BCS-29
Advanced Computer Architecture

Pipelined Processing

Linear & Nonlinear pipelines

Instruction Pipelines & Arithmetic Operations

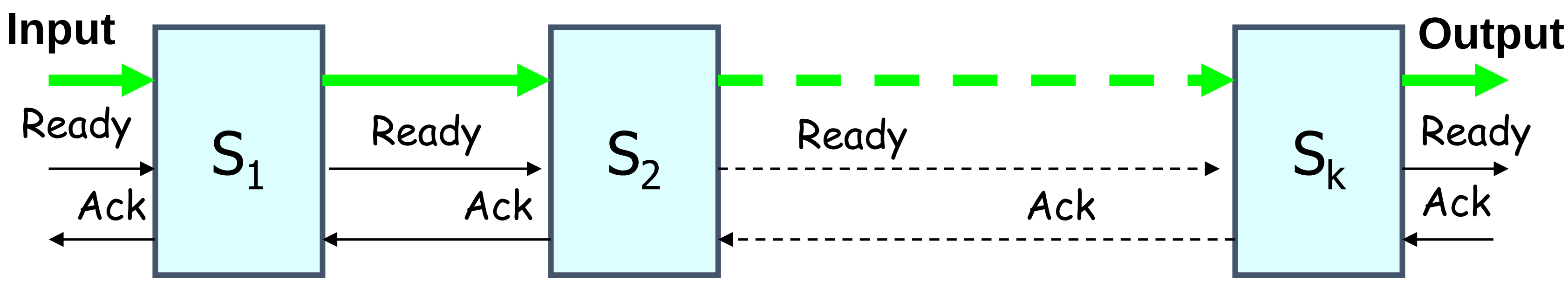
Principles of Pipelining

- A pipeline may be compared directly with an assembly line in a manufacturing plant. Thus
  - Input task or process is subdivided into a sequence of subtasks;
  - Each subtask is executed by a specialized hardware stage;
  - Many such hardware stages operate concurrently;
  - When successive tasks are streamed into the pipeline they are executed in an overlapped fashion at the subtask level.
  - The creation of the correct sequence of subtasks is crucial to the performance of the pipeline.
  - Slowest subtask is the bottleneck in the pipeline.

Linear Pipeline

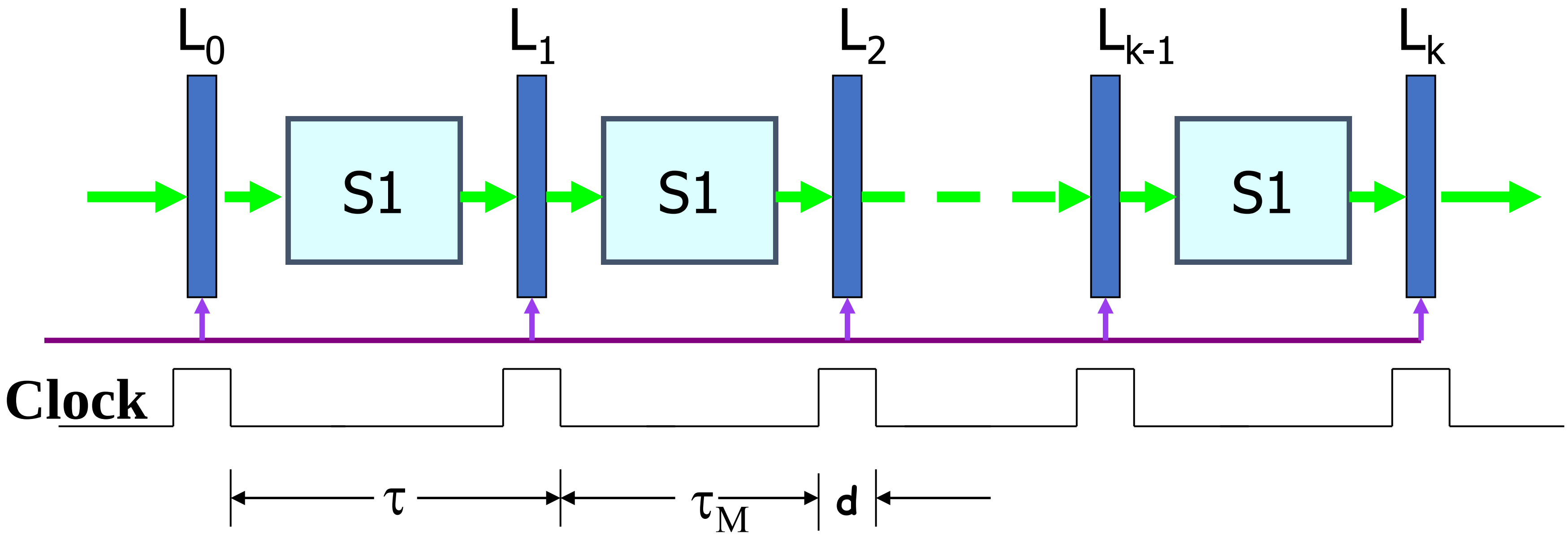
- A linear pipeline processor is a cascade of Processing Stages which are linearly connected to perform fixed function over a stream of data flowing from one end to the other.
- Linear pipeline are static pipeline because they are used to perform fixed functions.
- On the basis of the control of data flow along the pipeline. we model linear pipelines in two categories:
- Synchronous Pipeline
  - Clocked latches between Stage i and Stage i+1
  - Equal delays in all stages
- Asynchronous Pipeline (Handshaking)

Asynchronous Pipeline



- Data flow between adjacent stages in an asynchronous pipeline is controlled by a handshaking protocol.
- Asynchronous pipelines are useful in designing communication channels in massage passing multicomputer
- Asynchronous pipelines may have variable throughput rate. Different amount of delay may be experienced in different stages.

Synchronous Pipeline



- Clocked latches are used to interface between stages. On the arrival of a clock pulse, all latches transfer data to the next stage simultaneously.
- The pipeline stages are combinational logic circuits. It is desired to have approximately equal delays in all stages.
- These delays determine the clock period and thus the speed of the pipeline.

Reservation Table

- The utilization pattern of successive stages in a pipeline is specified by a Reservation Table

Time →

S1	X			
S2		X		
S3			X	
S4				X

- Reservation Table of a four stage linear pipeline

Time →

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
S1	X	X	X	X	X			
S2		X	X	X	X	X		
S3			X	X	X	X	X	
S4				X	X	X	X	X

- 5 tasks on 4 stages

Clock period and frequency

Consider t_i to be time delay due to logic circuitry in stage S_i , d to be time delay of each interface latch.

- Then
  - The **clock period**, τ , of a linear pipeline is given by

    $\tau = \max \{\tau_i\} + d$

    $= \tau_M + d$

  - The **frequency,** $\mathbf{f} = 1 / \tau$

    $= 1 / [\tau_M + d]$ (i.e., reciprocal of clock period)

Speedup, Efficiency & Throughput

- **Speedup**
  - Ideally, a linear pipeline of k stages can process n task in k + (n-1) clock so total time required is

$$T_K = \{ k + (n-1)\}\ t$$

  - Time required for nonpipelined processing

$$T_1 = n\ k\ t$$

Speedup factor $S_K = T_1 / T_k$

$$S_K = n\ k / k + (n-1)$$

- The maximum speedup is $S_K \rightarrow k$ as $n \rightarrow \infty$.
- This maximum speedup is very difficult to achieve because of the dependence structure of the program.

Speedup, Efficiency & Throughput

- Pipeline efficiency:

$$\eta = S_k / k$$

$$= n / k + (n-1)$$

So efficiency = 1 when $n \rightarrow \infty$ and

lower bound of η is 1 / k when n=1

- Throughput i.e. Number of task performed per unit time

$$H_k = n / \{ k + (n-1)\} \tau = n f / k + (n-1)$$

Maximum throughput is f, when efficiency = 1 as $n \rightarrow \infty$

Linear Instruction Pipelines

- Assume the following instruction execution phases:
  - Fetch (F)
  - Decode (D)
  - Operand Fetch (O)
  - Execute (E)
  - Write results (W)

<u>Execution</u>

F	I_1	I_2	I_3					
D		I_1	I_2	I_3				
O			I_1	I_2	I_3			
E				I_1	I_2	I_3		
W					I_1	I_2	I_3	

Dependencies

- Data Dependency
  (Operand is not ready yet)
- Instruction Dependency
  (Branching)

Will that Cause a Problem?

Data Dependency

I_1 -- Add R1, R2, R3

I_2 -- Sub R4, R1, R5

	1	2	3	4	5	6		
F	I_1	I_2						
D		I_1	I_2					
O			I_1	I_2				
E				I_1	I_2			
W					I_1	I_2		

Solutions

- STALL
- Forwarding
- Write and Read in one cycle
- ....

Instruction Dependency

I_1 – Branch o

I_2 –

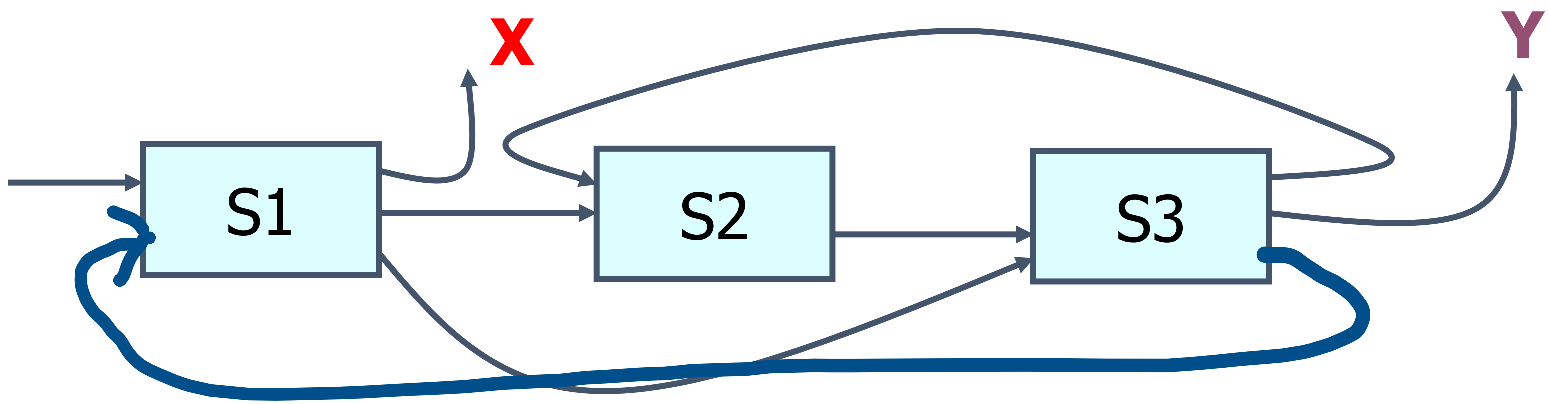
Solutions

- STALL
- Predict Branch taken
- Predict Branch not taken
- ....

	1	2		3	4		5	6
F	I_1	I_2						
D		I_1	I_2					
O			I_1	I_2				
E				I_1	I_2			
W					I_1	I_2		

Non Linear Pipelines

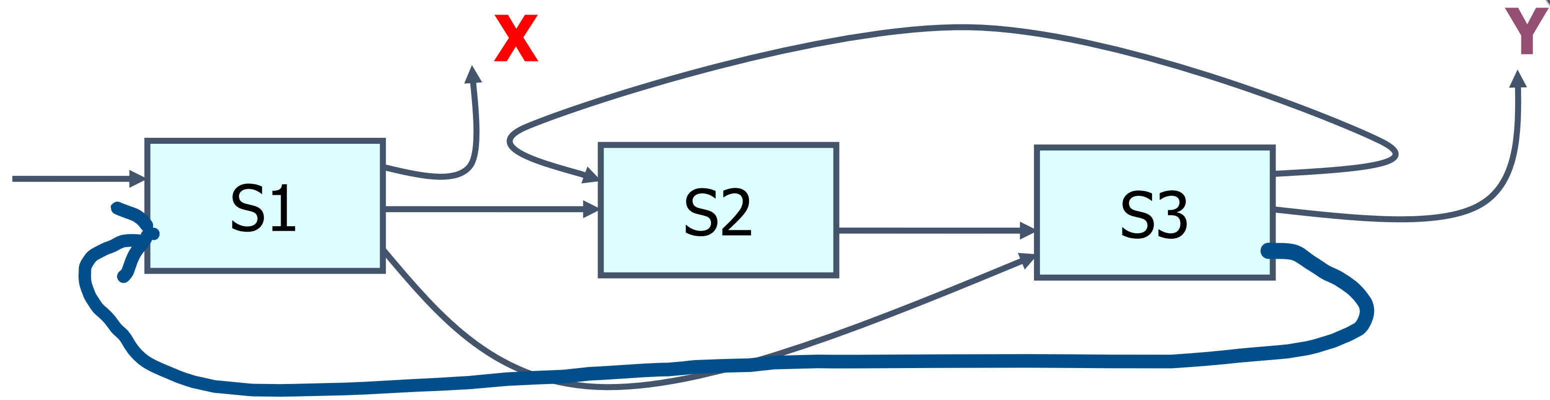
- Non-Linear pipeline are dynamic pipeline because they can be reconfigured to perform variable functions at different times.
- Non-Linear pipeline allows feed-forward and feedback connections in addition to the streamline connection.



Difference Between Linear and Non-Linear pipeline

Linear Pipeline	Non-Linear Pipeline
Linear pipeline are static pipeline because they are used to perform fixed functions.	Non-Linear pipeline are dynamic pipeline because they can be reconfigured to perform variable functions at different times.
Linear pipeline allows only streamline connections.	Non-Linear pipeline allows feed-forward and feedback connections in addition to the streamline connection.
It is 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.
The Output of the pipeline is produced from the last stage.	The Output of the pipeline is not necessarily produced from the last stage.
The reservation table is trivial in the sense that data flows in linear streamline.	The 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 Table



- There are two reservation tables corresponding to a function X and a function Y, respectively. Each function evaluation is specified by one reservation table.
- A dynamic pipeline may be specified by more than one reservation table.
- Each reservation table displays the time-space flow of data through the pipeline for one function evaluation.
- Different functions follow different paths through the pipeline.
- The number of columns in a reservation table is called the Evaluation time of a given function.

Reservation Table

Reservation table for function X

S1	X					X		X
S2		X		X				
S3			X		X		X	

Reservation table for function Y

S1	Y				Y	
S2			Y			
S3		Y		Y		Y

Reservation Table

- The check marks in each row of the reservation table correspond to the time instants (cycles) that a particular stage will be used.
- There may be multiple check marks in a row, which means repeated usage of the same stage in different cycles.
- Contiguous check marks in a row simply imply the extended usage of a stage over more than one cycle.
- Multiple check marks in a column mean that multiple stages need to be used in parallel during 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 of K means that two initiations are separated by K clock cycles.
- **Collision**
  - Any attempt by two or more initiations to use the same pipeline stage at the same time will cause a collision.
  - A collision implies resource conflicts between two initiations in the pipeline. Therefore, all collisions must be avoided in scheduling a sequence of pipeline initiations.

Forbidden Latency: Latencies that cause collisions.

Permissible Latency: Latencies that will not cause collisions.

Forbidden Latencies

- X after X

2

S1	X1		X2			X1		X2 X1
S2		X1		X2 X1		X2		
S3			X1		X2 X1		X2 X1	

5

S1	X1					X2 X1		X1
S2		X1		X1			X2	
S3			X1		X1		X1	X2

Forbidden Latencies

- X after X

4

S1	X1				X2	X1		X1
S2		X1		X1		X2		X2
S3			X1		X1		X2 X1	

7

S1	X1					X1		X2 X1
S2		X1		X1				
S3			X1		X1		X1	

Permissible Latencies

- X after X

1

S1	X1	X2				X1	X2	X1
S2		X1	X2	X1	X2			
S3			X1	X2	X1	X2	X1	X2

3

S1	X1			X2		X1		X1
S2		X1		X1	X2		X2	
S3			X1		X1	X2	X1	X2

Nonlinear Pipeline Design

- Latency Sequence
  - A sequence of permissible latencies between successive task initiations
  - Latency Sequence → 1, 8
- Latency Cycle
  - A Latency Cycle is a latency sequence which repeats the same subsequence (cycle) indefinitely.
  - Latencies Cycle → (1,8) → 1, 8, 1, 8, 1, 8 …
- Average latency
  - The average latency of a latency cycle is obtained by dividing the sum of all latencies by the number of latencies along the cycle.
  - Average Latency (of a latency cycle) → sum of all latencies / number of latencies along the cycle {(1+8)/2=4.5}
  - Constant Cycle → One latency value (3*)

Objective → Obtain the shortest average latency between initiations without causing collisions.

Latency Cycle (1,8)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
X1	X2				X1	X2	X1	X2	X3	X4				X3	X4	X3	X4	X5	X6	
	X1	X2	X1	X2						X3	X4	X3	X4						X5	X6
		X1	X2	X1	X2	X1	X2				X3	X4	X3	X4	X3	X4				X5

Average Latency = (1+8)/2 = 4.5

Scheduling events

- **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.
- **Collision vector**
  - By examining the reservation table, one can distinguish the set of permissible latencies and set of forbidden latencies.
  - The combined Set of permissible and forbidden latencies can be easily displayed by a collision vector

  - C = (Cm, Cm-1, …, C2, C1), m <= n-1
  - n = number of column in reservation table
  - Ci = 1 if latency i causes collision, 0 otherwise

Collision Vector

- Forbidden Latencies: 2, 4, 5, 7
- Collision Vector =

  1 0 1 1 0 1 0

- The next state is obtained by bitwise ORing the initial collision vector with the shifted register
  - C.V. = 1 0 1 1 0 1 0 (first state)

0 1 0 1 1 0 1   C.V. 1-bit right shifted

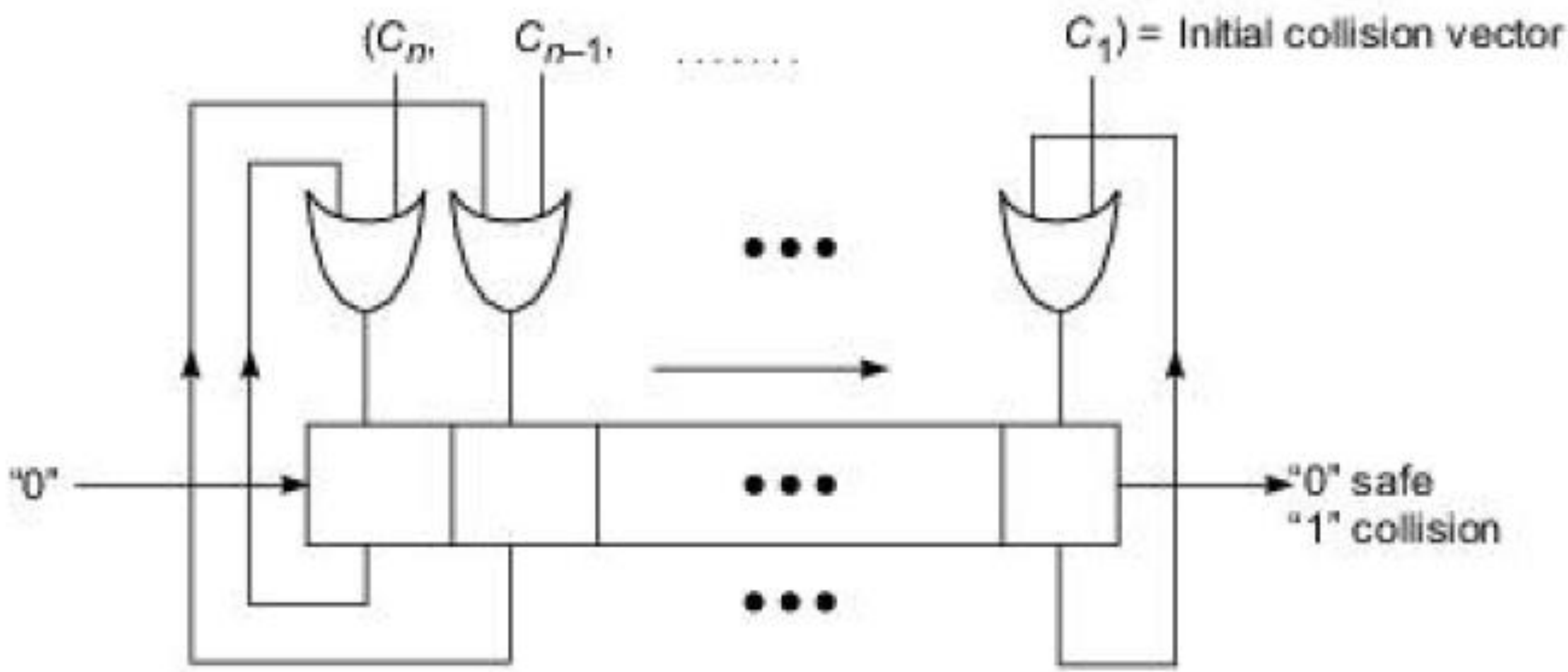
1 0 1 1 0 1 0   initial C.V.

---------------- OR

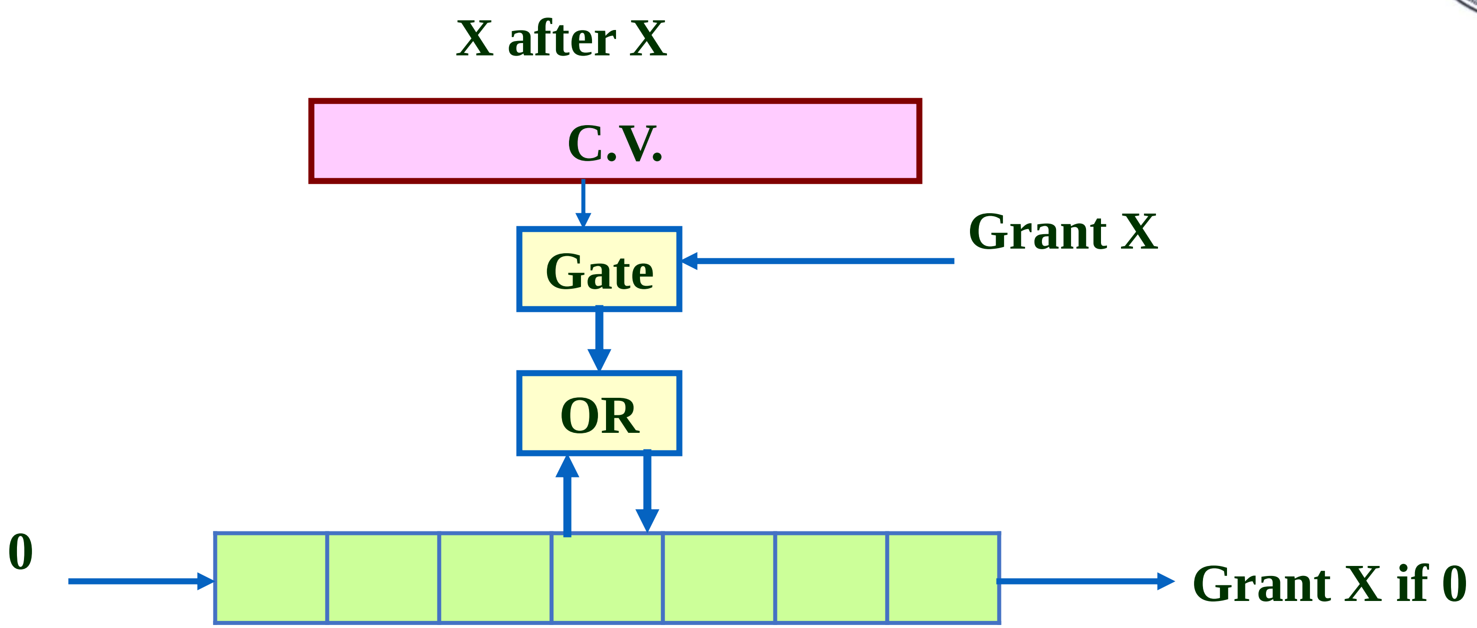
1 1 1 1 1 1 1

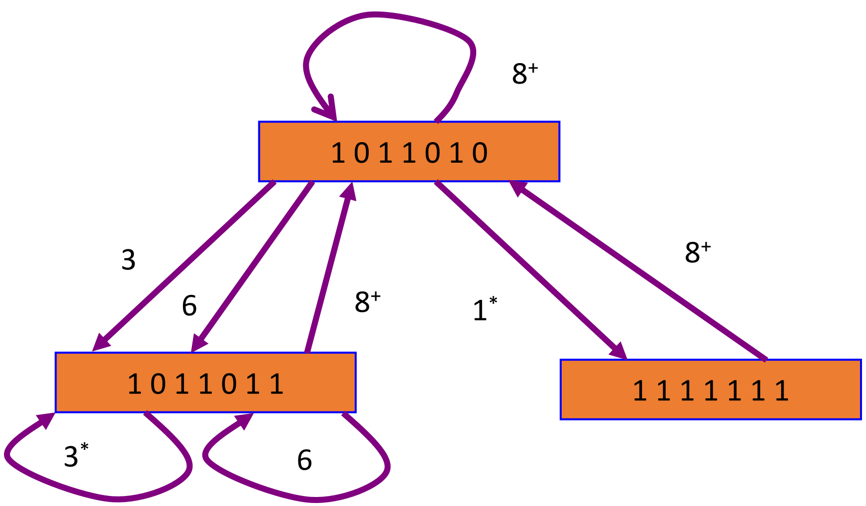
State Transition

State Transition using an n-bit Right shift register (n is maximum forbidden latency)



Latency Analysis





Cycles: (1, 8), (I, 8, 6, 8), (1, 8, 3, 8), (3), (6), [3, 8), (3, 6, 3) and many more are the legitimate cycles may be traced.

Latency Analysis

- Latency Sequence:
  - A sequence of permissible latencies between successive initiations
- Latency Cycle:
  - A latency sequence that repeats the same subsequence (cycle) indefinitely
- Simple cycles:
  - A simple cycle is a latency cycle in which each state appears only once.

    (3), (6), (8), (1, 8), (3, 8), and (6,8)
- Greedy Cycles:
  - Simple cycles whose edges are all made with minimum latencies from their respective starting states.
  - Greedy cycles must first be simple, and their average latencies must be lower than those of other simple cycles.

    (1,8), (3) → one of them is MAL(Minimum Average latency)

Minimum Average latency(MAL)

- The minimum-latency edges in the state diagrams are marked with asterisks.
- At least one of the greedy cycles will lead to the MAL.

- Bounds on the MAL
  - MAL is lower bounded by the maximum number of checkmarks in any row of the reservation table.
  - MAL is lower than or equal to the average latency of any greedy cycle in the state diagram.
  - 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 bund on the MAL.

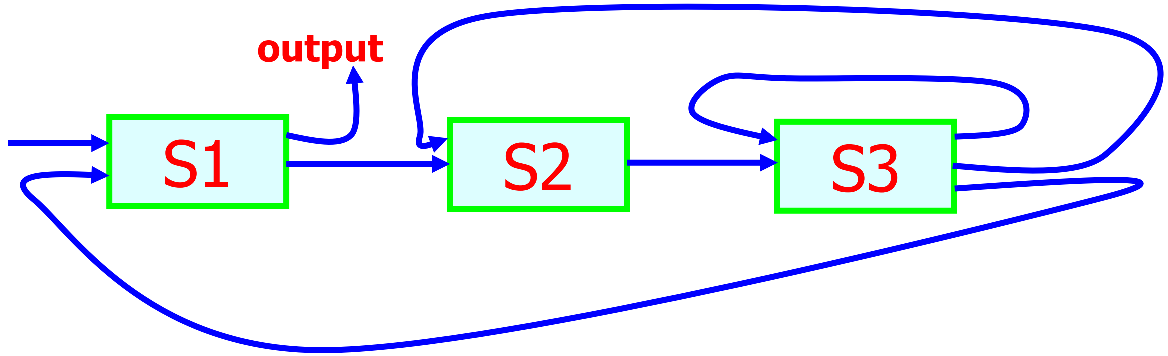
Optimization of MAL

- To optimize the MAL, one needs to find the lower bound by modifying the reservation table.
- The approach is to reduce the maximum number of check marks in any row.
- The modified reservation table must preserve the original function being evaluated.
- use of non-compute delay stages to increase pipeline performance with a shorter MAL.

Delay Insertion:

- The purpose of delay insertion is to 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 MAI...

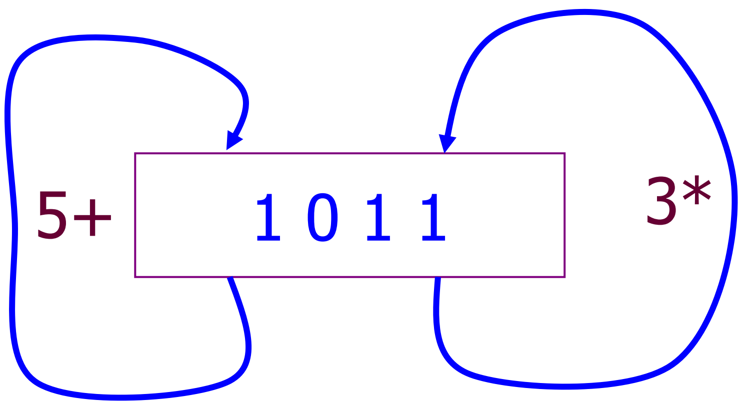
Delay Insertion



MAL = 3

Reservation Tables

	1	2	3	4	5
S1	X				X
S2		X		X	
S3			X	X	

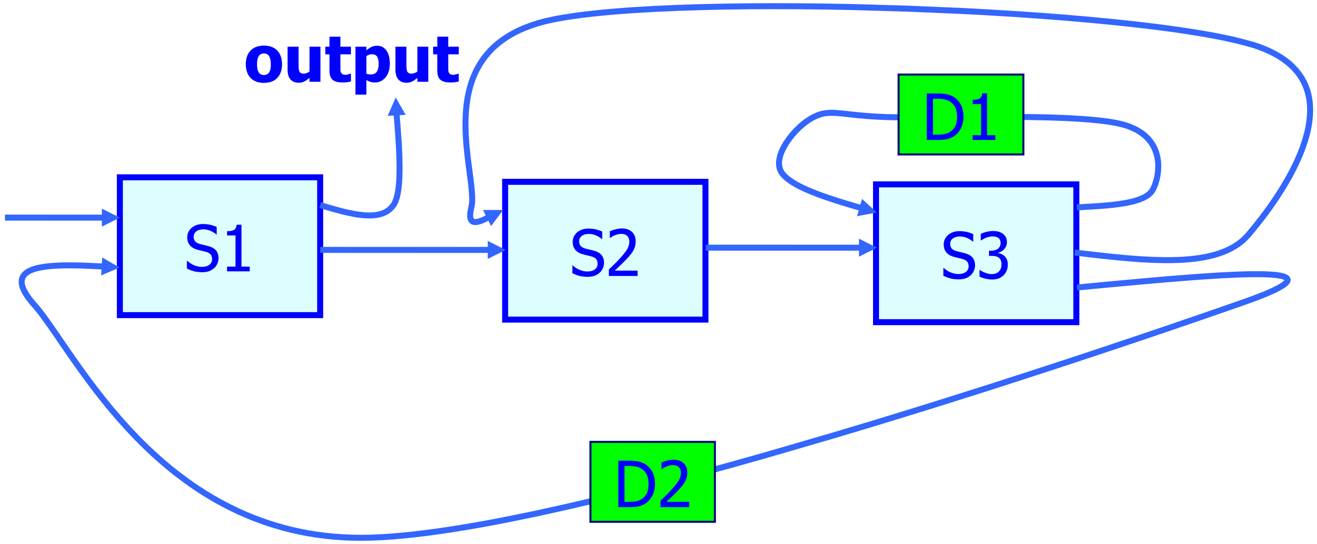


State Diagram

Forbidden Latencies: 1, 2, 4

C.V. → 1 0 1 1

Delay Insertion

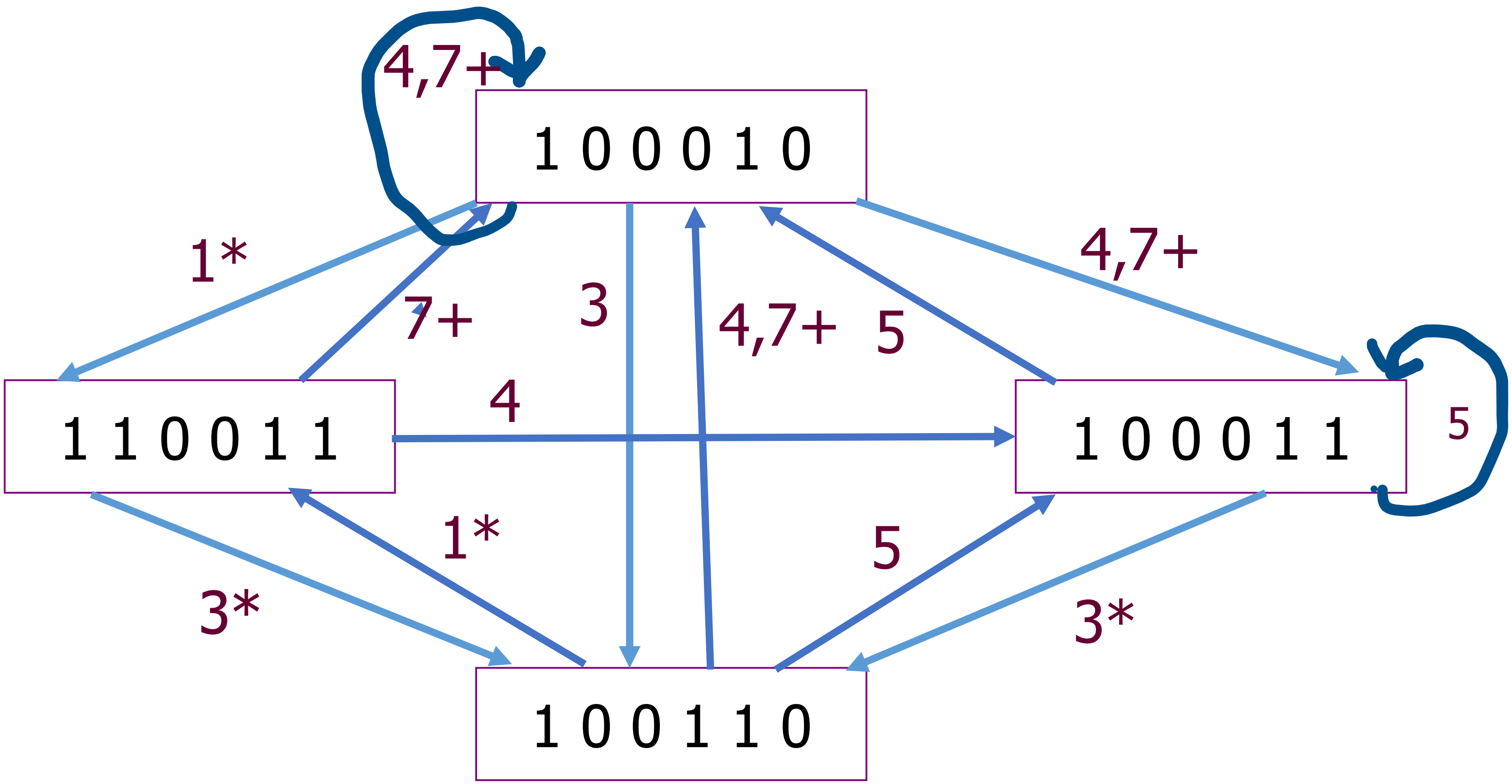


Forbidden: 2, 6

C.V. → 1 0 0 0 1 0

	1	2	3	4	5	6	7
S1	**X**						**X**
S2		**X**		**X**			
S3			**X**		**X**		
D1				**D**			
D2						**D**	

Delay Insertion



Greedy cycle (1, 3), resulting in a reduced MAL= (1 + 3)/2 = 2.