Rajiv Gandhi Proudyogiki Vishwavidyalaya, BhopalNew Scheme of Examinationas per AICTE Flexible Curricula VI Semester Bachelor of Technology (B.Tech.) Computer Science and Engineering

CS603 ACA

Topic Covered

Linear pipeline processors, Reservation Table

1.1 Linear pipeline processors

Linear pipelining Pipelining is a technique of that decomposes any sequential process into small subprocesses, which are independent of each other so that each subprocess can be executed in a special dedicated segment and all these segments operates concurrently. Thus whole task is partitioned to independent tasks and these subtask are executed by a segment. The result obtained as an output of a segment (after performing all computation in it) is transferred to next segment in pipeline and the final result is obtained after the data have been through all segments. Thus it could understand if take each segment consists of an input register followed by a combinational circuit. This combinational circuit performs the required sub operation and register holds the intermediate result. The output of one combinational circuit is given as input to the next segment. The concept of pipelining in computer organization is analogous to an industrial assembly line. As in industry there different division like manufacturing, packing and delivery division, a product is manufactured by manufacturing division, while it is packed by packing division a new product is manufactured by manufacturing unit. While this product is delivered by delivery unit a third product is manufactured by manufacturing unit and second product has been packed. Thus pipeline results in speeding the overall process. Pipelining can be effectively implemented for systems having following characteristics:

• A system is repeatedly executes a basic function.

• A basic function must be divisible into independent stages such that each stage have minimal overlap.

• The complexity of the stages should be roughly similar.

The pipelining in computer organization is basically flow of information. To understand how it works for the computer system lets consider an process which involves four steps / segment and the process is to be repeated six times. If single steps take t nsec time then time required to complete one process is 4 t nsec and to repeat it 6 times we require 24t nsec. Now let's see how problem works behaves with pipelining concept. This can be illustrated with a space time diagram given below figure, which shows the segment utilization as function of time. Lets us take there are 6 processes to be handled (represented in figure as P1, P2, P3, P4, P5 and P6) and each process is divided into 4 segments (S1, S2, S3, S4). For sake of simplicity we take each segment takes equal time to complete the assigned job i.e., equal to one clock cycle. The horizontal axis displays the time in clock cycles and vertical axis gives the segment number. Initially, process1 is handled by the segment 1. After the first clock segment 2 handles process 1 and segment 1 handles new process P2. Thus first process will take 4 clock cycles and remaining processes will be completed one process each clock cycle. Thus for above example total time required to complete whole job will be 9 clock cycles (with pipeline organization) instead of 24 clock cycles required for non pipeline configuration.

	1	2	3	4	5	6	7	8	9
P1	S1	S2	S3	S4					
P2		S1	S2	S3	S4				
P3			S1	S2	S3	S4			
P4				S1	S2	S3	S4		
P5					S1	S2	S3	S4	
P6						S1	S2	S3	S4

Fig: Space Timing Diagram

Speedup ratio : The speed up ratio is ratio between maximum time taken by non pipeline process over process using pipelining. Thus in general if there are n processes and each process is divided into k segments (subprocesses). The first process will take k segments to complete the processes, but once the pipeline is full that is first process is complete, it will take only one clock period to obtain an output for each process. Thus first process will take k clock cycles and remaining n-1 processes will emerge from the pipe at the one process per clock cycle thus total time taken by remaining process will be (n-1) clock cycle time.

Let tp be the one clock cycle time.

The time taken for n processes having k segments in pipeline configuration will be

= k*tp + (n-1)*tp= (k+n-1)*tp

The time taken for one process is tn thus the time taken to complete n process in non pipeline configuration will be

= n*tn

Thus speed up ratio for one process in non pipeline and pipeline configuration is

= n*tn / (n+k-1)*tp

If n is very large compared to k than

=tn / tp

If a process takes same time in both case with pipeline and non pipeline configuration than

tn = k*tp

Thus speed up ratio will

Sk =k*tp/tp =k

Theoretically maximum speedup ratio will be k where k are the total number of segments in which process is divided.

The following are various limitations due to which any pipeline system cannot operate at its maximum theoretical rate i.e., k (speed up ratio).

a. Different segments take different time to complete there suboperations, and in pipelining clock cycle must be chosen equal to time delay of the segment with maximum propagation time. Thus all other segments have to waste time waiting for next clock cycle. The possible solution for improvement here can if possible subdivide the segment into different stages i.e., increase the number of stages and if segment is not subdivisible than use multiple of resource for segment causing maximum delay so that more than one instruction can be executed in to different resources and overall performance will improve.

b. Additional time delay may be introduced because of extra circuitry or additional software requirement is needed to overcome various hazards, and store the result in the intermediate registers. Such delays are not found in non pipeline circuit.

c. Further pipelining can be of maximum benefit if whole process can be divided into suboperations which are independent to each other. But if there is some resource conflict or data dependency i.e., a instruction depends on the result of pervious instruction which is not yet available than instruction has to wait till result become available or conditional or non conditional branching i.e., the bubbles or time delay is introduced.

Efficiency :

The efficiency of linear pipeline is measured by the percentage of time when processor are busy over total time taken i.e., sum of busy time plus idle time. Thus if n is number of task , k is stage of pipeline and t is clock period then efficiency is given by

$\eta = n/ [k + n -1]$

Thus larger number of task in pipeline more will be pipeline busy hence better will be efficiency. It can be easily seen from expression as $n \rightarrow \infty$, $\eta \rightarrow 1$.

$\eta = Sk/k$

Thus efficiency η of the pipeline is the speedup divided by the number of stages, or one can say actual speed ratio over ideal speed up ratio. In steady stage where $n>>k$, η approaches 1.

Throughput:

The number of task completed by a pipeline per unit time is called throughput, this represents computing power of pipeline. We define throughput as

$W = n/[k*t + (n-1)*t] = \eta/t$

In ideal case as $\eta \rightarrow 1$ the throughout is equal to 1/t that is equal to frequency. Thus maximum throughput is obtained is there is one output per clock pulse.

Que.1.

A non-pipeline system takes 60 ns to process a task. The same task can be processed in six segment pipeline with a clock cycle of 10 ns. Determine the speedup ratio of the pipeline for 100 tasks. What is the maximum speed up that can be achieved?

Soln.

Total time taken by for non pipeline to complete 100 task is

= 100 * 60 = 6000 ns

Total time taken by pipeline configuration to complete 100 task is = (100 + 6 –1) *10 = 1050 ns

Thus speed up ratio will be = 6000 / 1050 = 4.76

The maximum speedup that can be achieved for this process is = 60 / 10 = 6

Thus, if total speed of non pipeline process is same as that of total time taken to complete a process with pipeline than maximum speed up ratio is equal to number of segments.

Que 2.

A non-pipeline system takes 50 ns to process a task. The same task can be processed in a six segment pipeline with a clock cycle of 10 ns. Determine the speedup ratio of the pipeline for 100 tasks. What is the maximum speed up that can be achieved?

Soln.

Total time taken by for non pipeline to complete 100 task is = 100 * 50 = 5000 ns

Total time taken by pipeline configuration to complete 100 task is = (100 + 6 –1) *10 = 1050 ns

Thus speed up ratio will be = 5000 / 1050 = 4.76

The maximum speedup that can be achieved for this process is = 50 / 10 = 5

The two areas where pipeline organization is most commonly used are arithmetic pipeline and instruction pipeline. An arithmetic pipeline where different stages of an arithmetic operation are handled along the stages of a pipeline i.e., divides the arithmetic operation into suboperations for execution of pipeline segments. An instruction pipeline operates on a stream of instructions by overlapping the fetch, decode, and execute phases of the instruction cycle as different stages of pipeline.

1.2 Non Linear Pipeline

The transfer of control is non linear. A dynamic pipeline allows feed forward and feedback connections in addition to streamline connection. A dynamic pipelining may initiate tasks from different reservation tables simultaneously to allow multiple numbers of initiations of different functions in the same pipeline.

Difference Between Linear and Non-Linear pipeline:

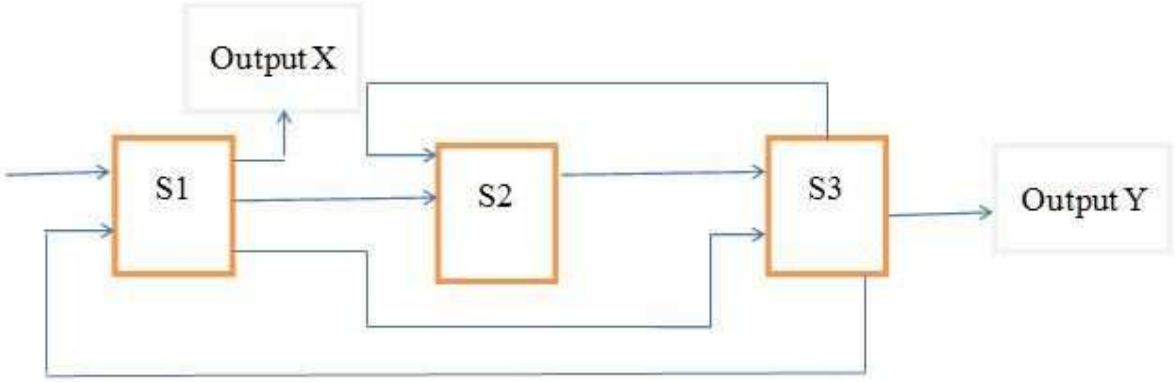
Linear Pipeline	Non-Linear Pipeline
Linear pipeline are static pipeline because they are used to perform fixed functions.	Non-Linear pipeline are dynamic pipeline because they can be reconfigured to perform variable functions at different times.
Linear pipeline allows only streamline connections.	Non-Linear 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	Function partitioning is relatively difficult because the pipeline stages are

sub functions.	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.

1.3 Reservation Table

Reservation tables are used how successive pipeline stages are utilized for a specific evaluation function. These reservation tables show the sequence in which each function utilizes each stage. The rows correspond to pipeline stages and the columns to clock time units. The total number of clock units in the table is called the evaluation time. A reservation table represents the flow of data through the pipeline for one complete evaluation of a given function.

A Three stage Pipeline



Reservation Table: Displays the time space flow of data through the pipeline for one function evaluation.

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

Reservation function for a function x

Latency: The number of time units (clock cycles) between two initiations of a pipeline is the latency between them. Latency values must be non-negative integers.

Collision: When two or more initiations are done at same pipeline stage at the same time will cause a collision. A collision implies resource conflicts between two initiations in the pipeline, so it should be avoided.

Forbidden and Permissible Latency: Latencies that cause collisions are called **forbidden latencies**. (E.g. in above reservation table 2, 4, 5 and 7 are forbidden latencies).

Latencies that do not cause any collision are called **permissible latencies**. (E.g. in above reservation table 1, 3 and 6 are permissible latencies).

Latency Sequence and Latency Cycle: A **Latency Sequence** is a sequence of permissible non-forbidden latencies between successive task initiations.

A **Latency cycle** is a latency sequence which repeats the same subsequence (cycle) indefinitely.

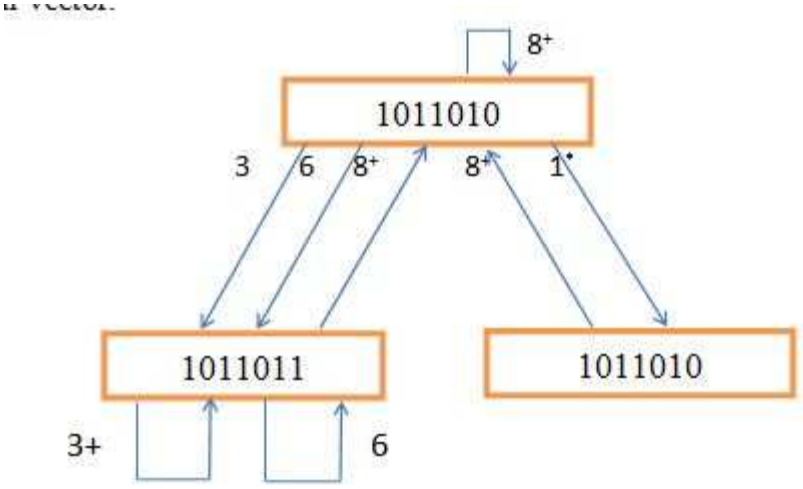
The **Average Latency** of a latency cycle is obtained by dividing the sum of all latencies by the number of latencies along the cycle.

The latency cycle (1, 8) has an average latency of (1+8)/2=4.5.

A **Constant Cycle** is a latency cycle which contains only one latency value. (E.g. Cycles (3) and (6) both are constant cycle).

Collision Vector: The combined set of permissible and forbidden latencies can be easily displayed by a **collision vector,** which is an m-bit (m<=n-1 in a n column reservation table) binary vector $C=(C_mC_{m-1}\ldots C_2C_1)$. The value of $C_i=1$ if latency I causes a collision and $C_i=0$ if latency i is permissible. (E.g. C_x= (1011010)).

State Diagram: Specifies the permissible state transitions among successive initiations based on the collision vector.



The minimum latency edges in the state diagram are marked with asterisk.

Simple Cycle, Greedy Cycle and MAL: A **Simple Cycle** is a latency cycle in which each state appears only once. In above state diagram only (3), (6), (8), (1, 8), (3, 8), and (6, 8) are simple cycles. The cycle(1, 8, 6, 8) is not simple as it travels twice through state (1011010).

A **Greedy Cycle** is a simple cycle whose edges are all made with minimum latencies from their respective starting states. The cycle (1, 8) and (3) are greedy cycles.

MAL (Minimum Average Latency) is the minimum average latency obtained from the greedy cycle. In greedy cycles (1, 8) and (3), the cycle (3) leads to MAL value 3.

References/Suggested readings

Advance Computer architecture: Kai Hwang