CPRE/COMS 583x Adaptive Computing Systems	Fall Semester 1998
September 29, 1998 and October 1, 1998	
Instructor: Arun K. Somani & Akhilesh Tyagi	Scribe: Seongwoo Kim

1 Introduction

Pipelining is one way of improving the overall processing performance of a processor. This architectural approach allows the simultaneous execution of several instructions. Pipelining is transparent to the programmer; it exploits parallelism at the instruction level by overlappoing the execution process of instruction. It is analogous to an assembly line where workers perform a specific task and pass the partially completed product to the next worker.

In this lecture, we learn basic concepts of pipelining and procedure to find latency for avoiding collision in both linear and nonlinear pipelines.

2 Linear pipeline

Figure 1 shows that a pipeline can be formed by splitting a given task, T, into k stages. Each stage performs a part of T. Pipelining requires latches (a register or buffer) for each stage, which cause extra delay in processing (we denote this by τ). These additional storage and time correspond to pipeline penalty. Let T_{seq} be time required to complete T on a sequential machine (no overlapped execution). Throughtput of this sequential machine (TP_{seq}) is $1/T_{seq}$. Let T_i be time for stage i to perform its partitioned task on a pipelined machine. τ is very small in comparison to T_i. If $T_{seq} = T_1 + T_2 + ... + T_k$, the time for the pipelined machine to perform T is

$$T_{pipe} = T_{seq} + k\tau = MAX\{T_i\} \cdot k + k\tau.$$

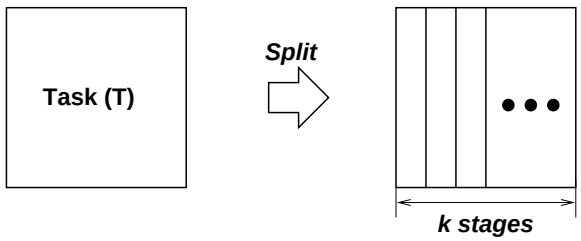

Figure 1: Basic pipeline model: A task is splitted into k stages.

For n T taskes on this pipelined machine,

$$\text{total time} \quad = \quad T_{seq} + k\tau + (\frac{T_{seq}}{k} + \tau)(n-1)$$

$$
\begin{aligned}
&= T_{seq} + \frac{n}{k}T_{seq} - \frac{T_{seq}}{k} + (n+k-1)\tau \\
&= (n+k-1)\frac{T_{seq}}{k} + (n+k-1)\tau. \\
TP_{pipe} &= \frac{n}{(n+k-1)(\frac{T_{seq}}{k}+\tau)} \\
&\simeq \frac{1}{\frac{T_{seq}}{k}+\tau}(fork << n) \\
&= \frac{k}{T_{seq}+k\tau} \simeq k \cdot TP_{seq}. \\
Speedup &= \frac{TP_{pipe}}{TP_{seq}} \simeq k.
\end{aligned}
$$

To find an optimal choice for the number of stages, k, we define $\mathsf{Cost} = \alpha k + \beta$ where β is original cost and α is addional cost for pipelining per stage. Thus, $\mathsf{Time} \times \mathsf{Cost} = (T_{seq} + k\tau)(\alpha k + \beta)$. This equation has a minimum value k, where

$$k = \sqrt{\frac{T_{seq} \cdot \beta}{\tau \cdot \alpha}}.$$

3 Non-linear pipeline

Pipelines can be achieved at various levels as follows.

1. Processor level: $P1, P2, ..., Pn$
2. Instruction execution level: $S1, S2, ..., Sn$
3. ALU level:

   integer multiplication $(M1, M2, ..., Mn)$,

   floating point addition $(FA1, FA2, ..., FAn)$, etc.

Depending on functions we need to realize in a pipeline, we can come up with nonlinear pipelines. Figure 2 and Figure 3 are examples of nonlinear pipelines. There are two types of pipelines: static and dynamic. A static pipeline can perform only one function at a time, whereas a dynamic pipeline can perform more than one function at a time. A pipeline reservatin table shows when stages of a pipeline are in use for a particular function. Each stage of the pipeline is represented by a row in the reservation table. Each row of the reservation table is in turn broken into columns, one per clock cycle. The number of columns indicates the total number of time units required for the pipeline to perform a particular function. Suppose two functions, X and Y, are executed using a three-stage dynamic pipeline shown in Figure 3. Two reservation tables are shown in Figure 4.

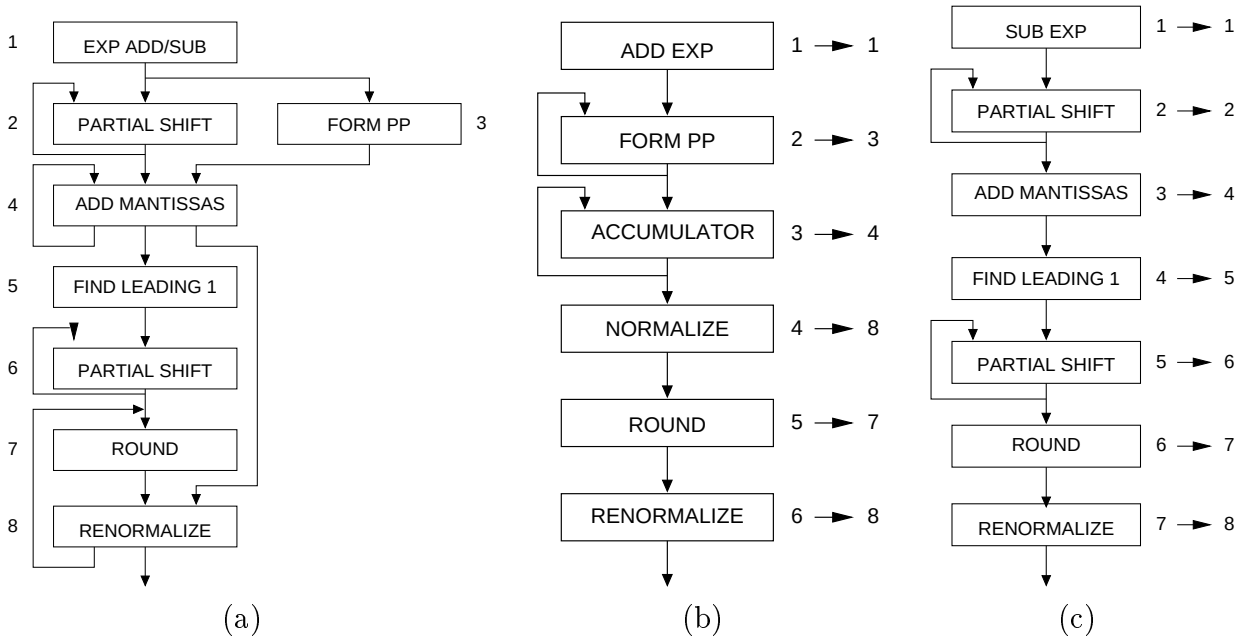

Figure 2: Design of multi-function pipeline: (a) A FP ADD/MULT unit. (b) A FP MULT unit. (c) A FP ADD unit

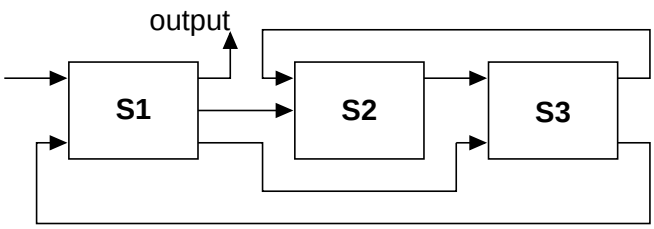

Figure 3: Three-stage nonlinear pipeline.

1	2	3	4	5	6	7	8
X					X		X
	X		X				
		X		X		X	

1	2	3	4	5	6
Y				Y	
		Y			
	Y		Y		Y

Figure 4: Reservation tables for fuctions X and Y.

The delay, or number of time units separating two initiations, is called *latency*. A collision will occur if two pieces of input data are initiated with a latency equal to the distance between two X or Y. Reservation table with two or more X's or Y's in any given row has one or more forbidden latencies, which, if not prohibited, would allow two data to collide or arrive at the same stage of the pipeline at the same time. The forbidden list F is simply a list of integers corresponding to these prohibited latencies. They are $F_X : 2, 4, 5, 7$ and $F_Y : 2, 4$.

A collision vector is a string of binary digits of length N, where N is the largest forbidden latency in the forbidden list. The initial collision vector, C, is created from the forbidden list in the following way: each compondent c_i of C, for $i = 1$ to N, is 1 if i is an element of the forbidden list. Otherwise, c_i is zero. Zeros in the collision vector indicate allowable latencies, or times when initiations are allowed into the pipline. We can easily obtain two collision vectors, $C_x : 1011010$ and $C_y : 1010$.

State diagrams can be used to show the different state of a pipeline for a given time slice. Once a state diagram is created, it is easier to derive schedules of input data for the pipeline that have no collisions. Figure 5 shows the state diagrams for two functions. The arc represents the latency. From the state diagrams, we can find the cycles that can avoid collision. Cycles (1, 5) and C (3) are examples for function X and Cycles (1, 8), (3), (6), (8), (3, 8), and (6, 8) are for function Y.

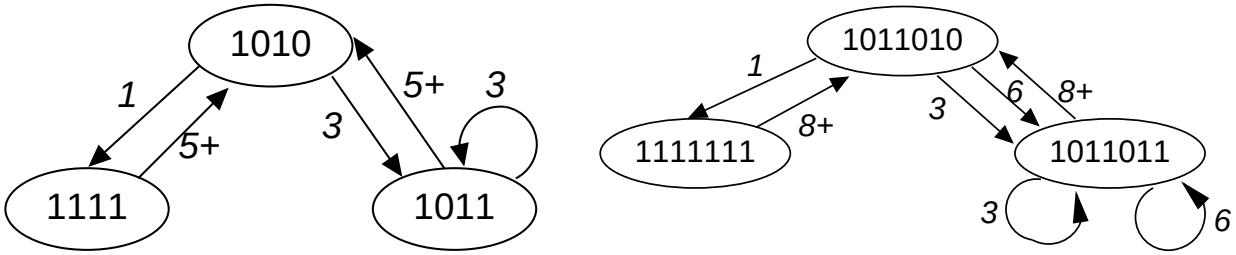

Figure 5: State diagrams: (a) X. (b) Y.

A pipeline may have several average latencies associated with different cycles. The minimum average latency (MAL) is simply the smallest such ratio.

With a dynamic pipeline given in Figure 3, it is possible for different input data requiring two functions to be present in the pipeline at the same time. Therefore, collisions between these data must be considered as well. For X and Y, the fordiddent lists are $F_{xx} = \{2, 4, 5, 7\}$, $F_{xy} = \{1, 3, 4\}$, $F_{yx} = \{1, 3, 5, 7\}$, and $F_{yy} = \{2, 4\}$. Thus, corresponding collision vectors are $V_{xx} = (1011010)$, $V_{yx} = (1010101)$, $V_{xy} = (0001101)$, and $V_{yy} = (0001010)$. The collision vectors for function X and Y form the collison matrix M_x and M_y, respectively, they are,

$$M_x = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}, \quad M_y = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Based on these, we can build a state diagram as shown in Figure 6. The resulting state diagram is much more complicated than a static pipeline state diagram due to the larger number of potential collisions.

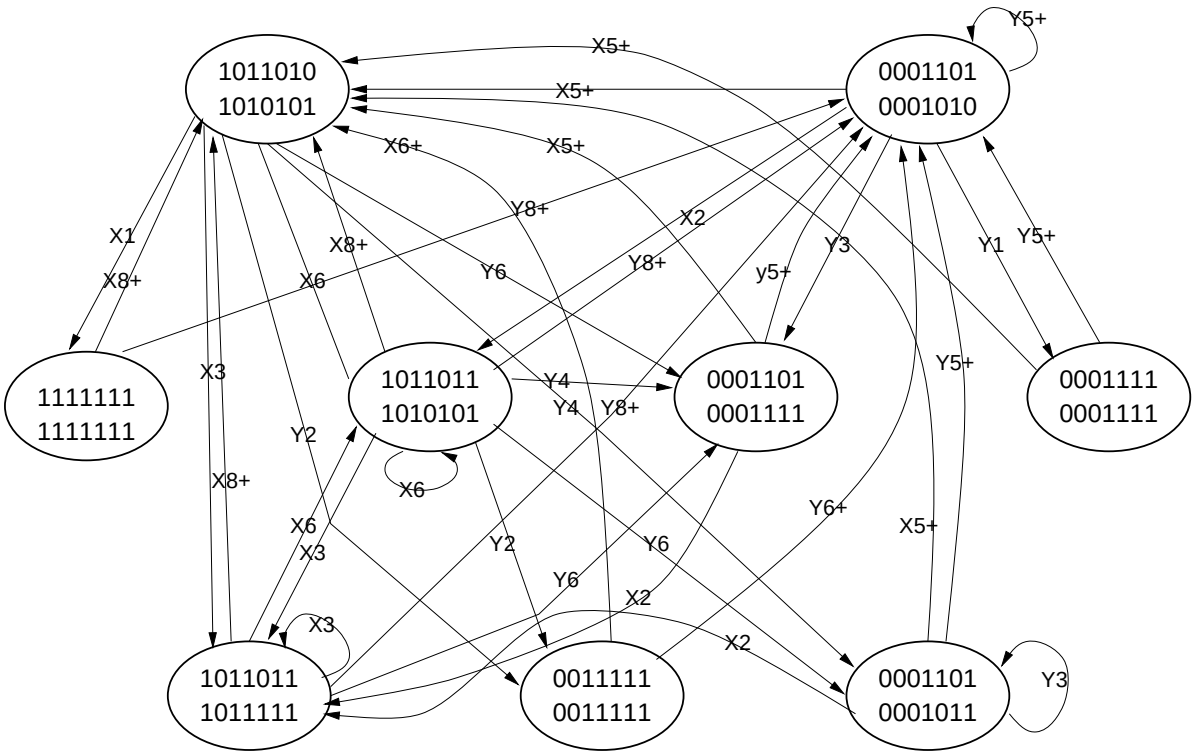

Figure 6: State diagram of a dynamic pipeline.