6004 Spring 1998: L18

Control Structures

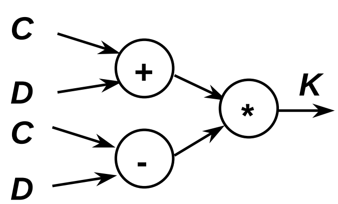
1

Motivation: Computing Expressions

Given an expression:

$K = (C + D) * (C - D)$

Can build a circuit for the expression



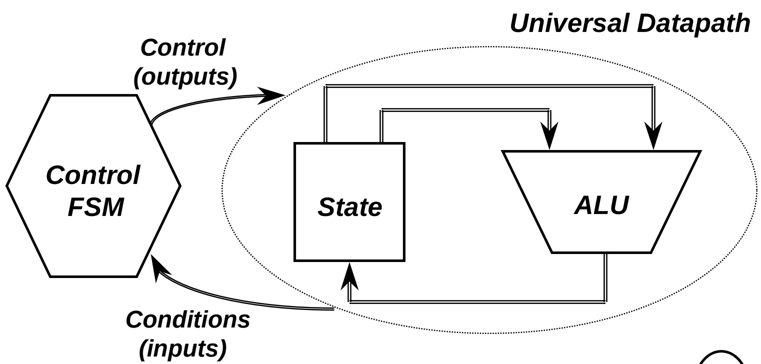
Circuit size α **size of expression**

2

6004 Spring 1998: L18

Abstraction: Control/Datapath

Sequence a single function unit through expression

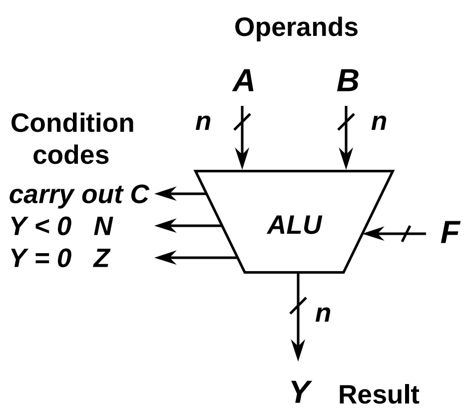


Size of datapath is constant

3

Arithmetic/Logic Unit (ALU)

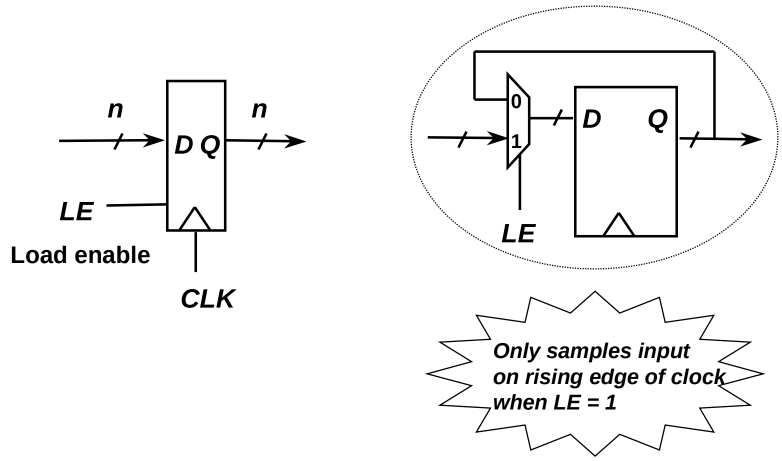
Purely combinational logic



F	Y
000	A + B
001	A - B
010	A - 1
011	A and B
100	A or B
101	A * B
.	
.	

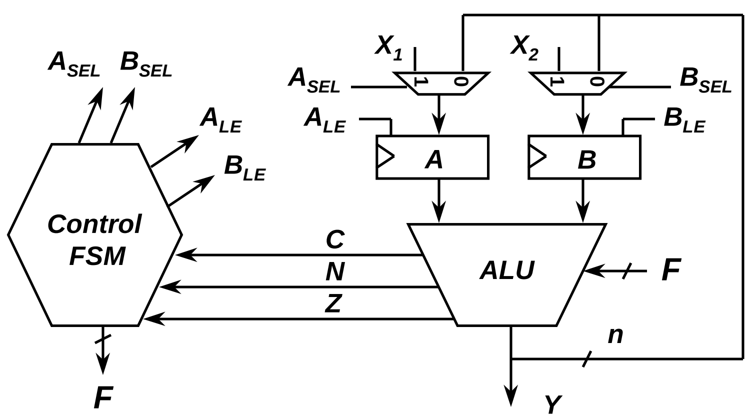
4

Loadable Register



5

Our First Datapath



*How can we compute K = (C + D) * D?*

6

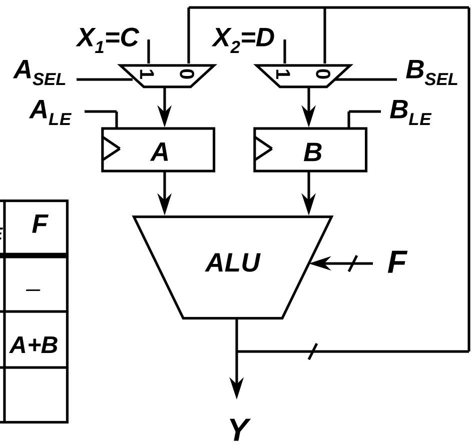
6004 Spring 1998: L18

Control FSM for K = (C + D) * D

Inputs appear at X_1 , X_2 in first clock cycle alone

Control FSM
current state **outputs**

	A_{SEL}	B_{SEL}	A_{LE}	B_{LE}	F
s_1	1	1	1	1	–
s_2	0	0	1	0	A+B
s_3					



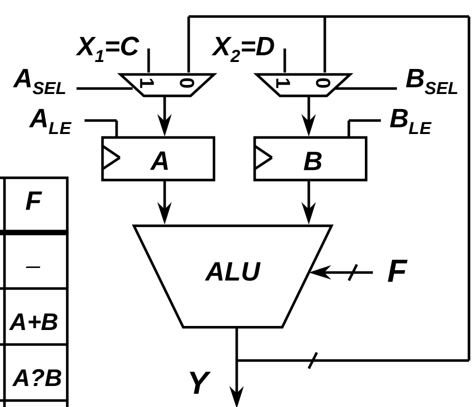
Control FSM has no inputs, merely sequences through states

7

Control FSM for K = (C + D) * (C - D)

Control FSM
current state **outputs**

	A_{SEL}	B_{SEL}	A_{LE}	B_{LE}	F
s_1	1	1	1	1	–
s_2	0	0	1	0	A+B
s_3	0	0	?	?	A?B



Where do we store C, D, C+D?

8

Storage Requirements

- **Closer look at expression**

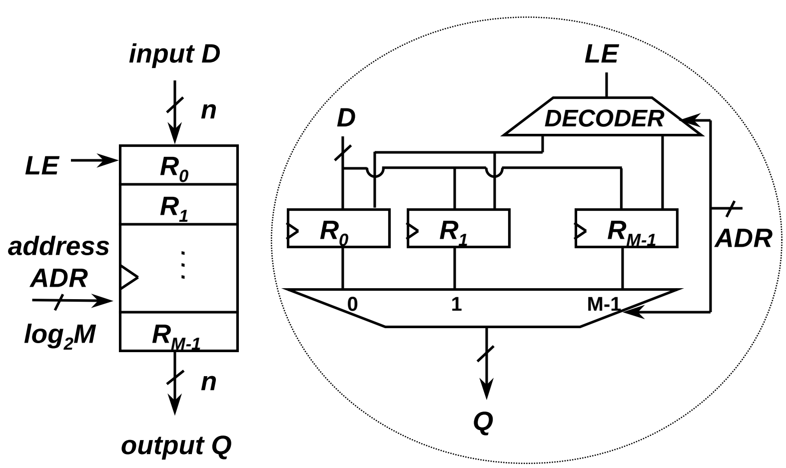
$$K = \underbrace{(C + D)}_{T1} * \underbrace{(C - D)}_{T2}$$

- **Need to store *C*, *D*, *T1*, *T2* and possibly K (though not all at the same time)**

- **Need more than two registers**

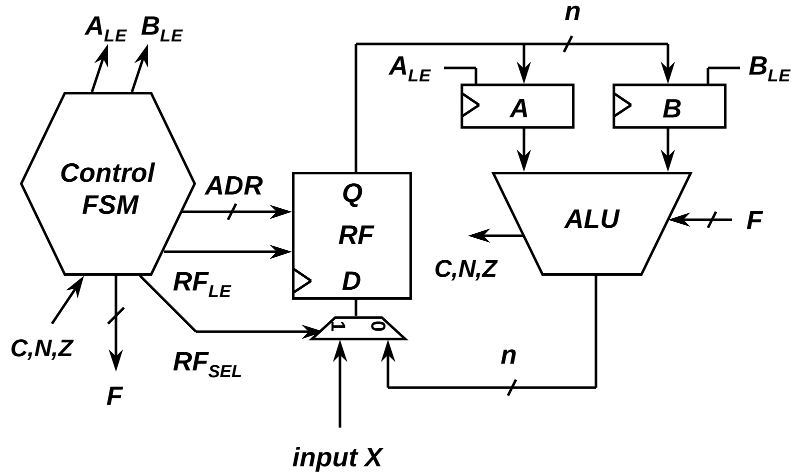
9

Register File (RF) a.k.a. SRAM



10

Our Second Datapath



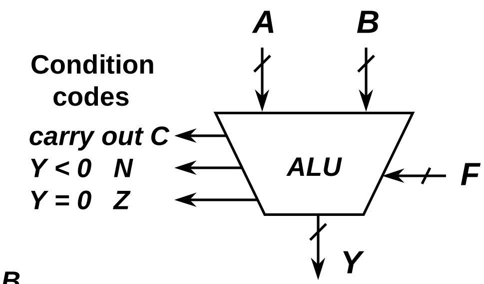
11

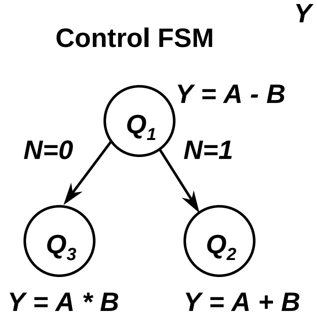
Control FSM for K = (C + D) * (C - D)

	RF_{SEL}	RF_{LE}	ADR	A_{LE}	B_{LE}	F	
s_1	1	1	000	0	0	–	Load input C in R_0
s_2	1	1	001	0	0	–	Load input D in R_1
s_3	0	0	000	1	0	–	Load R_0 value in A
s_4	0	0	001	0	1	–	Load R_1 value in B
s_5							Put C+D in R_0
s_6							Put C-D in R_1
s_7	0	0	000	1	0	–	Load R_0 value in A
s_8	0	0	001	0	1	–	Load R_1 value in B
s_9	0	1	000	0	0	A*B	Put (C+D)*(C-D) in R_0

12

What About Conditionals?



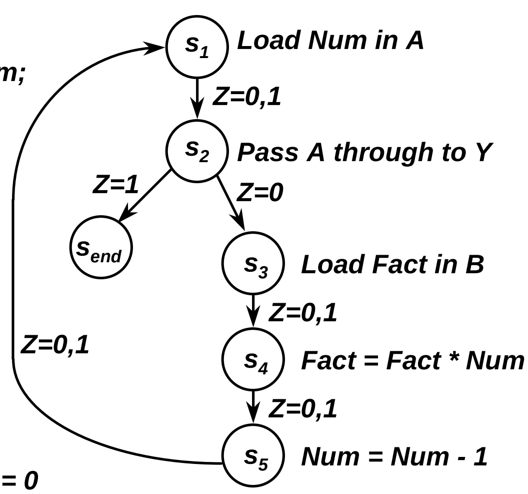


if (A < B) then Y = A + B
else Y = A * B

13

Computing Factorial

```
Fact = 1;
while ( Num > 0 ) {
    Fact = Fact * Num;
    Num = Num - 1;
}
```



Note that $Z=1 \Rightarrow Y = 0$

14

Computing Factorial - II

Fact = 1;
while (Num > 0) {
*Fact = Fact * Num;*
Num = Num - 1;
}

R_0 corresponds to variable Num
R_1 corresponds to variable Fact assumed to be 1 initially

input		outputs							
Z	CS	NS	RF_{SEL}	RF_{LE}	ADR	A_{LE}	B_{LE}	F	
0,1	s_1	s_2	0	0	000	1	0	–	Load Num in A
1	s_2	s_{end}	0	0	–	0	0	A	Num = 0 exit
0	s_2	s_3	0	0	–	0	0	A	Num > 0 continue
0,1	s_3	s_4	0	0	001	0	1	–	Load Fact in B
0,1	s_4	s_5	0	1	001	0	0	A*B	Fact = Fact * Num
0,1	s_5	s_1	0	1	000	0	0	A-1	Num = Num - 1

15

Observe...

- **Same datapath could perform many different tasks (e.g., K = (C + D) * (C - D), Factorial)**
  - **Difference was control FSM**
- **Do we want to write control FSMs for each task we want the datapath to perform? Answer: ________**
- **Do we have a choice?**

16

6004 Spring 1998: L18

Control FSMs Are Tedious and Boring!

- **<u>Register-Transfer Language</u> (RTL) or <u>machine language</u> is an alternative to filling in 1's and 0's in FSM**

Destination Address ⟵ Source Value

<*Address*> denotes value at *Address*

Example:

$$R_2 \leftarrow <R_0> + <R_1>$$

Value stored in R_0 is added to value stored in R_1 and the sum is deposited in R_2

17

Which would you rather write?

$$K = (C + D) * (C - D)$$

Say!
Is this
software!?

	RF_{SEL}	RF_{LE}	ADR	A_{LE}	B_{LE}	F	
s_1	1	1	000	0	0	–	$R_0 \leftarrow C$
s_2	1	1	001	0	0	–	$R_1 \leftarrow D$
s_3	0	0	000	1	0	–	$A \leftarrow <R_0>$
s_4	0	0	001	0	1	–	$B \leftarrow <R_1>$
s_5	0	1	000	0	0	A+B	$R_0 \leftarrow <A> + <B>$
s_6	0	1	001	0	0	A-B	$R_1 \leftarrow <A> - <B>$
s_7	0	0	000	1	0	–	$A \leftarrow <R_0>$
s_8	0	0	001	0	1	–	$B \leftarrow <R_1>$
s_9	0	1	000	0	0	A*B	$R_0 \leftarrow <A> * <B>$

18

RTL or Machine Language Requirements

- **Need abstract description of machine**
  - **Set of registers: R_0, R_1, …**
  - **Set of operations: +, *, or …**
  - **Coded sequence of instructions:**
    $R_1 \leftarrow \langle R_2 \rangle + \langle R_3 \rangle$
- **Need precise understanding of how instructions change machine state**
- **Need to define instructions that affect control flow (conditionals)**

We will see a lot of this in coming weeks

19