7.10 Datapaths

Datapaths are used in all standard processor and ASIC implementations to perform complex numerical computation or data manipulations.

A datapath consists of some temporary storage in addition to arithmetic, logic, and shift units.

Example: $sum = \sum_{i=1}^{100} x_i$

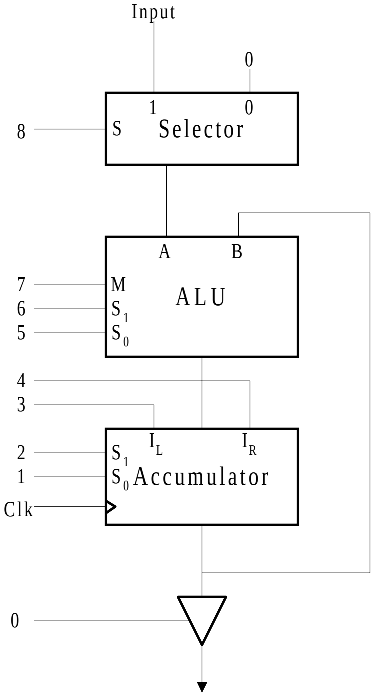
```
   sum = 0
loop:
   for i = 1 to 100
      sum = sum + x
   end loop
```

loop:

The selector selects either 0 or some outside data as the left operand for the ALU.
The right operand for the ALU is always the content of the accumulator.
The accumulator is a shift register with a parallel load.

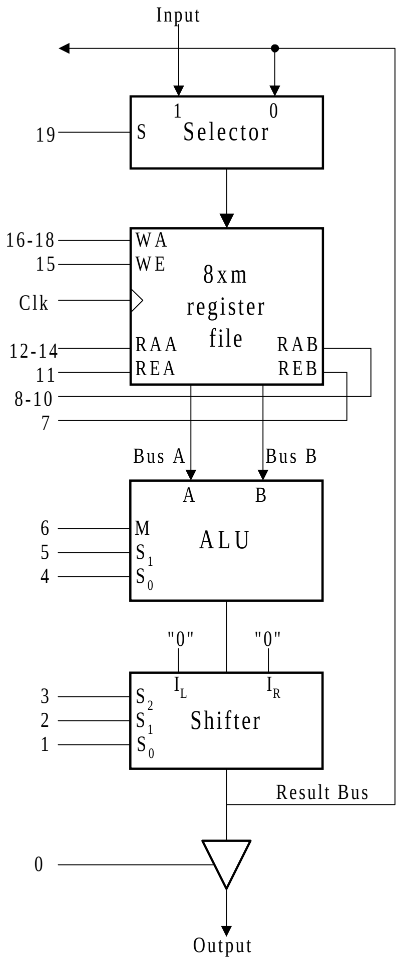
8	7	6	5	4	3	2	1	0
Input select	ALU controls			Shift values		Accumulator controls		Output enable

9-bit control word



Datapath schematic

A more complex datapath



M	S_1	S_0	ALU operstions
0	0	0	Complement A
0	0	1	AND
0	1	0	XOR
0	1	1	OR
1	0	0	Decrement A
1	0	1	Add
1	1	0	Subtract
1	1	1	Increment A

ALU operations

S_2	S_1	S_0	Shift operstions
0	0	0	Pass
0	0	1	Pass
0	1	0	Not used
0	1	1	Not used
1	0	0	Shift left
1	0	1	Rotate left
1	1	0	Shift right
1	1	1	Rotate right

Shift operations

19	18 17 16 15	14 13 12 11	10 9 8 7	6 5 4	3 2 1	0
In E	Write address	Read address A	Read address B	ALU operations	Shift operations	O E

20-bit control word

Example: A one’s counter that will count the number of 1’s in an input data word and return the result after completion.

```
1.  Data := input
2.  Ocount := 0
3.  Mask := 1
While Data ≠ 0 repeat
    4.  Temp := Data AND Mask
    5.  Ocount := Ocount + Temp
    6.  Data := Data >> 1
    End while
7.  Output := Ocount
```

Algorithm for one’s count

R_1:	Data
R_2:	Mask
R_3:	Ocount
R_4:	Temp

Register assignment

Control words	IE	Write address	Read address A	Read Address B	ALU operation	Shifter operation	OE
1	1	R_1	X	X	X	X	0
2	0	R_3	0	0	Add	Pass	0
3	0	R_2	0	X	Increment	Pass	0
4	0	R_4	R_1	R_2	AND	Pass	0
5	0	R_3	R_3	R_4	Add	Pass	0
6	0	R_1	R_1	0	Add	Shift right	0
7	0	None	R_3	0	Add	Pass	1

Repeat While Data ≠ 0

Control words for one’s counter

The one’s-counter controller (FSM) has:

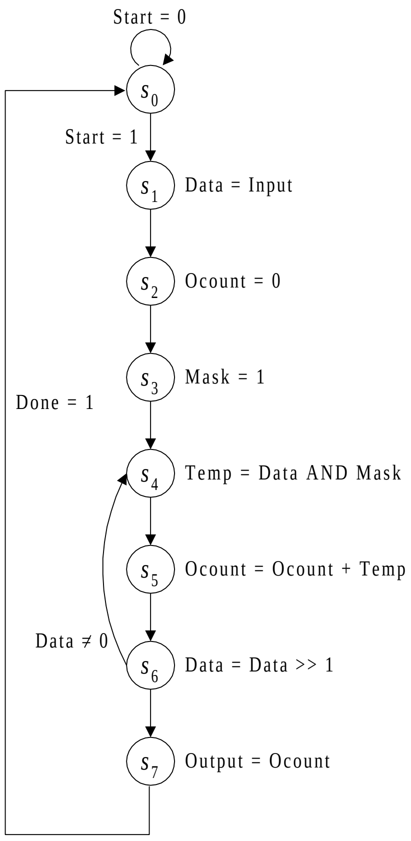
- Two input signals, *Start* and *Data* = 0, and one output signal, *Done*.
- Eight states
- On each clock cycle, the FSM will supply 20 control signals in the control word.

States	$Q_2\ Q_1\ Q_0$	Start, Data=0			
		00	01	10	11
s_0	0 0 0	0 0 0	0 0 0	0 0 1	0 0 1
s_1	0 0 1	0 1 0	0 1 0	0 1 0	0 1 0
s_2	0 1 0	0 1 1	0 1 1	0 1 1	0 1 1
s_3	0 1 1	1 0 0	1 0 0	1 0 0	1 0 0
s_4	1 0 0	1 0 1	1 0 1	1 0 1	1 0 1
s_5	1 0 1	1 1 0	1 1 0	1 1 0	1 1 0
s_6	1 1 0	1 0 0	1 1 1	1 0 0	1 1 1
s_7	1 1 1	0 0 0	0 0 0	0 0 0	0 0 0

Next-state table

Q	Q_{next}	D
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-Flop excitation table



FSM

Using D flip-flops to implement the next-state table, the resulting implementation table is the same as the original next-state table.

	$Q_2 = 0$				$Q_2 = 1$			
Start Data=0 \ $Q_1 Q_0$	00	01	11	10	00	01	11	10
00	0 000	1 010	3 100	2 011	0 101	1 110	3 000	2 100
01	4 000	5 010	7 100	6 011	4 101	5 110	7 000	6 111
11	12 001	13 010	15 100	14 011	12 101	13 110	15 000	14 111
10	8 001	9 010	11 100	10 011	8 101	9 110	11 000	10 100

Q_2(next), Q_1(next), Q_0(next)

Figure 7.29 (a) Karnaugh Map representation

$$Q_2\,(next) = Q_2'Q_1Q_0 + Q_2Q_1' + Q_2Q_0'$$
$$Q_1\,(next) = Q_1'Q_0 + Q_2'Q_1Q_0' + (Data = 0)Q_1Q_0'$$
$$Q_0\,(next) = Q_2'Q_1Q_0' + Q_2Q_1'Q_0' + StartQ_1'Q_0' + (Data = 0)Q_2Q_0'$$

Figure 7.29 (b) Next-state equations

Figure 7.30 (a) Output logic table

			Write address	Read address A	Read address B	ALU	Shifter	
State	$Q_2\ Q_1\ Q_0$	IE	$WA_2\ WA_1\ WA_0\ WE$	$RA_2\ RA_1\ RA_0\ RE$	$RB_2\ RB_1\ RB_0\ RE$	$M\ S_1\ S_0$	$S_2\ S_1\ S_0$	OE
s_0	0 0 0	0	X X X 0	X X X 0	X X X 0	X X X	X X X	0
s_1	0 0 1	1	0 0 1 1	X X X 0	X X X 0	X X X	X X X	0
s_2	0 1 0	0	0 1 1 1	X X X 0	X X X 0	1 0 1	0 0 0	0
s_3	0 1 1	0	0 1 0 1	X X X 0	X X X 0	1 1 1	0 0 0	0
s_4	1 0 0	0	1 0 0 1	0 0 1 1	0 1 0 1	0 0 1	0 0 0	0
s_5	1 0 1	0	0 1 1 1	0 1 1 1	1 0 0 1	1 0 1	0 0 0	0
s_6	1 1 0	0	0 0 1 1	0 0 1 1	X X X 0	1 0 1	1 1 0	0
s_7	1 1 1	0	X X X 0	0 1 1 1	X X X 0	1 0 1	0 0 0	1

Figure 7.30 (b) Output equations

$$IE = Q_2'Q_1'Q_0$$
$$WA_2 = Q_1'Q_0'$$
$$WA_1 = Q_2Q_0 + Q_2'Q_1$$
$$WA_0 = Q_1'Q_0 + Q_1Q_0'$$
$$WE = Q_2Q_1' + Q_2'Q_0 + Q_1Q_0'$$

$$RAA_2 = 0$$
$$RAA_1 = Q_0$$
$$RAA_0 = 1$$
$$REA = Q_1$$

$$RAB_2 = Q_0$$
$$RAB_1 = Q_0'$$
$$RAB_0 = 0$$
$$REB = Q_2Q_1'$$

$$M = Q_1 + Q_0$$
$$S_1 = Q_2'Q_0$$
$$S_0 = 1$$
$$S_2 = S_1 = Q_2Q_1Q_0'$$
$$S_0 = 0$$
$$OE = Q_2Q_1Q_0$$

Example: Question 7.20: For the datapath in Figure 7.26, develop a field-insertion algorithm and control words for all the statements. Assume that the datapath is 8 bits wide and that the field-insertion algorithm inserts the four least-significant bits of the source word into the middle of the destination word: for example, your field-insertion algorithm should take two words, A=$a_7a_6a_5a_4a_3a_2a_1a_0$ and B=$b_7b_6b_5b_4b_3b_2b_1b_0$, and generate the resultant word C=$b_7b_6a_3a_2a_1a_0b_1b_0$ by replacing $b_5b_4b_3b_2$ with $a_3a_2a_1a_0$.

Solution:

Ctr words	IE	Write address	Read address A	Read address B	ALU operation	Shifter operation	OE	Comment	Register content
1	1	R_1	X	X	X	X	0	Load A	$a_7a_6a_5a_4a_3a_2a_1a_0$
2	0	R_1	R_1	0	Add	Shift left	0	Shift left A 4 bits	$a_3a_2a_1a_0$0000
3	0	R_1	R_1	0	Add	Shift left	0		
4	0	R_1	R_1	0	Add	Shift left	0		
5	0	R_1	R_1	0	Add	Shift left	0		
6	0	R_1	R_1	0	Add	Rotate right	0	Rotate right A 4 bits	0000$a_3a_2a_1a_0$
7	0	R_1	R_1	0	Add	Rotate right	0		
8	0	R_1	R_1	0	Add	Rotate right	0		
9	0	R_1	R_1	0	Add	Rotate right	0		
10	1	R_2	X	X	X	X	0	Load B	$b_7b_6b_5b_4b_3b_2b_1b_0$
11	0	R_2	R_2	0	Add	Rotate right	0	Rotate right B 2 bits	$b_1b_0b_7b_6b_5b_4b_3b_2$
12	0	R_2	R_2	0	Add	Rotate right	0		
13	0	R_2	R_2	0	Add	Shift right	0	Shift right B 4 bits	0000$b_1b_0b_7b_6$
13	0	R_2	R_2	0	Add	Shift right	0		
13	0	R_2	R_2	0	Add	Shift right	0		
13	0	R_2	R_2	0	Add	Shift right	0		
14	0	R_2	R_2	0	Add	Rotate left	0	Rotate left B 4 bits	$b_1b_0b_7b_6$0000
15	0	R_2	R_2	0	Add	Rotate left	0		
16	0	R_2	R_2	0	Add	Rotate left	0		
17	0	R_2	R_2	0	Add	Rotate left	0		
18	0	R_1	R_1	R_2	OR	Pass	0	A ← A OR B	$b_1b_0b_7b_6a_3a_2a_1a_0$
19	0	R_1	R_1	0	Add	Rotate left	0	Rotate left A 2 bits	$b_7b_6a_3a_2a_1a_0b_1b_0$
20	0	R_1	R_1	0	Add	Rotate left	0		
21	0	None	R_1	0	Add	Pass	1	Output A	

Control words for the field-insertion algorithm