between this new matrix and the initial collision matrix M_A, because the original forbidden latencies for functional unit A still have to be considered in later initiations.

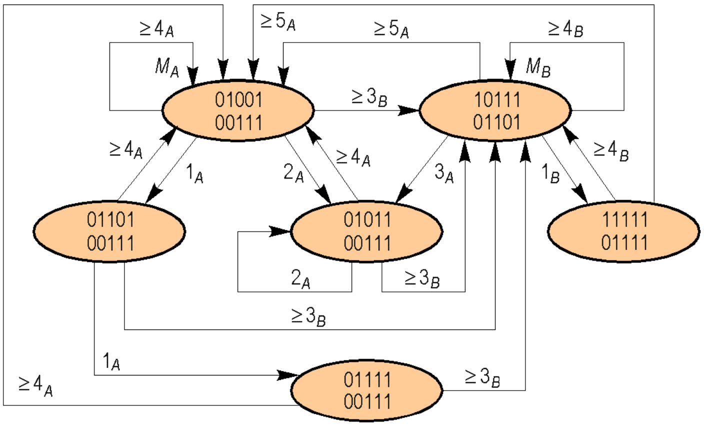


Figure 5.37. State diagram of a dynamic pipeline.

If the first allowable latency for vector C_{AB} in matrix M_A is chosen (in this case 3), the entire matrix is shifted right three places, with zeros filling in on the left. Then a logical OR operation is performed between this new matrix and the initial collision matrix for function B, because the original collisions for function B are still possible and have to be considered. The shifting and logical OR operations continue until all possible allowable latencies are considered and the state diagram is complete.

5.7. Problems

5.7.1. Assume a pipeline with four stages that performs an add operation on two input numbers. The pipeline has a delay of two units of time for each stage. Also, assume that there are adequate memory and buffers. Ignore storage time, memory access time, time to set up the pipeline control circuit, and so on. It is necessary to sum 16 numbers with this pipeline. Describe a method that gives the minimum possible time to perform the sum of the numbers. Estimate the total time required by this method.

5.7.2. Indicate the type of data hazards (RAW, WAR, and WAW) that exist between the following instructions of a RISC processor:

```
ADD    R1, R2, R3           ; R1 = R2 + R3
ADD    R4, R1, R4           ; R4 = R1 + R4
ADD    R3, R1, R2           ; R3 = R1 + R2
ADD    R1, R1, R4           ; R1 = R1 + R4
```

5.7.3. Identify all possible RAW, WAR, and WAW hazards that exist between the instructions of the following program of a hypothetical RISC processor:

```
       LOAD   R4, #A        ; load constant A into R4
       LOAD   R5, #B        ; load constant B into R5
       LOAD   R6, #C        ; load constant C into R6
       LOAD   R9, #0        ; clear R9
       BEQ    R4, R5, ADR1  ; if R4=R5 then go to ADR1
       ADD    R9, R4, R5    ; R9 = R4 + R5
       MUL    R9, R9, R9    ; square the contents of R9
       ADD    R9, R9, #1    ; increment R9
ADR1:  STORE  (R1), R9      ; store R9 in the memory location
                            ; addressed by R1
```

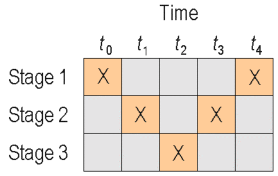
5.7.4. Compare Tomasulo's method with the scoreboard method. Describe the advantages and disadvantages of each.

5.7.5. Calculate the throughput for a pipelined instruction in which $p_b = 0.1$, $p_t = 0.4$, and $c = 4$. Assume that a branch prediction technique is used to improve the performance of the pipeline. What will be the new throughput for the instruction when $p_r = 0.7$ and $d = 1$?

5.7.6. Design a pipeline that efficiently multiplies a stream of 4-bit numbers A_1, A_2, A_3, …, with a constant 4-bit number B. Assume that A and B are positive integers.

5.7.7. Assume a pipeline that can perform addition or multiplication operations. The pipeline consists of k stages, and each stage has a delay of $1/k$ time units. When switching between operations (from addition to multiplication, or vice versa), the pipeline should be drained before the data for the new operation can be applied to the pipeline. Assume that there are adequate memory and buffers. Also, ignore storage time, memory access time, time to set up the pipeline control circuit, and so on. Determine the minimum time required to multiply two $n \times n$ matrices (where $n > k$) to produce an $n \times n$ matrix product.

5.7.8. Consider the reservation table of a static pipeline shown in Figure 5.38.



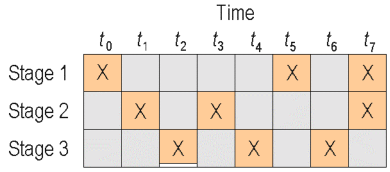
Time

	t_0	t_1	t_2	t_3	t_4
Stage 1	X				X
Stage 2		X		X	
Stage 3			X		

Figure 5.38. Reservation table of a three-stage static pipeline for Problem 5.7.8.

(*a*) Write the forbidden list; (*b*) Draw the state diagram; (*c*) Calculate the minimum average latency and the minimum latency.

5.7.9. Consider the reservation table of a static pipeline shown in Figure 5.39. (*a*) Write the forbidden list; (*b*) Draw the state diagram; (*c*) Calculate the minimum average latency and the minimum latency; (*d*) Suppose that the pipeline is to be operated with a constant latency L such that the resulting pipeline efficiency is as close to 0.5 as possible. Determine the value of L in this case.

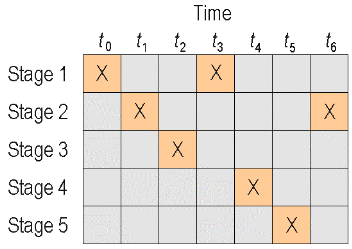


Time

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Stage 1	X					X		X
Stage 2		X		X				X
Stage 3			X		X		X	

Figure 5.39. Reservation table of a three-stage static pipeline for Problem 5.7.9.

5.7.10. Construct the state diagram for the pipeline reservation table shown in Figure 5.40 and calculate the pipeline's minimum average latency.

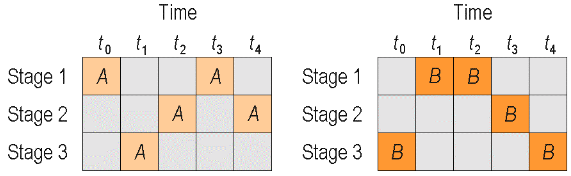


Time

	t_0	t_1	t_2	t_3	t_4	t_5	t_6
Stage 1	X			X			
Stage 2		X					X
Stage 3			X				
Stage 4					X		
Stage 5						X	

Figure 5.40. Reservation table of a five-stage static pipeline for Problem 5.7.10.

5.7.11. Consider the reservation table of a dynamic pipeline shown in Figure 5.41. (*a*) Write the collision matrices; (*b*) Draw the state diagram.



Time

	t_0	t_1	t_2	t_3	t_4
Stage 1	*A*			*A*	
Stage 2			*A*		*A*
Stage 3		*A*			

Time

	t_0	t_1	t_2	t_3	t_4
Stage 1		*B*	*B*		
Stage 2				*B*	
Stage 3	*B*				*B*

Figure 5.41. Reservation table of a three-stage dynamic pipeline for Problem 5.7.11.

5.7.12. Consider a six-stage instruction pipeline with the following stages: Instruction fetch (IF), Instruction decode (ID), Execute 1 (E1), Execute 2 (E2), Execute 3 (E3), and Write-back (WB). The following code fragment of a RISC processor is to be executed in this pipeline:

```
LOAD   R4, #A        ; load constant A into R4
LOAD   R5, #B        ; load constant B into R5
ADD    R8, R4, R5    ; R8 = R4 + R5
STORE  (R1), R8      ; store R8 in the memory location
                     ; addressed by R1
LOAD   R6, #C        ; load constant C into R6
```

```
       ADD    R9, R5, R6    ; R9 = R5 + R6
       STORE  (R2), R9      ; store R9 in the memory location
                            ; addressed by R2
```

(*a*) Show the execution of this program in the style of Figure 5.6, and determine the number of clock cycles needed for its complete execution. (*b*) Determine a valid reordering of the program that will reduce its execution time. Show the execution of the reordered program.

5.7.13. Assume an instruction pipeline with the following stages: Instruction fetch (IF), Instruction decode (ID), Execution and effective address calculation (EX), Memory access (MEM), Write-back (WB). This pipeline neither stalls nor forwards on hazards, so `NOP` instructions have to be added. (*a*) Rewrite the following code inserting as few `NOP` instructions as needed for proper execution:

```
COPY:  LOAD   R1, 100(R2)   ; load R1 from memory location
                            ; addressed by R2+100
       STORE  200(R2), R1   ; store R1 in the memory location
                            ; addressed by R2+200
       ADD    R2, R2, #-4   ; R2 = R2 - 4
       BNE    R2, 0, COPY   ; if R2 # 0 then go to COPY
```

(*b*) Reorder the instructions, if possible, to minimize the number of `NOP` instructions while preserving correctness.

5.7.14. Consider the logic diagram for a pipelined 3 × 3-bit multiplier presented in Figure 5.31. (*a*) The six unconnected terminals of some of the *M* cells are redundant in that they always carry the logic value 0. Certain connected lines are also redundant in this sense and are included only to make the stages uniform. Identify all such redundant connections. (*b*) Consider a general $n \times n$ version of this pipelined multiplier. Assuming that the stages are identical and are labeled S_0, S_1, …, S_{n-1}, show that the total number of 1-bit buffer registers of type *R* needed is $3n^2 - n$.

5.7.15. Design a pipelined serial adder that adds four unsigned binary numbers presented serially (with the least significant bits first) and produces their sum, also in serial form. The adder has four input lines *x*, *y*, *z*, *u* and a single output line *S*.

5.7.16. Consider the pipelined serial adder designed for Problem 5.7.15. Suppose that the adder is reset in clock cycle 0. The least significant bits of four integer numbers N_1, N_2, N_3, and N_4 to be added are applied to the adder in clock cycle 1, and four new data bits are applied in each subsequent clock cycle. If each number consists of 32 bits, each with the value 1, therefore representing $2^{32} - 1$, in what clock cycle will the most significant bit of the sum $N_1+N_2+N_3+N_4$ be loaded into the output *S* flip-flop?

5.7.17. Consider again the pipelined serial adder designed for Problem 5.7.15. Table 5.2 shows the data entered into the pipeline at the indicated times. Determine the value of S for each clock cycle.

Table 5.2. The data entered into the pipelined serial adder for Problem 5.7.16.

Clock cycle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
x	0	1	0	1	1	0	0	0	0
y	0	1	1	1	1	0	0	0	0
z	0	0	0	1	1	0	0	0	0
u	0	1	1	1	0	0	0	0	0
S									

5.7.18. A floating-point pipeline has five stages S_1, S_2, S_3, S_4, and S_5, whose delays are 30, 20, 25, 20, and 35 ns, respectively. What is the pipeline's maximum throughput in millions of floating-point operations per second (MFLOPS)?

5.7.19. In digital signal processing it is sometimes necessary to multiply a high-speed stream of n-bit numbers Y_1, Y_2, Y_3 … by a single number X. The output should be a stream of n-bit results Y_1X, Y_2X, Y_3X, … moving at the same rate as the input stream. Assuming that X and Y_i are positive n-bit binary integers, design a pipeline to carry out this type of multiplication efficiently.

5.7.20. Let $X = x_0, x_1, \ldots, x_{n-1}$ and $Y = y_0, y_1, \ldots, y_{n-1}$ be two fixed-point vectors of length n. The double-length vector $Z = z_0, z_1, \ldots, z_{2n-2}, z_{2n-1}$ defined by

$$z_i = \sum_{j=0}^{n-i} x_i \times y_{i-j}$$

where $x_j = y_j = 0$ if $j < 0$ is called *convolution* of X and Y. This operation is useful in applications such as digital signal processing. Design a one-dimensional systolic array to implement convolution. The array should have the general structure of a pipeline with the X, Y, and Z vectors flowing horizontally. Describe the function of the processing cell and draw a diagram to illustrate the operation of the systolic array in the style of Figure 5.33.