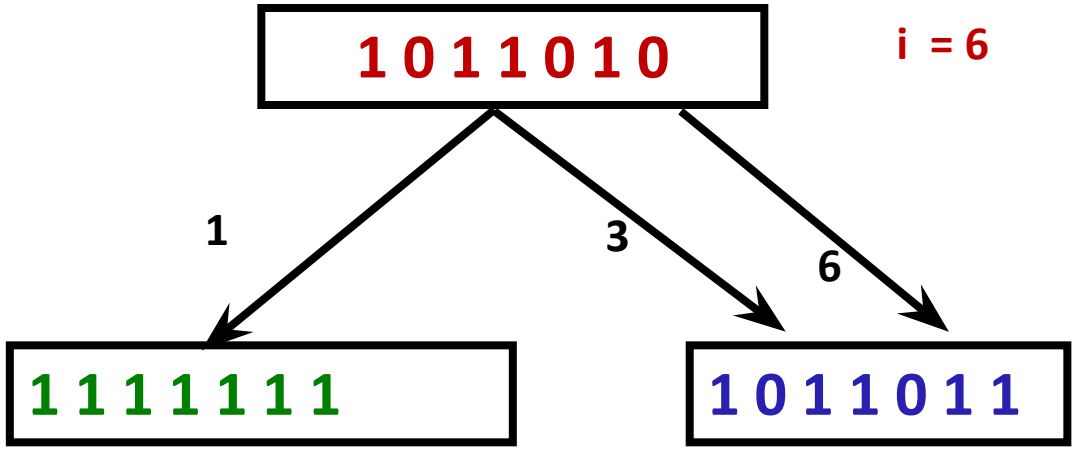
CV_i – 0 0 0 1 0 1 1 (shift right 1011010 by 3 bit)

CV* 1 0 1 1 0 1 1

State Diagram



State Diagram

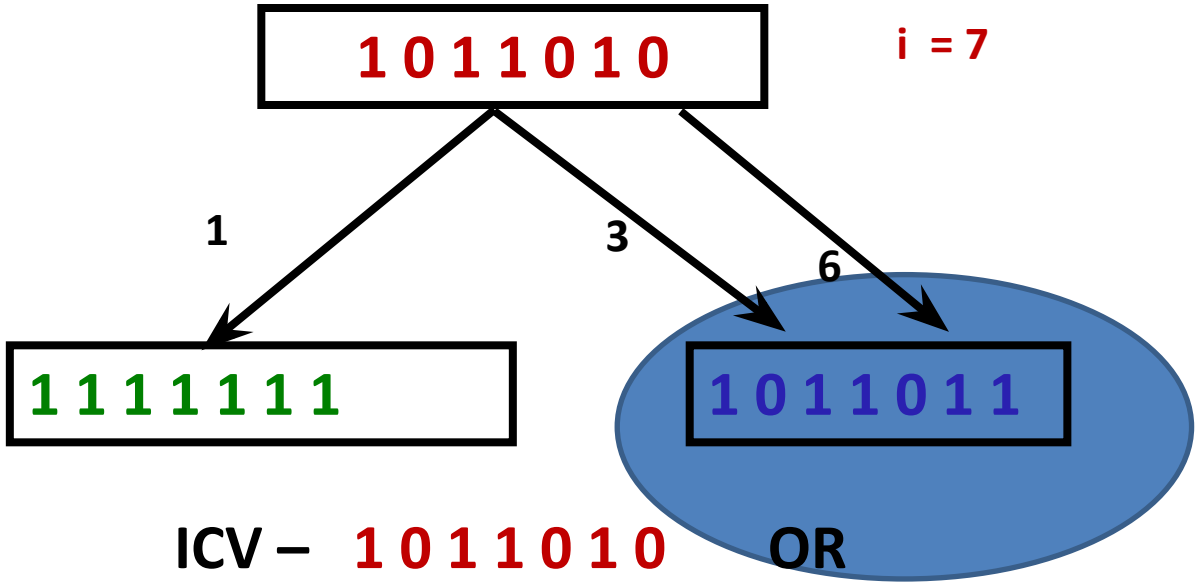


ICV – 1 0 1 1 0 1 0 OR

CV_i – 0 0 0 0 0 0 1 (shift right 1011010 by 6 bit)

CV* 1 0 1 1 0 1 1

State Diagram

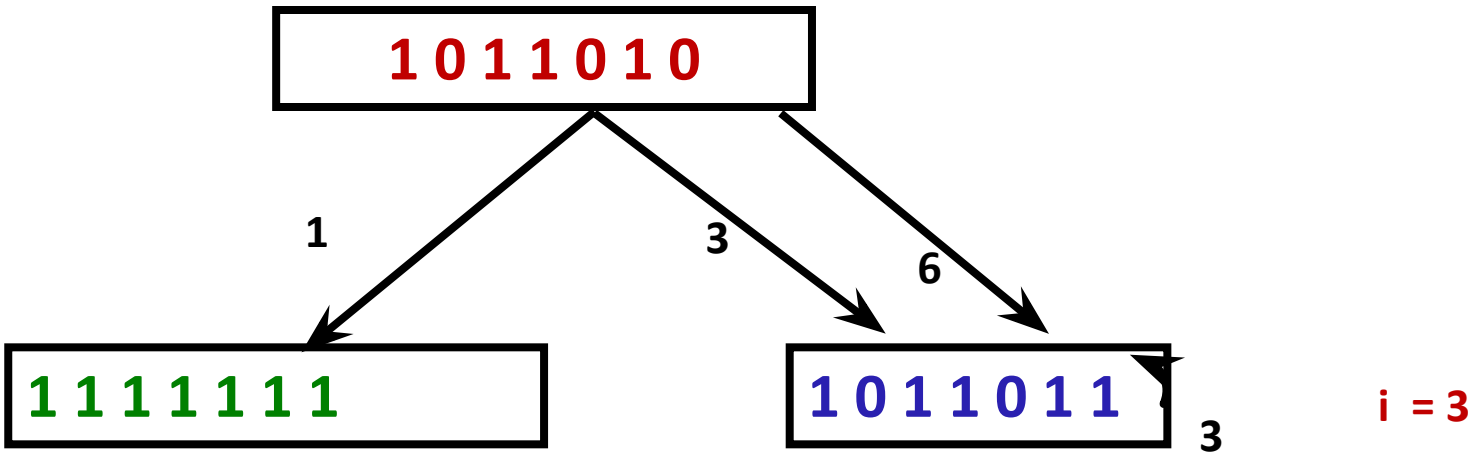


ICV – 1 0 1 1 0 1 0 OR

CV_i – 0 0 0 0 0 0 1 (shift right 1011010 by 6 bit)

CV* 1 0 1 1 0 1 1

State Diagram

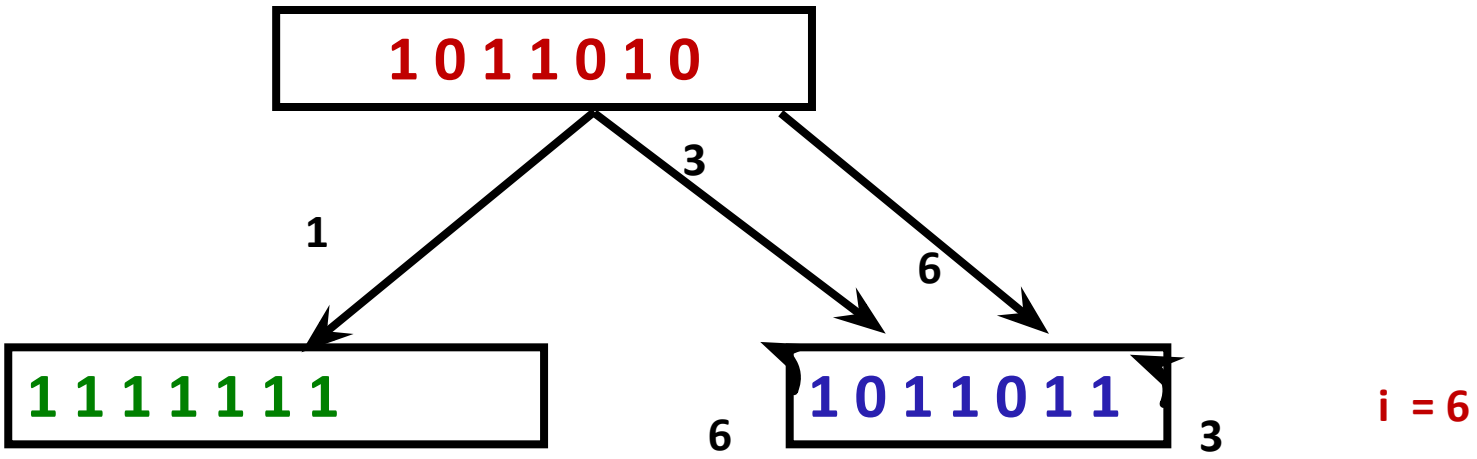


ICV – 1 0 1 1 0 1 0 OR

CV_i – 0 0 0 1 0 1 1 (shift right 1 0 1 1 0 1 1 by 3 bit)

CV* 1 0 1 1 0 1 1

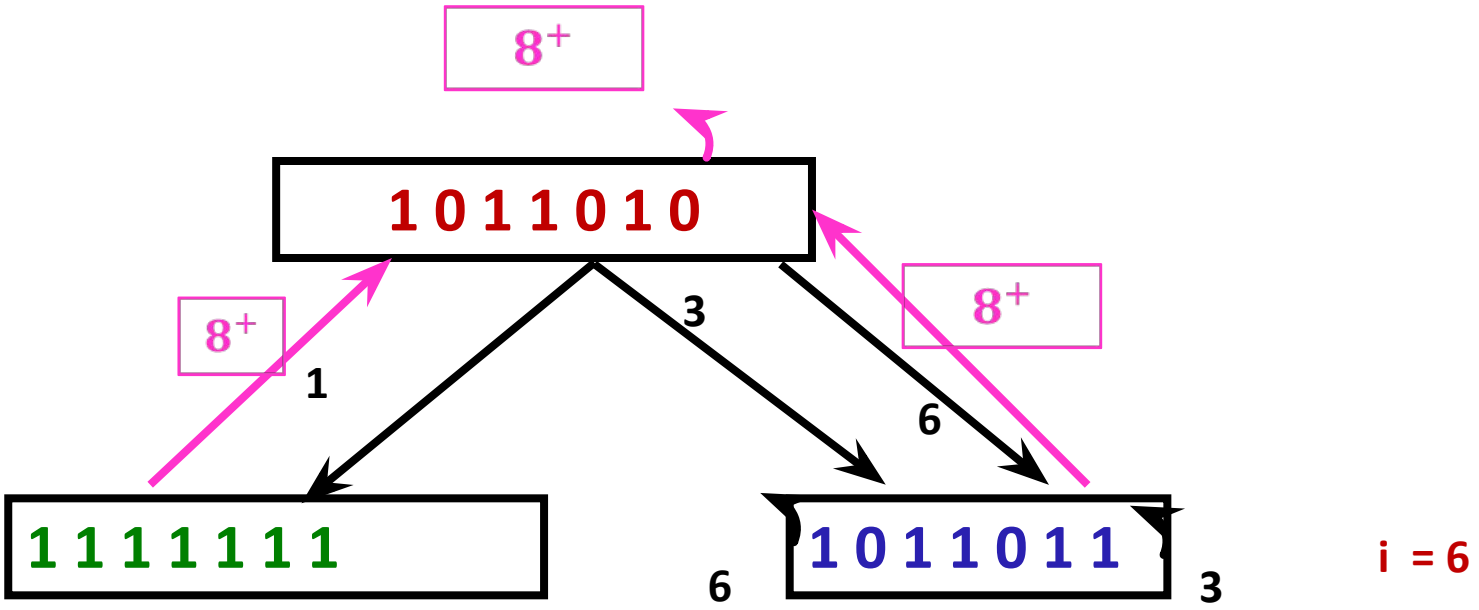
State Diagram



ICV – 1 0 1 1 0 1 0 **OR**

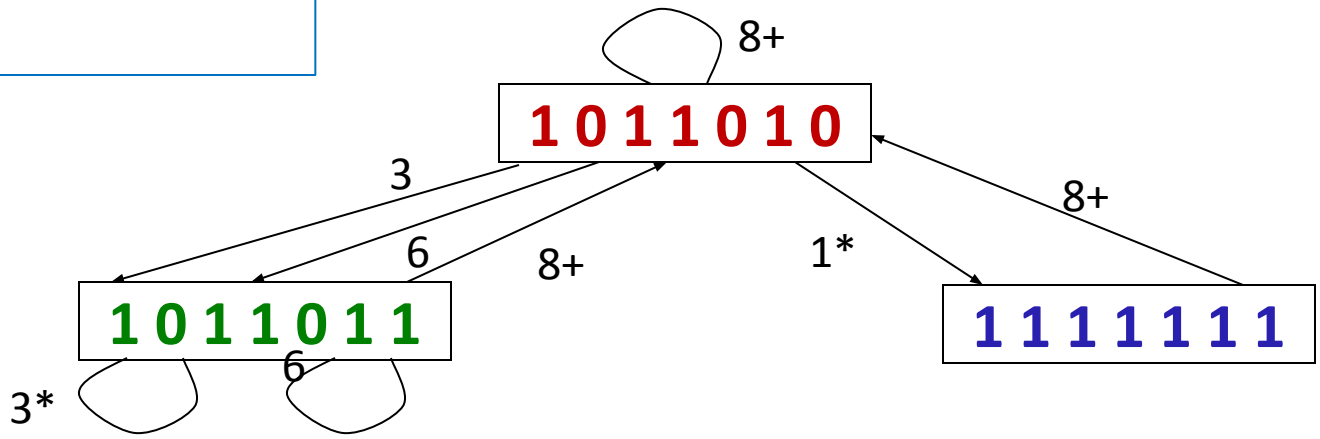
CV_i – 0 0 0 0 0 0 1 (shift right 1 0 1 1 0 1 1 by 6 bit)

CV* 1 0 1 1 0 1 1


8+
1 0 1 1 0 1 0
8+
1
3
6
8+
1 1 1 1 1 1 1
6
1 0 1 1 0 1 1
3
i = 6

Latency Cycles

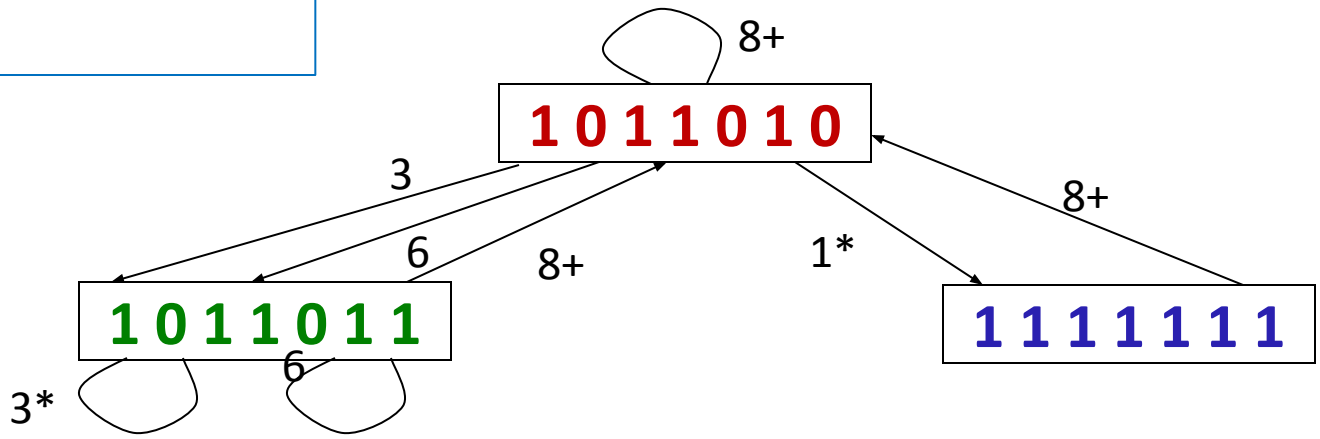
(3), (6), (8), (1, 8), (3, 8) , (6,8),
(1, 8, 6, 8), ……



- **Simple Cycle:** latency cycle in which each state is encountered only once.
- **Complex Cycle:** consists of more than one simple cycle in it.

Simple Cycles

(3) , (6) , (8), (1, 8), (3, 8), (6, 8)

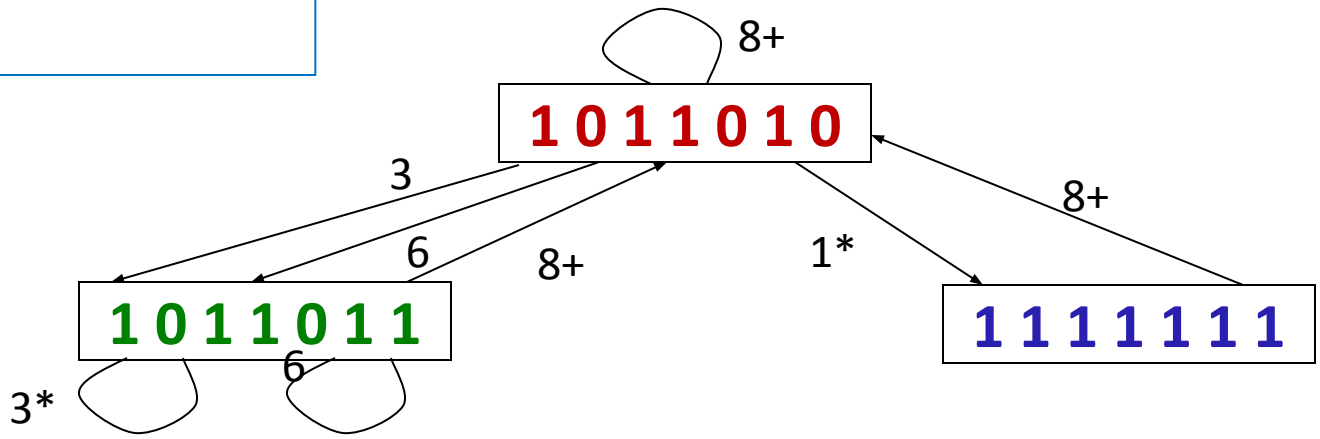


Greedy Cycles- (3), (1, 8)

(from different starting states)

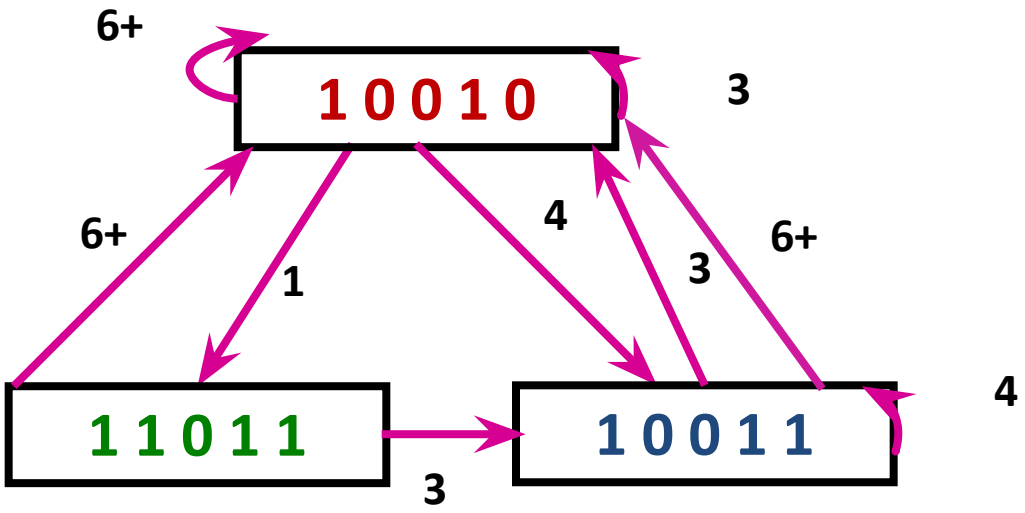
latency(1+8)/2=4.5

MAL = 3



Simple cycles & Greedy cycles

- Latency cycles are (3), (6), (4), (4,3), (1,6), (4, 6) (1,3,3) , (1,3,6), (1,6,4,3), (1,6,4,3),etc
- Simple cycles are (3), (6), (4), (4,3), (1,6) and (1,3,3)
- Greedy cycle is (1,3,3)
- In the above example, cycle that offers MAL is **(1, 3, 3)**
- **Minimum Average Latency** = (1+3+3) / 3 = **2.333**
- **MAL upper bound = 2+1 = 3** (no. of 1's in ICV + 1)
- **MAL lower bound = 2** (max. no. of check marks in any row of RT)



Pipeline Schedule Optimization

An optimization technique based on the MAL

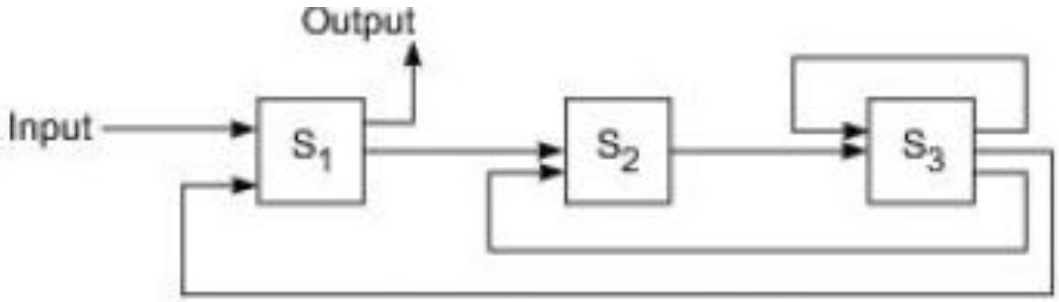
- Bounds on the MAL-Minimal average latency

(MAL) achievable by any control strategy on a statically reconfigured pipeline executing a given reservation table:

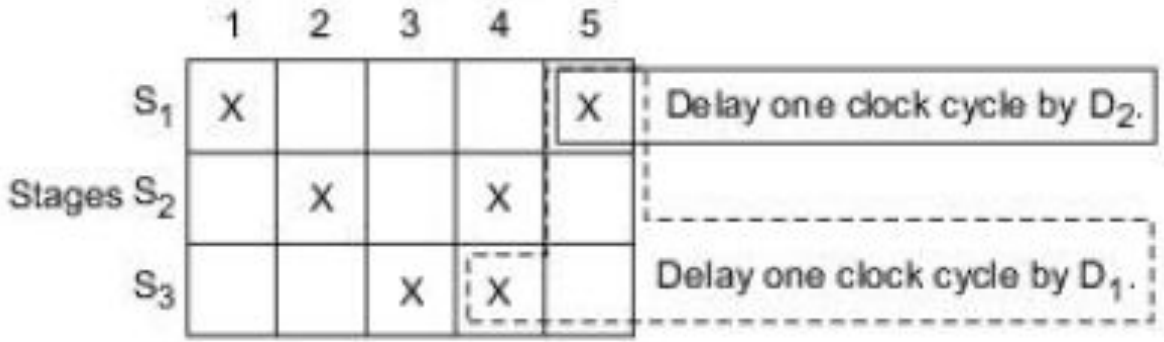
1. The MAL is lower-hounded by the maximum number of checkmarks in any row of the reservation table.
2. The MAL is lower than or equal to the average latency of any greedy cycle in the state diagram.
3. The average latency of any greedy cycle is upper-bounded by the number of 1’s in the initial collision vector plus 1. This is also an upper bound on the MAL.

Delay Insertion

- modify the reservation table, yielding a new collision vector.
- This leads to a modified state diagram, which may produce greedy cycles meeting the lower hound on the MAL

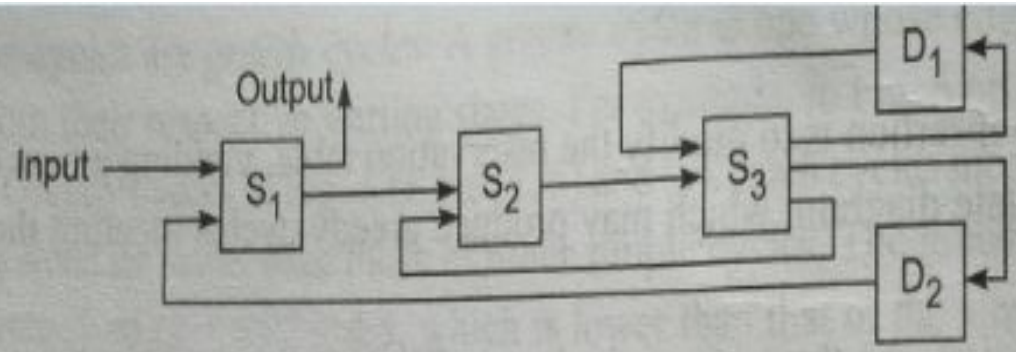


(a) A three-stage pipeline

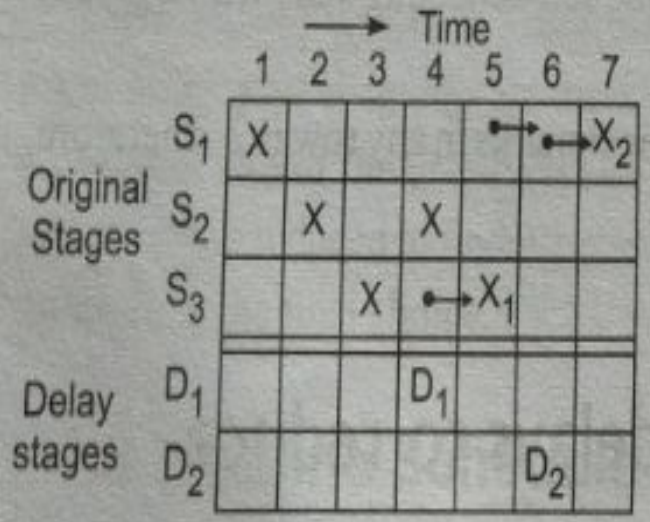


(b) Reservation table and operations being delayed

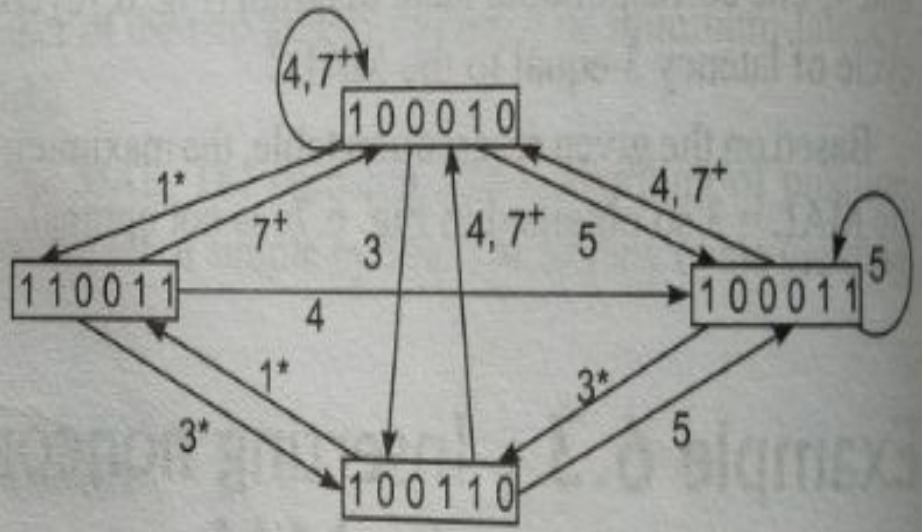
Inserting non compute delays to reduce MAL



(a) Insertion of two noncompute delay stages



(b) Modified reservation table



(c) Modified state diagram with a reduced MAL = (1 + 3)/2 = 2

Fig. 6.8 Insertion of two delay stages to obtain an optimal MAL for the pipeline in Fig. 6.7

Inserting non compute delays to reduce MAL

- Inserting a non compute delay stage D1 after stage S3 will delay both X1 and X2 operations by one cycle beyond time 4.
- Inserting yet another delay stage D2 after second usage of S1 will delay operation X2 by one more cycle.
- After Delay insertion (D1 and D2 as extra stages)
- 5-stage pipeline, X1 delayed one cycle from 4 to 5, X2 delayed two cycles from 5 to 7.
- RT = 5 + 2 = 7 columns, 3 + 2 = 5 rows
- FL = { 2, 6 }
- Collision vector = (1 0 0 0 1 0)
- State diagram contains new 4 states
- Greedy Cycle - (1, 3), MAL = (1+3) / 2 = 2 , MAL lower bound = 2
- Delay thus improves pipeline performance, yielding a lower bound for MAL

Pipeline throughput

❑ Pipeline throughput

Initiation rate or average no. of task initiations per clock cycle.

If N tasks are initiated within n pipeline cycles, then initiation rate or pipeline throughput is N/n

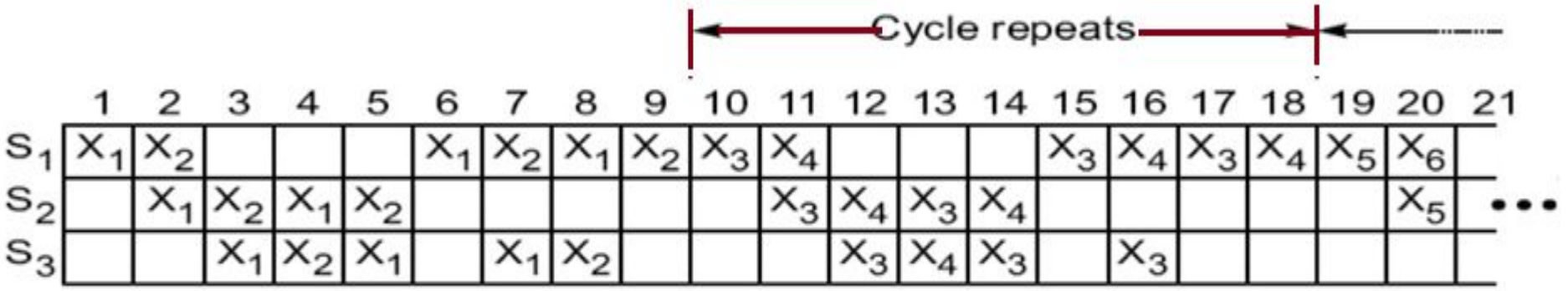
pipeline throughput = Inverse of MAL = 1/(MAL)

(shorter MAL □ higher throughput)

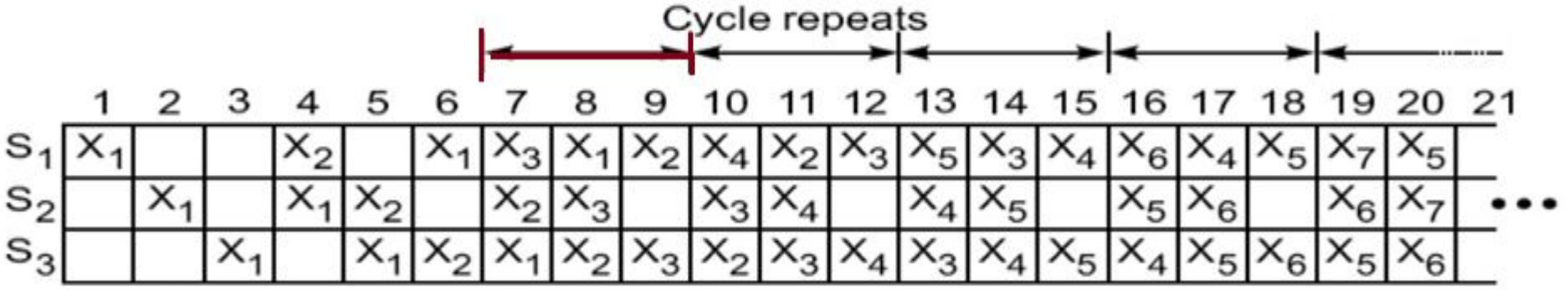
highest achievable throughput is one task initiation per cycle,

when MAL=1, since $1 \leq$ MAL $\leq$ shortest latency of any greedy cycle.

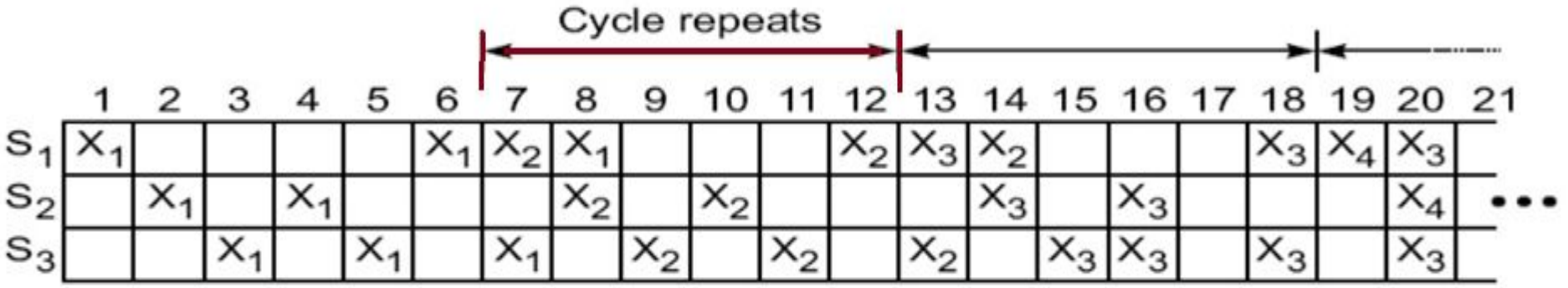
Unless MAL reduced to 1, pipeline throughput becomes a fraction.



(a) Latency cycle (1, 8) = 1, 8, 1, 8, 1, 8, ..., with an average latency of 4.5



(b) Latency cycle (3) = 3, 3, 3, 3, ..., with an average latency of 3



(c) Latency cycle (6) = 6, 6, 6, 6, ..., with an average latency of 6

Three valid latency cycles for the evaluation of function X

Pipeline Efficiency

- Stage utilization - Percentage of time that each pipeline stage is used over a sufficiently long series of task initiations.
- Pipeline Efficiency – accumulated rate of all stage utilizations.
- Latency (3) - Within each latency of three cycles, two pipeline stages S1 and S3 continuously and completely utilized after time 6. Stage S2 is used two cycles and idle for one cycle.

  Cycles(7 to 9)pattern repeats

✔ Pipeline Efficiency = 8 / 9 = 88.8 %

- Latency (1,8) - Cycles(10 to 18)

✔ Pipeline Efficiency = 14 / 27 = 51.8 %

- Latency (6) - Cycles(7 to 12)

✔ Pipeline Efficiency = 8 / 18 = 44.44 %

Pipeline Efficiency

- Pipeline throughput and pipeline efficiency are related to each other.
- Higher throughput results from a shorter latency cycle.
- Higher efficiency implies idle time for pipelines stages.
- Higher throughput accompanies higher efficiency.
- Relationship between two measures is a function of Reservation Table and Initiation cycle.
- Atleast one stage of pipeline should be fully (100 %) utilized at the steady state in any acceptable initiation cycle, else initiation cycle may not be optimal.

Pipeline Throughput

- The highest achievable throughput is one task initiation per cycle, when the MAL equals 1 since $1 \leq \text{MAL} \leq$ the shortest latency of any greedy cycle

Pipeline Efficiency

The percentage of time that each pipeline stage is used over a sufficiently long series of task initiations is the stage utilization .

- The accumulated rate of all stage utilizations determines the pipeline efficiency

- Higher throughput results from a shorter latency cycle. Higher efficiency implies less idle time for pipeline stages.