CS405
Computer System Architecture

MODULE 4

Syllabus

Message Passing Mechanisms-Message Routing schemes, Flow control Strategies, Multicast Routing Algorithms.

Pipelining and Superscalar techniques – Linear Pipeline processors and Nonlinear pipeline processors

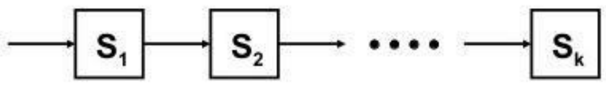
COMPUTER SYSTEMS ARCHITECTURE

Linear Pipeline Processor
PART 1
(Malayalam)

MODULE 4- PART 7

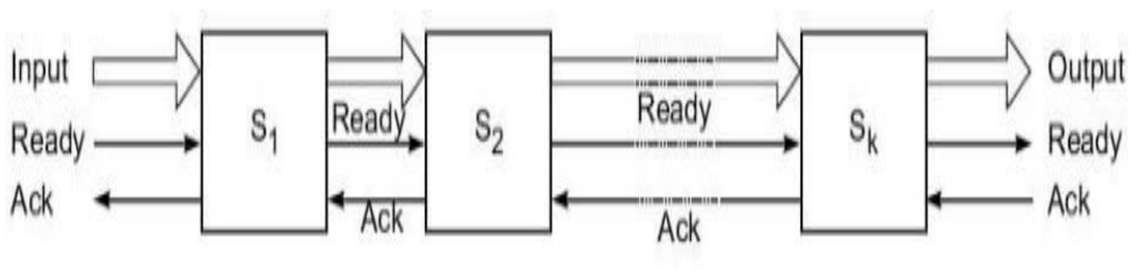
Linear Pipeline processor

- A 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.
- External input is supplied to the pipeline from stage S1.The processed result is passed from stage Si to stage Si+1, for all i=1,2..k-1. The final result is produced from the last pipeline stage.



- Applications
  - Instruction Execution
  - Arithmetic Computation
  - Memory Access Operations
- Depending on the control of data flow along the pipeline, linear pipelines are model into two categories:
  1. Asynchronous Model
  2. Synchronous Model

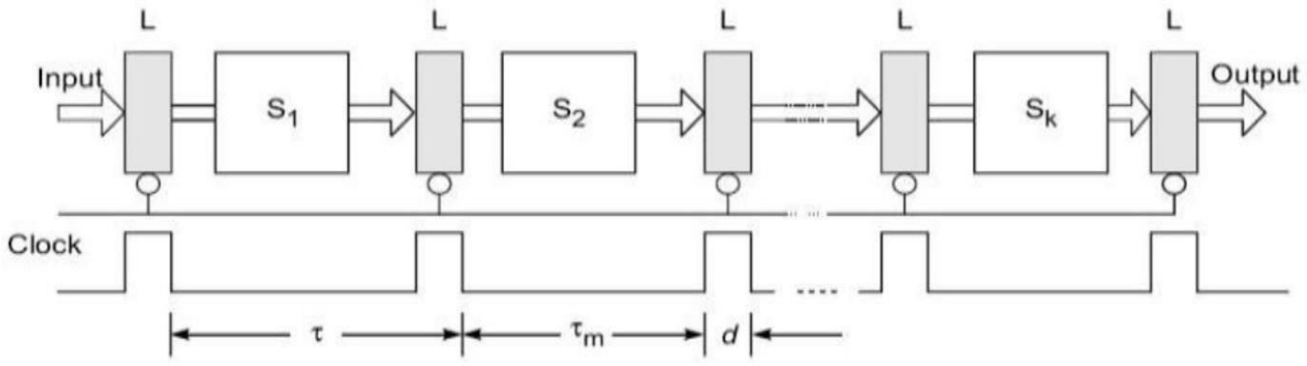
Asynchronous Pipeline Model



(a) An asynchronous pipeline model

- Advantage
  - Useful in designing communication channel message passing multi-computers.
- Disadvantage
  - May have variable throughput rate
  - Different amounts of delay may be experienced in different stages

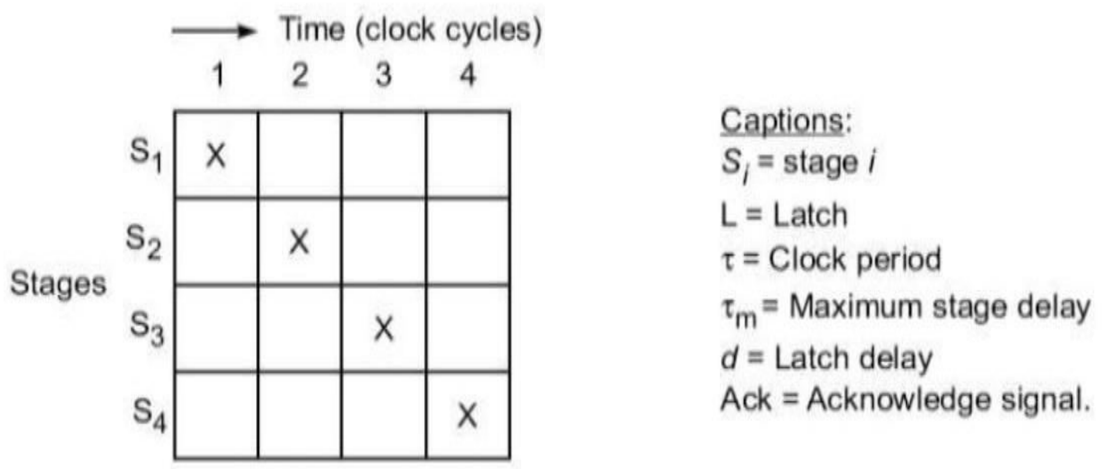
Synchronous Pipeline Model



(b) A synchronous pipeline model

- Closed latches are used to interface between stages. the latches are made with master-slave flip-flops which can isolate inputs from outputs. When a clock pulse arrives, all data is transferred to the next stage simultaneously
- The pipeline stages are combinational logic circuits. It is desired to have equal delays in all stages as delays determines the clock period and the speed of the pipeline.
- The utilization pattern is specified by pipelines in successive stages in a synchronous pipeline is defined by a reservation table.

Reservation Table



Time (clock cycles)

Stages	1	2	3	4
S_1	X			
S_2		X		
S_3			X	
S_4				X

Captions:
S_i = stage i
L = Latch
τ = Clock period
τ_m = Maximum stage delay
d = Latch delay
Ack = Acknowledge signal.

(c) Reservation table of a four-stage linear pipeline

- The utilization pattern of successive stages in a synchronous pipeline is specified by a reservation table.
- For a linear pipeline, the utilization follows the diagonal streamline pattern
- This table is essentially a space-time diagram depicting the precedence relationship in using the pipeline stages.
- For a k-stage linear pipeline, k clock cycles are needed for data to flow through the pipeline.

- Successive tasks or operations are initiated one per cycle to enter the pipeline.
- Once the pipeline is filled up, one result emerges fi'om the pipeline for each additional cycle.
- This throughput is sustained only if the successive tasks are independent of each other.

Asynchronous Model vs Synchronous Model

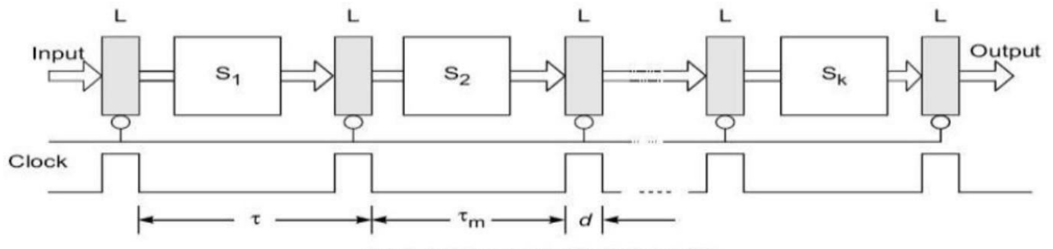
Asynchronous Model	Synchronous Model
Data flow between adjacent stages is controlled by handshaking protocol.	Clocked latches are used to interface between stages.
Different amount of delay may be experienced in different stages.	Approximately equal amount of delay is experienced in all stages.
Ready and Acknowledgement signals are used for communication purpose.	No such signals are used for communication purpose.
Transfer of data to various stages in not simultaneous.	All latches Transfer data to next stage simultaneously.
No Concept of Combinational Circuit is used in pipeline stages.	Pipeline stages are combinational logic circuits.

Clocking and Timing Control

- Clock Period or Clock cycle (τ_m)

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

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



(b) A synchronous pipeline model

- At the rising edge of the clock pulse. the data is latched to the master flip-flops of each latch register.
- The clock pulse has a width equal to d

$$\tau_m >> d$$

- This implies that the maximum stage delay τ_m dominates the clock period.

- Pipeline Frequency (f)
- Inverse of clock period
- $f = 1/\tau_m$

- If one result is expected to come out of the pipeline per cycle → f represents the maximum throughput of the pipeline.

- Depending on initiation rates of successive tasks entering the pipeline, the actual throughput maybe lower than f.
- This is because more than 1 clock cycle has elapsed between successive task initiations.

Clock Skewing

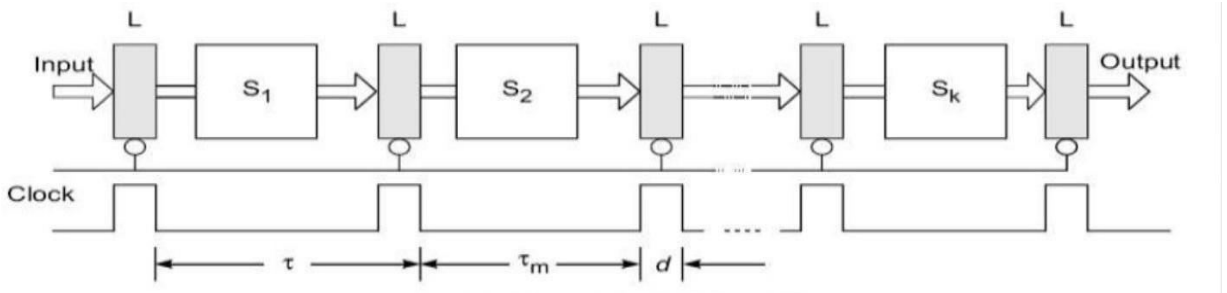
- Ideally, we expect the clock pulses to arrive at all the stages at the same time but due to a problem known as clock skewing, the same clock pulse might arrive at different stages with a time offset of s.
- tmax – time delay of longest logic path within a stage
- tmin – time delay of shortest logic path within a stage
- To avoid race in two successive stages, chose

$\tau_m \geq t_{max} + s$ and $d \leq t_{min} - s$.

- These constraints translate into the following bounds on the clock period when clock skew takes effect:

$$d + t_{max} + s \leq \tau \leq \tau_m + t_{min} - s$$

- In the ideal case s = 0, $t_{max} = \tau_m$ and $t_{min} = d$.
- Thus, we have $\tau = \tau_m + d$, consistent without the effect of clock skewing



(b) A synchronous pipeline model

Speedup, Efficiency and Throughput

- Linear Pipeline
  - a linear pipeline of it stages can process n tasks in k +(n-1) clock cycles
  - where It cycles are needed to complete the execution of the very first task and the remaining n-1 tasks require n-1 cycles
  - total time required Tk $T_k = [k + (n-1)]\tau$
  - τ →clock period

- Non Pipelined Processor
  - it has a flow-through delay of $k\tau$
  - The amount of time it takes to execute n tasks on this non pipelined processor is $T_1 = nk\tau$
  - The Speedup factor of a k-stage pipeline over an equivalent non pipelined processor is defined as:

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

Optimal Number of Stages

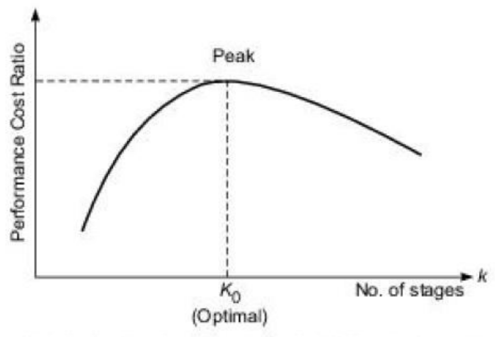
- t - total time needed on non-pipelined sequential program
- To execute the same program on a k-stage pipeline with an equal flow-through delay t.
- Clock period needed, p = t/k + d, d=latch delay
- Maximum throughput= f =1/p = 1/(t/k +d)
- Total pipeline cost= c+ kh, c is cost of all logic gates,
- h is cost of each latch
- The optimal choice of the number of pipeline stages should be able to maximize the performance/cost ratio for the target processing load.
- PCR(Performance/Cost Ratio) = max throughput/ total cost

$$PCR = \frac{f}{c + kh} = \frac{1}{(t/k + d)(c + kh)}$$

- plots the PCR as a function of k. The peak of the PCR curve corresponds to an optimal choice for the number of desired pipeline stages

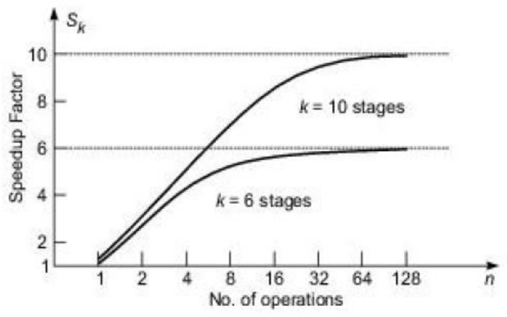
$$k_0 = \sqrt{\frac{t \cdot c}{d \cdot h}}$$

  – t →the total flow through delay of the pipeline.
  – c→total stage cost c,
  – d→latch delay
  – h→latch cost must he considered to achieve the optimal value k0



(b) Optimal number of pipeline stages (Eqs. 6.6 and 6.7)

- plots the speedup factor as a function of n, the number of tasks performed by the pipeline.
- For small values of n. the speedup can be very poor.
- The smallest value of Sk is 1 when n = 1.
- the number of pipeline stages cannot increase indefinitely due to practical constraints on costs, control complexity, circuit implementation, and packaging limitations.
- the stream length n also affects the speedup; the longer the better in using a pipeline.



(a) Speedup factor as a function of the number of operations (Eq. 6.5)

Efficiency and Throughput

- The efficiency of a linear k-stage pipeline is defined as Ek.
- Upper bound Ek → 1, when n→infinity
- Lower bound Ek = 1/k, when n=1

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

- The pipeline throughput, Hk is defined as the number of tasks (operations) performed per unit time

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

COMPUTER SYSTEMS ARCHITECTURE

Non Linear Pipeline Processor
PART 1
(Malayalam)

MODULE 4- PART 8

Non- Linear Pipeline Processor

- 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.

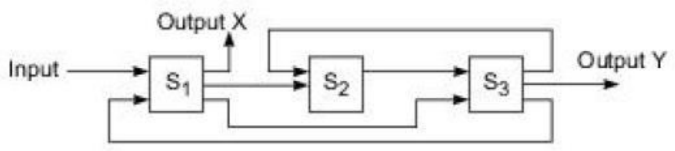
Linear Pipeline vs Non-Linear Pipeline

Linear Pipeline	Non-Linear Pipeline
static pipeline- used to perform fixed functions	dynamic pipeline - can be reconfigured to perform variable functions at different times.
allows only streamline connections	allows feed-forward and feedback connections in addition to streamline connection.
relatively easy to partition a given function into a sequence of linearly ordered sub functions.	Function partitioning is relatively difficult because the pipeline stages are interconnected with loops in addition to streamline connections.
Output of the pipeline is produced from the last stage.	Output of the pipeline is not necessarily produced from the last stage.
reservation table is trivial in the sense that data flows in linear streamline.	reservation table is non-trivial in the sense that there is no linear streamline for data flows.
Static pipelining is specified by single Reservation table.	Dynamic pipelining is specified by more than one Reservation table.
All initiations to a static pipeline use the same reservation table.	A dynamic pipeline may allow different initiations to follow a mix of reservation tables.

Reservation and Latency Analysis

- In a static pipeline, it is relatively easy to partition a given function into s sequence of linearly ordered sub functions.
- However, function partitioning in a dynamic pipeline becomes quite involved because the pipeline stages are interconnected with loops in addition to streamline connections.

- This pipeline has three stages.
- Besides the streamline connections from S1 to S2 and from S2 to S3,
- there is a feed forward connection from S1 to S3 and two feedback connection from S3 to S2 and from S3 to S1
- following different dataflow patterns, one can use the same pipeline to evaluate different functions.



(a) A three-stage pipeline

→ Time

Stages	1	2	3	4	5	6	7	8
S_1	X					X		X
S_2		X		X				
S_3			X		X		X	

(b) Reservation table for function X

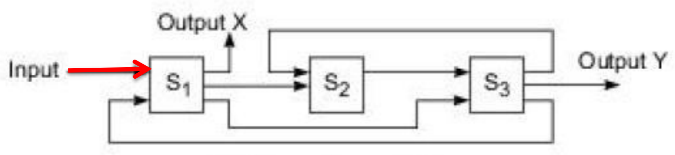
→ Time

Stages	1	2	3	4	5	6
S_1	Y				Y	
S_2			Y			
S_3		Y		Y		Y

(c) Reservation table for function Y

Fig. 6.3 A dynamic pipeline with feed forward and feedback connections for two different functions

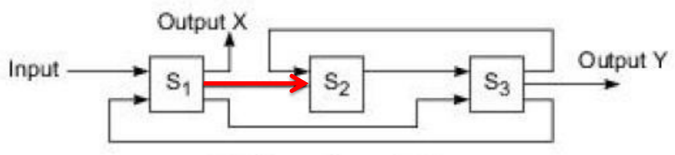
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**							
S2								
S3								

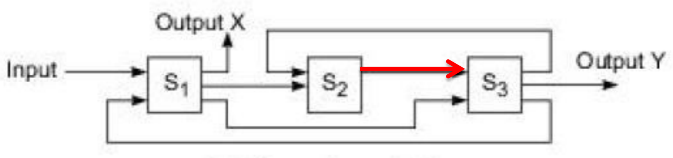
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**							
S2		X						
S3								

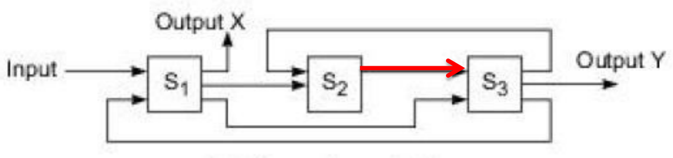
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**							
S2		X						
S3			X					

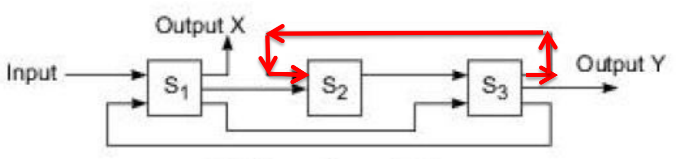
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**							
S2		X						
S3			X					

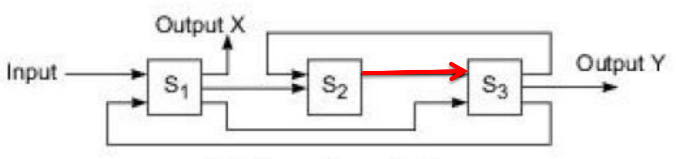
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**							
S2		X		X				
S3			X					

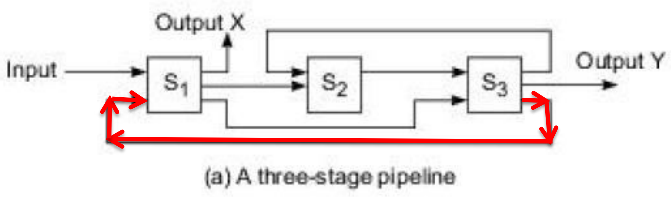
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**							
S2		X		X				
S3			X		X			

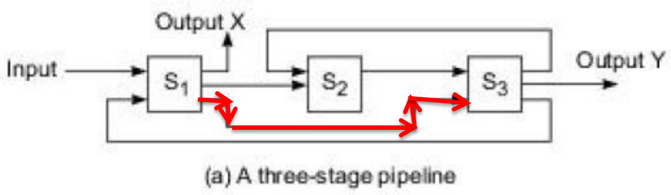
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**					X		
S2		X		X				
S3			X		X			

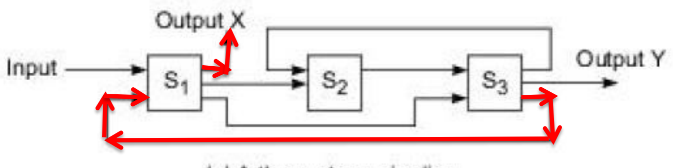
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	**X**					X		
S2		X		X				
S3			X		X		X	

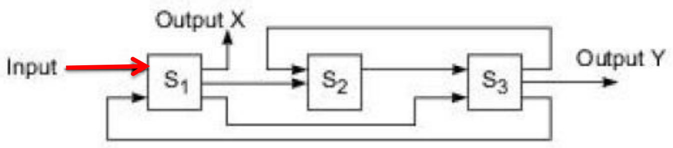
Reservation table for X function



(a) A three-stage pipeline

	1	2	3	4	5	6	7	8
S1	X					X		X
S2		X		X				
S3			X		X		X	

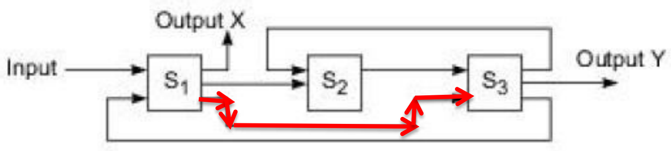
Reservation table for Y function



(a) A three-stage pipeline

	1	2	3	4	5	6
S1	Y					
S2						
S3						

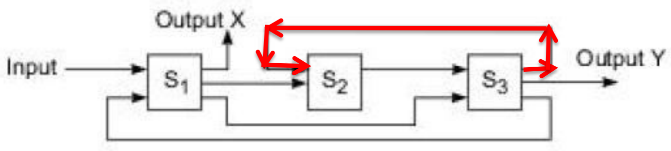
Reservation table for Y function



(a) A three-stage pipeline

	1	2	3	4	5	6
S1	**Y**					
S2						
S3		**Y**				

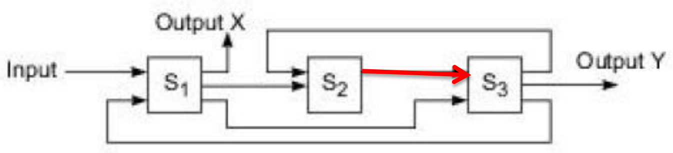
Reservation table for Y function



(a) A three-stage pipeline

	1	2	3	4	5	6
S1	**Y**					
S2			**Y**			
S3		**Y**				

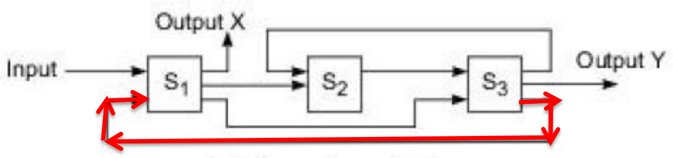
Reservation table for Y function



(a) A three-stage pipeline

	1	2	3	4	5	6
S1	**Y**					
S2			**Y**			
S3		**Y**		**Y**		

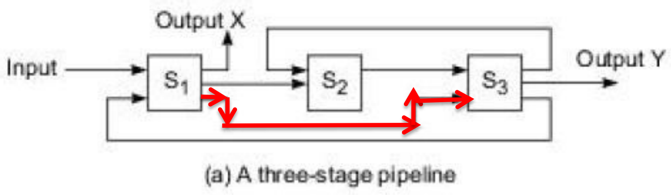
Reservation table for Y function



(a) A three-stage pipeline

	1	2	3	4	5	6
S1	**Y**				**Y**	
S2			**Y**			
S3		**Y**		**Y**		

Reservation table for Y function



(a) A three-stage pipeline

	1	2	3	4	5	6
S1	Y				Y	
S2			Y			
S3		Y		Y		Y

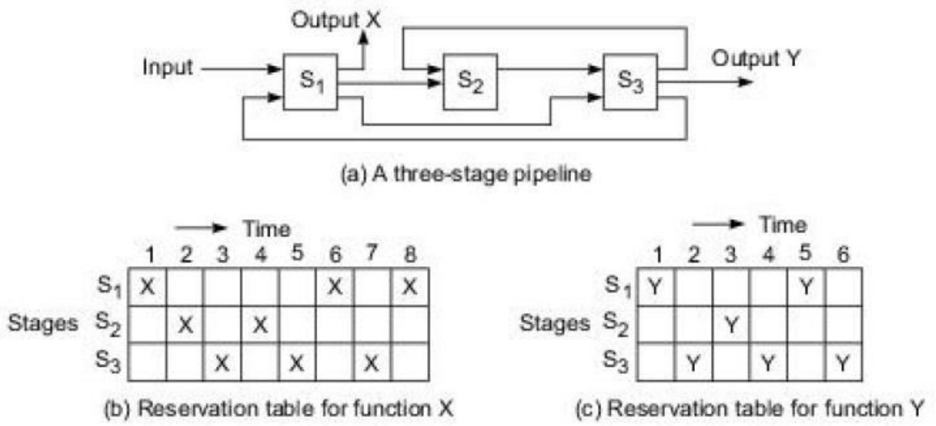
Reservation Table

- 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
- Pipeline initiation table → one function evaluation
- Static pipeline → Same RT for each function
- Dynamic pipeline → different initiations can follow a mix of reservation tables
- No.of Columns in RT → evaluation time for a given function
- Checkmarks in a row → cycles that a particular stage will be used
- Multiple checkmarks in a row → repeated usage of the same stage in different cycles. Contiguous checkmarks in a row →extended usage of same stage over more than one cycle
- Multiple checkmarks in a column → multiple stages used in parallel in a particular clock cycle.

Latency Analysis

- **Latency:** The number of time units (clock cycles) between two initiations of a pipeline is the latency between them.
- A latency value k means that two initiations are separated by k clock cycles.
- **Collision:** An attempt by two or more initiations to use the same pipeline stage at the same time. It implies resource conflicts between two initiations in the pipeline. Some latencies cause collision, some not.
- **Forbidden Latencies →** latencies that will cause collision
- **Permissible Latencies→** latencies that will not cause collision
- **Average Latency** of a latency cycle is obtained by dividing the sum of all latencies by the number of latencies along the cycle
- A **Constant Cycle** is a latency cycle which contains only one latency value. (E.g. Cycles (3) and (6) both are constant cycle).

- To detect a forbidden latency, one needs simply to check the distance between any two checkmarks in the same row of the reservation table


Output X
Input
S_1
S_2
S_3
Output Y
(a) A three-stage pipeline
Time
Stages
(b) Reservation table for function X
(c) Reservation table for function Y

Stages	1	2	3	4	5	6	7	8
S_1	X					X		X
S_2		X		X				
S_3			X		X		X	

(b) Reservation table for function X

Stages	1	2	3	4	5	6
S_1	Y				Y	
S_2			Y			
S_3		Y		Y		Y

(c) Reservation table for function Y

Fig. 6.3 A dynamic pipeline with feed forward and feedback connections for two different functions

Latencies of X:2,4,5,7

Latencies of Y:4,2,

•

Collision with scheduling latency 2

INITIAL INITIATIONS DENOTED AS X1

	1	2	3	4	5	6	7	8	9	10
S1	**X1**					**X1**		**X1**		
S2		**X1**		**X1**						
23			**X1**		**X1**		**X1**			

Collision with scheduling latency 2

AFTER X1 POST TWO CLOCK CYCLE X2 INITIALISATION ARRIVES

	1	2	3	4	5	6	7	8	9	10
S1	**X1**		**X2**			**X1**		**X1,X2**		**X2**
S2		**X1**		**X1,X2**						
23			**X1**		**X1,X2**		**X1, X2**		**X2**	

Collision with scheduling latency 2

COLLISIONS

	1	2	3	4	5	6	7	8	9	10
S1	**X1**		**X2**		**X3**	**X1**		**X1,X2**		**X2,X3**
S2		**X1**		**X1,X2**		**X3**				
23			**X1**		**X1,X2**		**X1, X2,X3**		**X2,X3**	

Collision with scheduling latency 5

INITIAL INITIATIONS DENOTED AS X1

	1	2	3	4	5	6	7	8	9	10
S1	**X1**					**X1**		**X1**		
S2		**X1**		**X1**						
23			**X1**		**X1**		**X1**			

Collision with scheduling latency 5

INITIAL INITIATIONS DENOTED AS X1

	1	2	3	4	5	6	7	8	9	10
S1	**X1**					**X1,X2**		**X1**		
S2		**X1**		**X1**			**X2**		**X2**	
23			**X1**		**X1**		**X1**	**X2**		**X2**

Latency Sequences and Cycles

- Latency sequence is a sequence of permissible non forbidden latencies between successive task initiations.
- Latency cycle is a latency sequence which repeats the same subsequence (cycle) indefinitely.

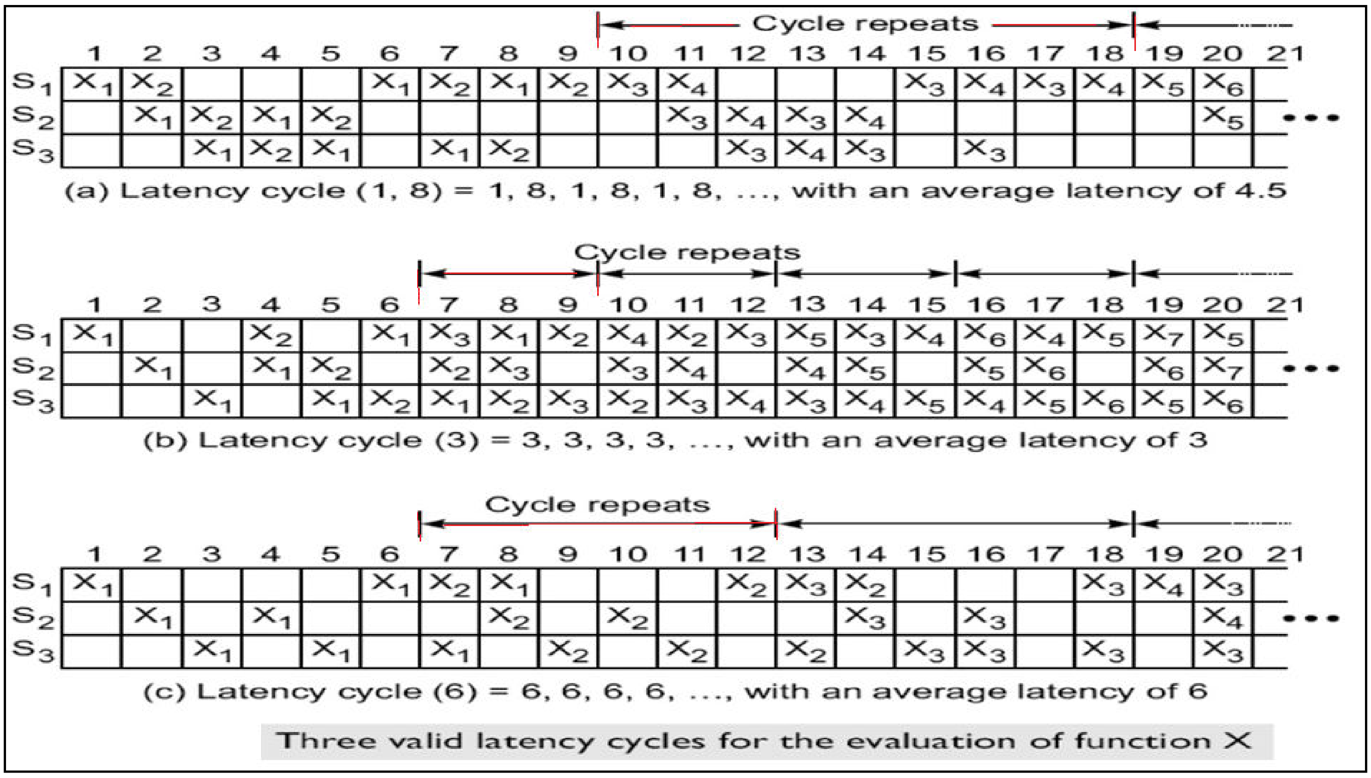
LATENCY CYCLE(1,8) = 1, 8, 1, 8, 1,8, ….

avg latency = (1+8) /2 = 4.5

LATENCY CYCLE(3) = 3, 3, 3, …. avg latency = 3

LATENCY CYCLE(6) = 6, 6, 6, …. avg latency = 6

Minimum Average Latency (MAL) ?


Cycle repeats
Cycle repeats
Cycle repeats

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	X_1	X_2				X_1	X_2	X_1	X_2	X_3	X_4				X_3	X_4	X_3	X_4	X_5	X_6	
S_2		X_1	X_2	X_1	X_2						X_3	X_4	X_3	X_4						X_5	• • •
S_3			X_1	X_2	X_1		X_1	X_2				X_3	X_4	X_3		X_3					

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	X_1			X_2		X_1	X_3	X_1	X_2	X_4	X_2	X_3	X_5	X_3	X_4	X_6	X_4	X_5	X_7	X_5	
S_2		X_1		X_1	X_2		X_2	X_3		X_3	X_4		X_4	X_5		X_5	X_6		X_6	X_7	• • •
S_3			X_1		X_1	X_2	X_1	X_2	X_3	X_2	X_3	X_4	X_3	X_4	X_5	X_4	X_5	X_6	X_5	X_6	

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	X_1					X_1	X_2	X_1				X_2	X_3	X_2				X_3	X_4	X_3	
S_2		X_1		X_1				X_2		X_2				X_3		X_3				X_4	• • •
S_3			X_1		X_1		X_1		X_2		X_2		X_2		X_3	X_3		X_3		X_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

- A constant cycle is a latency cycle which contains only one latency value.
- The average latency of a constant cycle is simply the latency itself.

COMPUTER SYSTEMS ARCHITECTURE

Non Linear Pipeline Processor
Collision-Free Scheduling
PART 2
(Malayalam)

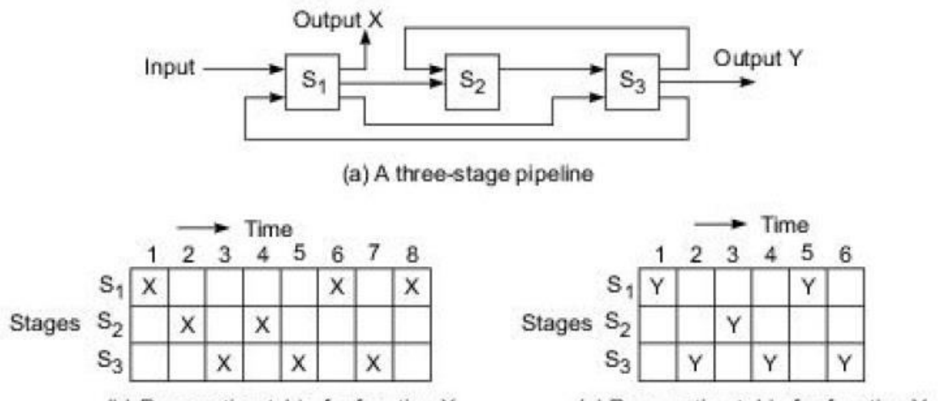
MODULE 4- PART 9

Collision-Free Scheduling

- When scheduling events in a nonlinear pipeline, the main objective is to obtain the shortest average latency between initiations without causing collisions
- We need to study following
  - – Collision vectors
  - – State diagrams
  - – Single cycles
  - – Greedy cycles
  - – Minimal Average Latency

Collision vectors

- Combined set of permissible and forbidden latencies.
- Forbidden Latencies: 2, 4, 5, 7
- Collision vector
  - $C = (C_m, C_{m-1}, \ldots, C_2, C_1)$, $m <= n-1$
  - n = number of column in reservation table
  - C contains n-1 values with C1 as LSB
  - Ci = 1 if latency i is forbidden(causes a collision)
  - Ci = 0 if latency i is permissible.
-



(a) A three-stage pipeline

Stages	1	2	3	4	5	6	7	8
S_1	X					X		X
S_2		X		X				
S_3			X		X		X	

(b) Reservation table for function X

Stages	1	2	3	4	5	6
S_1	Y				Y	
S_2			Y			
S_3		Y		Y		Y

(c) Reservation table for function Y

Fig. 6.3 A dynamic pipeline with feed forward and feedback connections for two different functions

Latencies of X:,2,4,5,7

Latencies of Y:4,2,

M=7

Collision Vector of X =C7C6C5C4C3C2C1

1 0 1 1 0 1 0

M=4

Collision Vector of Y = C4C3C2C1

1 0 1 0

State Diagram

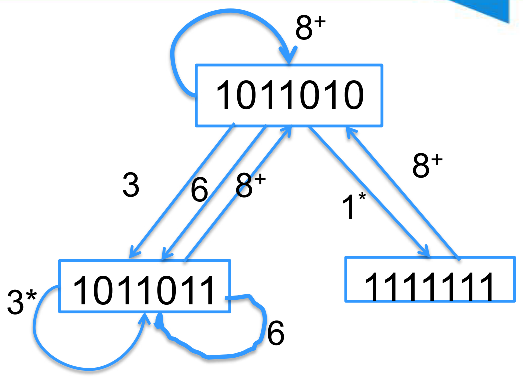
- From collision vector one can construct a state diagram specifying the permissible state transitions among successive initiations
- The collision vector, like Cx above, corresponds to the initial state of the pipeline at time 1 and thus is called an initial collision vector

- Consider initial state 1011010
- Find out the permissible latencies in the collision vector ie 1,3,6
- The new state can be obtained with permissible latency 1
- Ie by shifting initial state 1 bit to the right and OR it with initial state
- After shifting 0101101
- After shifting, logical 0 enters from left end of the shift registers
- New state is calculated by

```
 0101101
+1011010
--------
 1111111
```

- With Latency 3   With Latency 6

```
 0001011        0000001
+1011010       +1011010
--------       --------
 1011011        1011011
```



- In the new state the permissible latencies are 3 and 6(1011011)
- With latency 3   With Latency 6
-

```
 0001011        0000001
+1011010       +1011010
--------       --------
 1011011        1011011
```

- **Simple Cycle:** latency cycle in which each state is encountered only once.
- **Complex Cycle:** consists of more than one simple cycle in it.
- **Greedy Cycle:** A simple cycle is a greedy cycle if each latency (edge) contained in a cycle is the minimal latency (outgoing arc) from a state in the cycle
- A greedy cycle is one whose edges are all made with minimum latencies from their respective starting states(marked in asterisks *)
- A greedy cycle is simple, and average latency is lower than all other simple cycles.
- A good task initiation sequence should include the greedy cycle.
- Collision free scheduling reduces to finding of greedy cycles from simple cycles and one GC will yield to Minimum Average Latency (MAL)

Pipeline Schedule Optimization

❖ Bounds on Minimum Average Latency (MAL)

❑ MAL $\geq$ max. no. of check marks in any row of RT (lower bound)

❑ MAL $\leq$ avg. latency of any greedy cycle in state diagram (upper bound)

❑ Avg. latency of any greedy cycle (also MAL) $\leq$
(no. of 1's in initial collision vector + 1) (upper bound)

❑ Upper Bound on MAL

- Consider a greedy cycle $(k_1,k_2,..,k_n)$
- Let p = no. of 1's in initial collision vector
  - $\Rightarrow k_1 \leq p + 1$
  - $k_2 \leq 2p - k_1 + 2$
  - $k_3 \leq 3p - k_1 - k_2 + 3$ ....
  - $k_n \leq np - k_1 - k_2 \ldots - k_{n-1} + n$
  - $\Rightarrow k_1 + k_2 \ldots + k_n \leq np + n$ $\Rightarrow$ MAL $\leq p + 1$

Latency Cycles

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

Simple Cycles (no figure repeats)

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

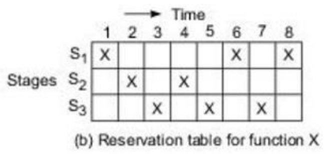
Greedy Cycles- (3), (1, 8)

(from different starting states)

Minimal Average Latency (MAL) = 3

(max no of check marks=3)

MAL upper bound = 4+1 = 5



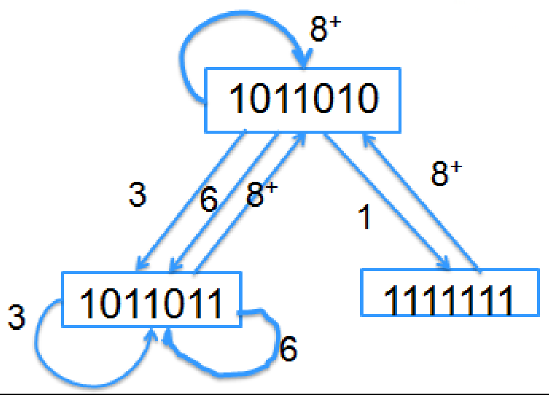
Stages	1	2	3	4	5	6	7	8
S_1	X					X		X
S_2		X		X				
S_3			X		X		X	

(b) Reservation table for function X

	m 2 1
Collision vector for X	1 0 1 1 0 1 0

1 : collision

0 : no collision



Minimum Average Latency(MAL)

1. Find Greedy cycles
2. Calculate average latency of greedy cycles
3. Then find the minimum value in the average latencies ie the MAL

Eg Greedy cycle =(1,8),(3)

Average latency of (1,8)=(1+8)/2=4.5

Average latency of (3)=3

Minimum Average Latency=3

Steps to find MAL

Given Reservation Table

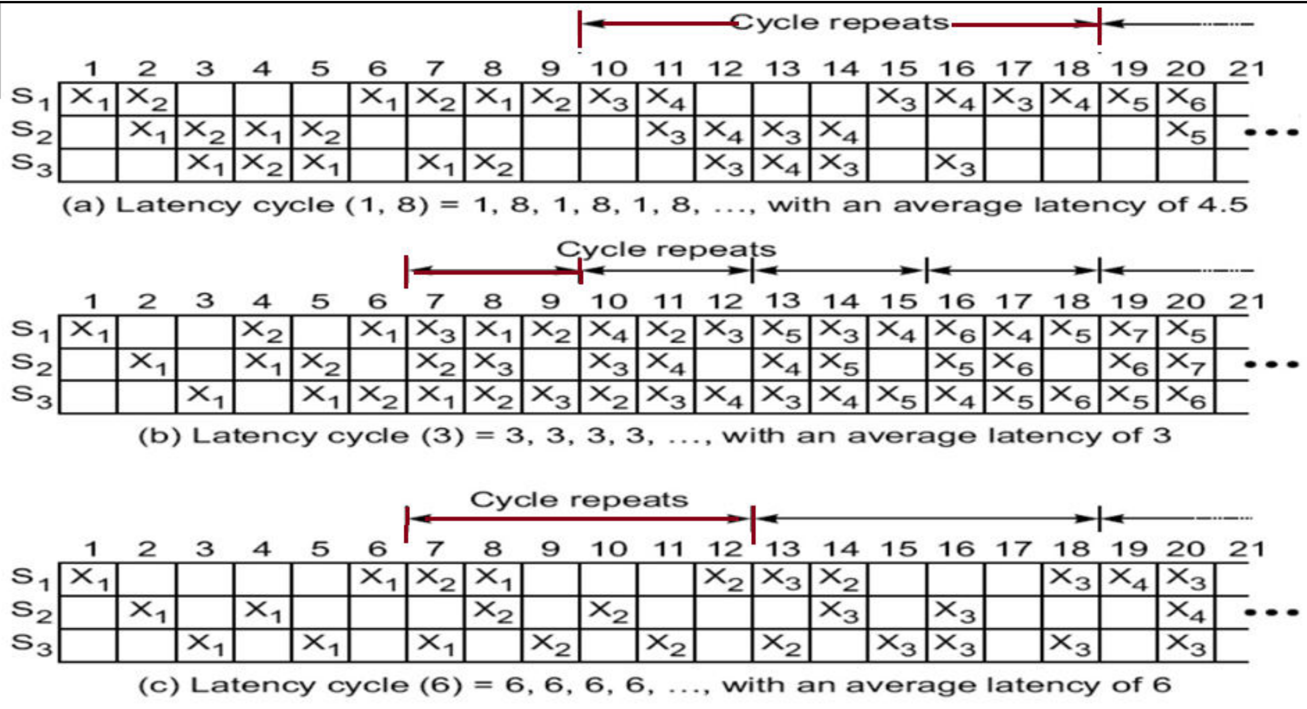
1. Find the forbidden set of latencies
2. State the collision vector
3. Draw the state transition diagram
4. List simple cycles and greedy cycles
5. Calculate MAL (minimum average latency)

Pipeline Schedule Optimization

- Optimization technique based on MAL:
- Idea is to insert non-compute delay stages into the original pipeline
- this will modify reservation table,(reduce maximum no.of check marks in a row, preserving the original function being evaluated)
- this will result in a new collision vector.
- and will result in an improved state diagram.
- produce greedy cycles meeting the lower bound on MAL
- Purpose is to yield an optimum latency cycle.

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 $1 \le \text{MAL} \le$ shortest latency of any greedy cycle.
- Unless MAL reduced to 1, pipeline throughput becomes a fraction.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	X_1	X_2				X_1	X_2	X_1	X_2	X_3	X_4				X_3	X_4	X_3	X_4	X_5	X_6	
S_2		X_1	X_2	X_1	X_2						X_3	X_4	X_3	X_4						X_5	• • •
S_3			X_1	X_2	X_1		X_1	X_2				X_3	X_4	X_3		X_3					

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	X_1			X_2		X_1	X_3	X_1	X_2	X_4	X_2	X_3	X_5	X_3	X_4	X_6	X_4	X_5	X_7	X_5	
S_2		X_1		X_1	X_2		X_2	X_3		X_3	X_4		X_4	X_5		X_5	X_6		X_6	X_7	• • •
S_3			X_1		X_1	X_2	X_1	X_2	X_3	X_2	X_3	X_4	X_3	X_4	X_5	X_4	X_5	X_6	X_5	X_6	

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

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
S_1	X_1					X_1	X_2	X_1				X_2	X_3	X_2				X_3	X_4	X_3	
S_2		X_1		X_1				X_2		X_2				X_3		X_3				X_4	• • •
S_3			X_1		X_1		X_1		X_2		X_2		X_2		X_3	X_3		X_3		X_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 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.

END OF MODULE 4