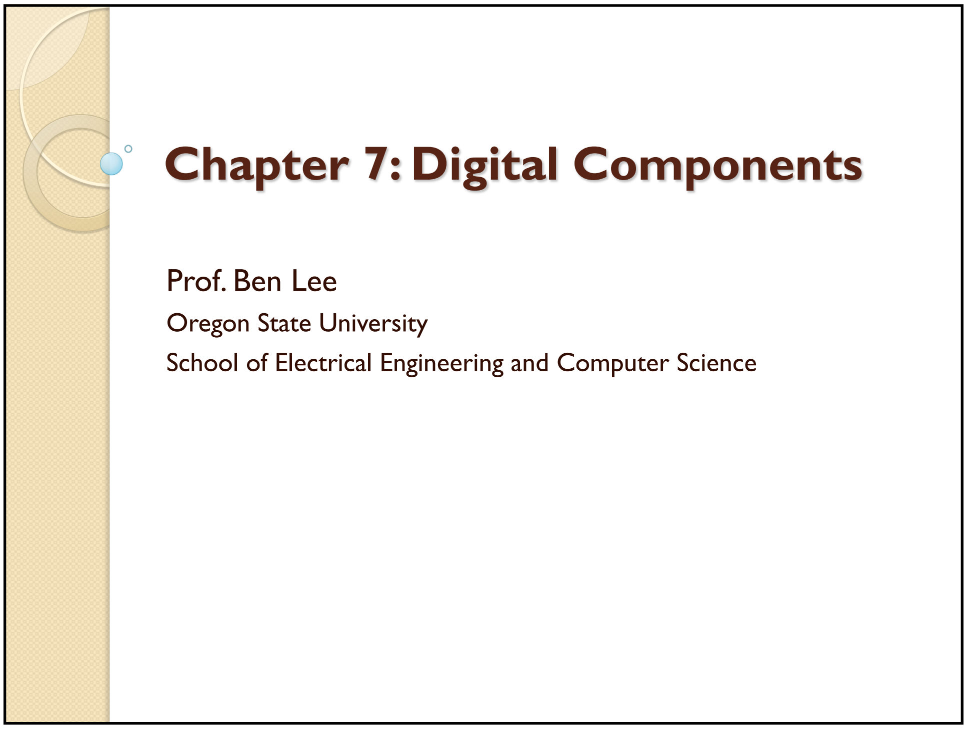
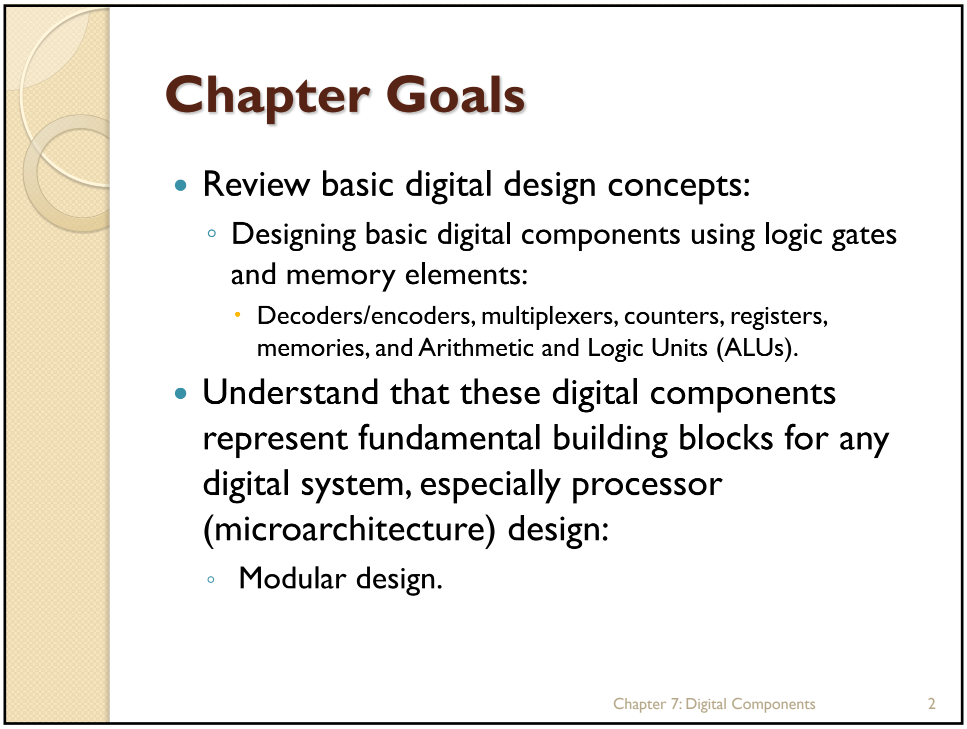


Chapter 7: Digital Components

Prof. Ben Lee
Oregon State University
School of Electrical Engineering and Computer Science

1



Chapter Goals

- Review basic digital design concepts:
  - Designing basic digital components using logic gates and memory elements:
    - Decoders/encoders, multiplexers, counters, registers, memories, and Arithmetic and Logic Units (ALUs).
- Understand that these digital components represent fundamental building blocks for any digital system, especially processor (microarchitecture) design:
  - Modular design.

2

Contents

7.1 Introduction

7.2 Multiplexers

7.3 Decoders

7.4 Memory Elements

7.5 Registers

7.6 Register File

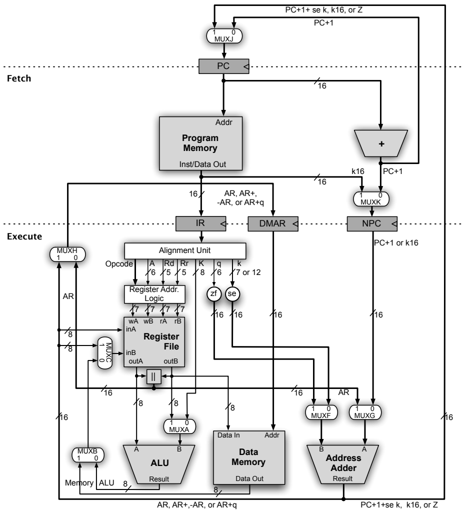
Basic ALU (see Chapter 9)

7.1 Introduction

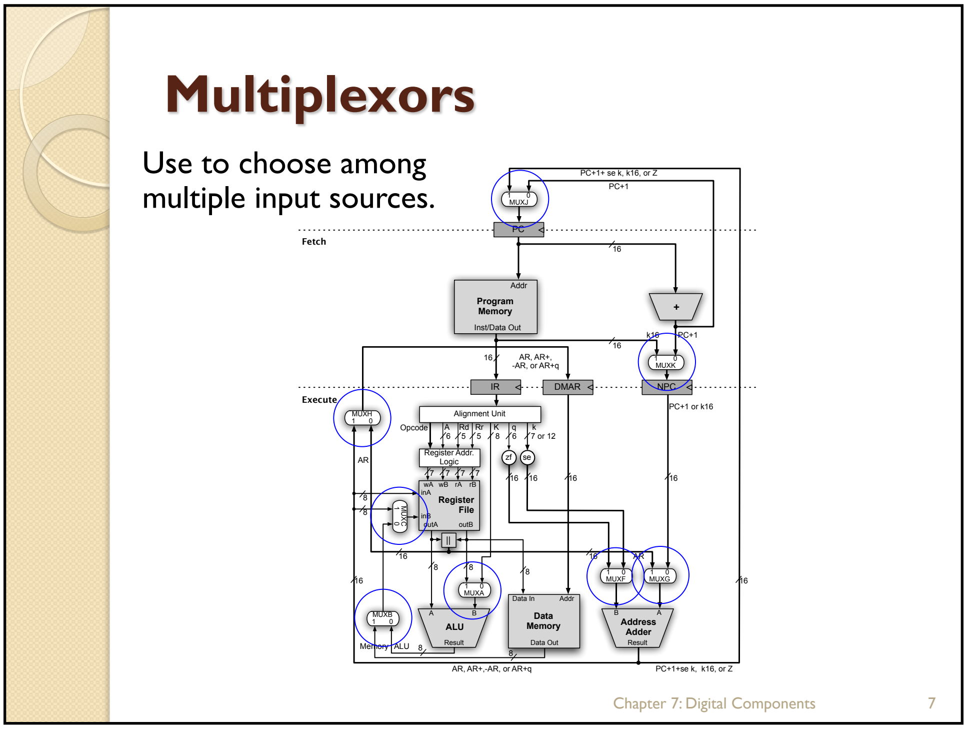
Basic Digital Components

- Multiplexors
- Decoders/encoders
- Registers
- Memories
- …later we will see Arithmetic and Logic Units (ALUs)

AVR Microarchitecture

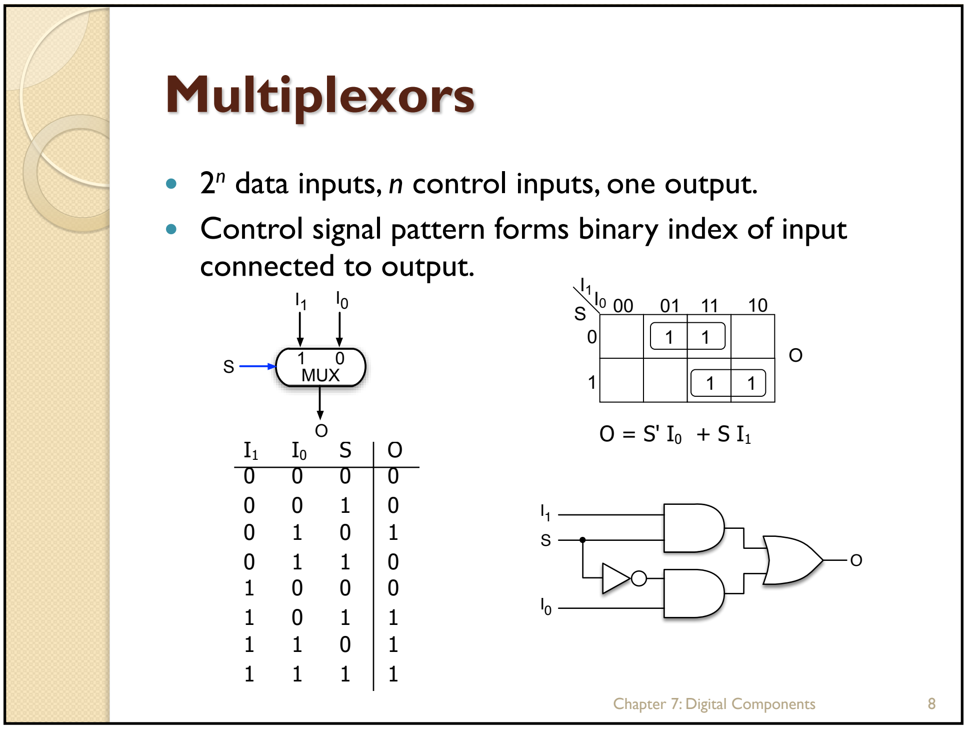


7.2 Multiplexer



Multiplexors

Use to choose among multiple input sources.



Multiplexors

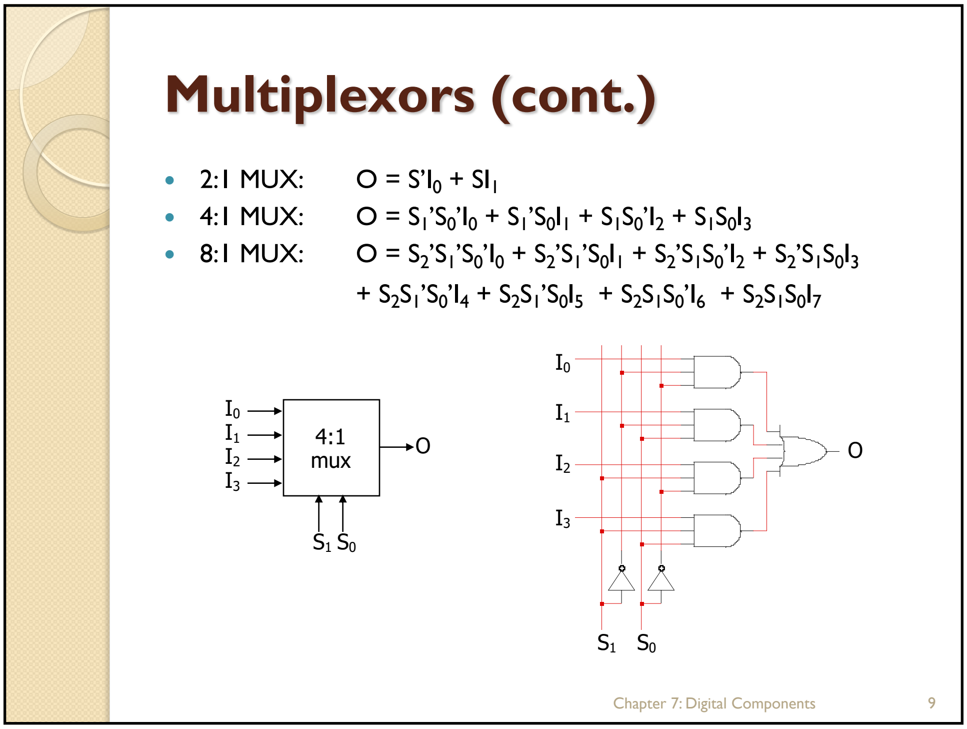
- 2^n data inputs, *n* control inputs, one output.
- Control signal pattern forms binary index of input connected to output.

I_1	I_0	S	O
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

S \ $I_1 I_0$	00	01	11	10
0		1	1	
1			1	1

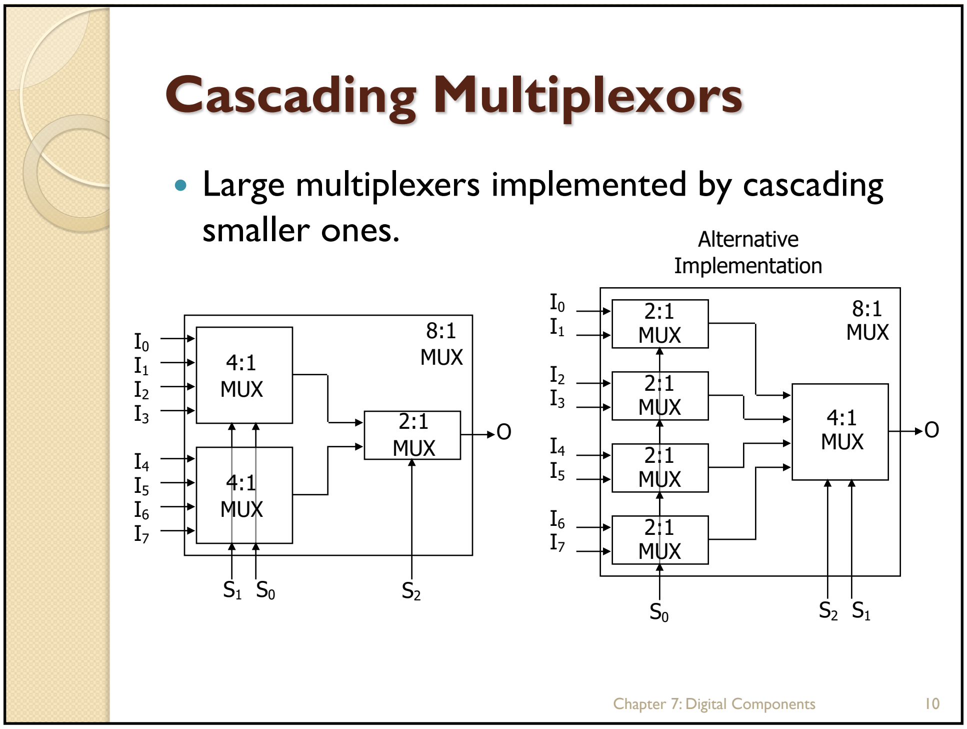
O

$O = S' I_0 + S I_1$



Multiplexors (cont.)

- 2:1 MUX: $O = S'I_0 + SI_1$
- 4:1 MUX: $O = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$
- 8:1 MUX: $O = S_2'S_1'S_0'I_0 + S_2'S_1'S_0I_1 + S_2'S_1S_0'I_2 + S_2'S_1S_0I_3$
$+ S_2S_1'S_0'I_4 + S_2S_1'S_0I_5 + S_2S_1S_0'I_6 + S_2S_1S_0I_7$



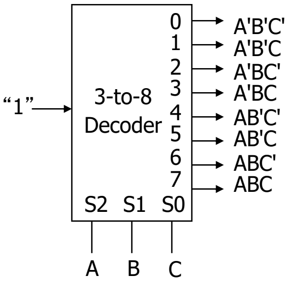
Cascading Multiplexors

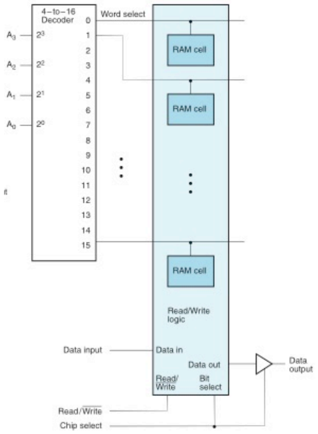
- Large multiplexers implemented by cascading smaller ones.

7.3 Decoder

Decoders

- Single data input, *k* control inputs, 2^k outputs.
- Used to select one of 2^k components:
  - RAM cells in memory
  - Registers in register file



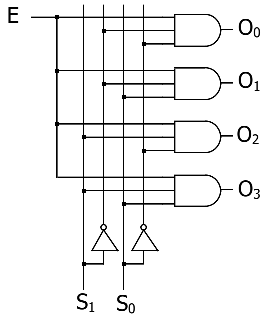


2-to-4 Decoder

- Implementing a 2-to-4 decoder

Truth Table

E	S_1	S_0	O_3	O_2	O_1	O_0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



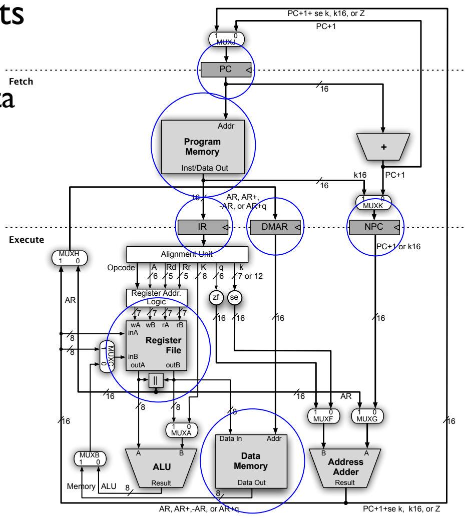
$$O_0 = E \bullet S_1' \bullet S_0'$$
$$O_1 = E \bullet S_1' \bullet S_0$$
$$O_2 = E \bullet S_1 \bullet S_0'$$
$$O_3 = E \bullet S_1 \bullet S_0$$

We can also implement active low enable

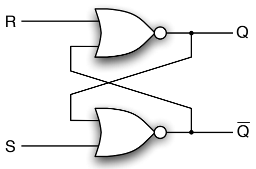
7.4 Memory Elements

Memory Elements

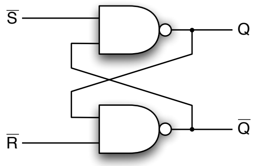
- Bi-stable elements used in
  - Program and Data memories
  - Registers
  - Register File



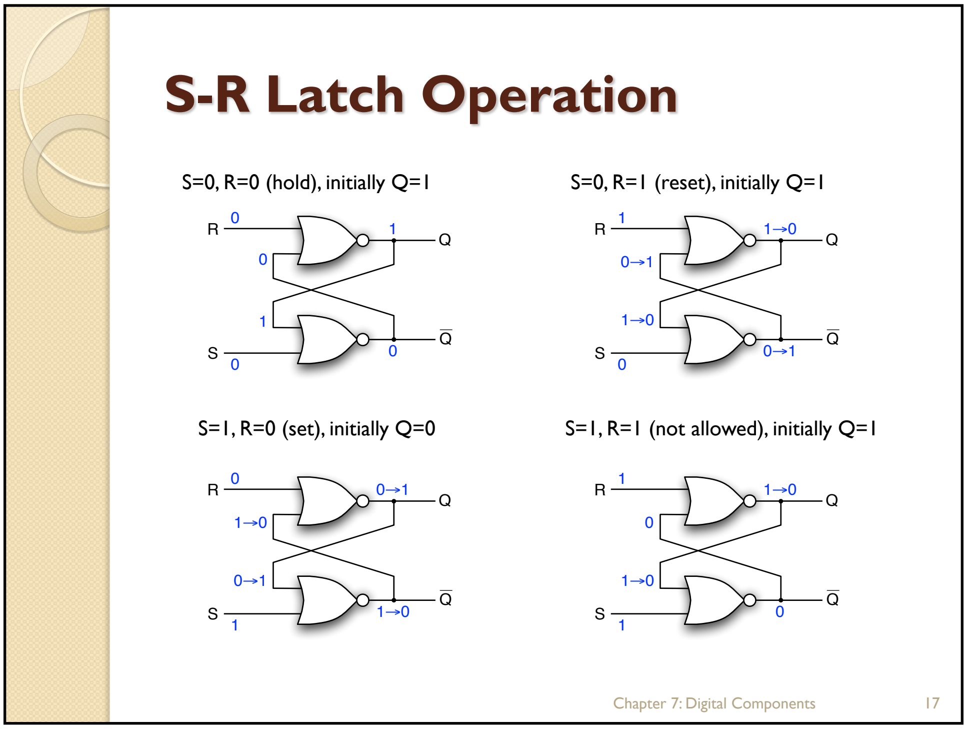
Memory Element: S-R Latch



S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	not allowed



$\overline{S}$	$\overline{R}$	Q
0	0	not allowed
0	1	1
1	0	0
1	1	hold



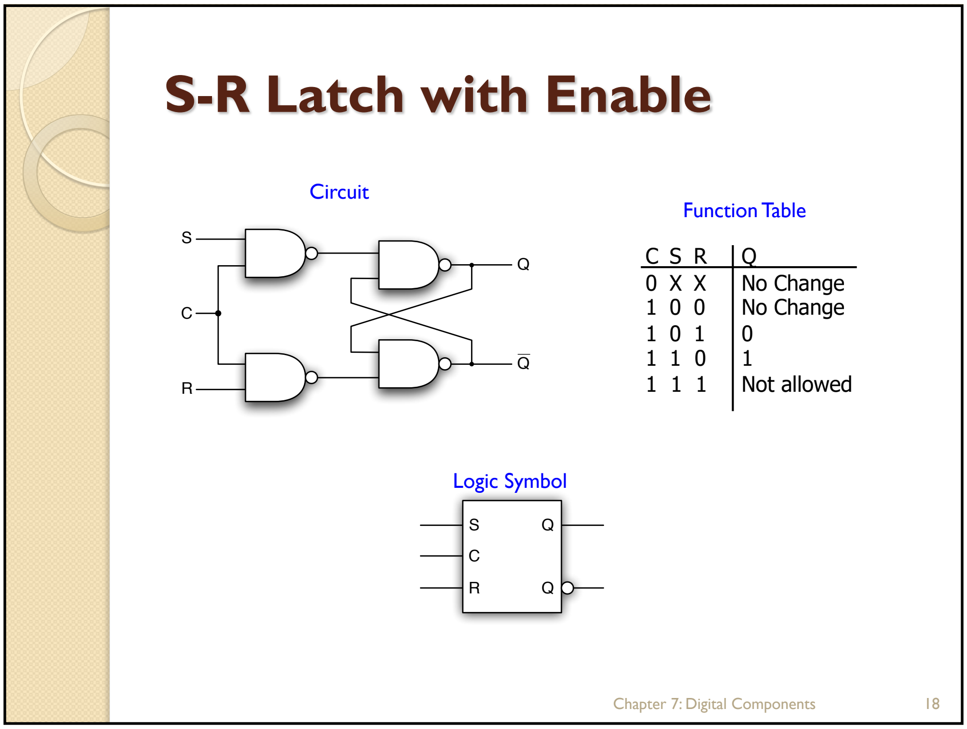
S-R Latch Operation

S=0, R=0 (hold), initially Q=1

S=0, R=1 (reset), initially Q=1

S=1, R=0 (set), initially Q=0

S=1, R=1 (not allowed), initially Q=1



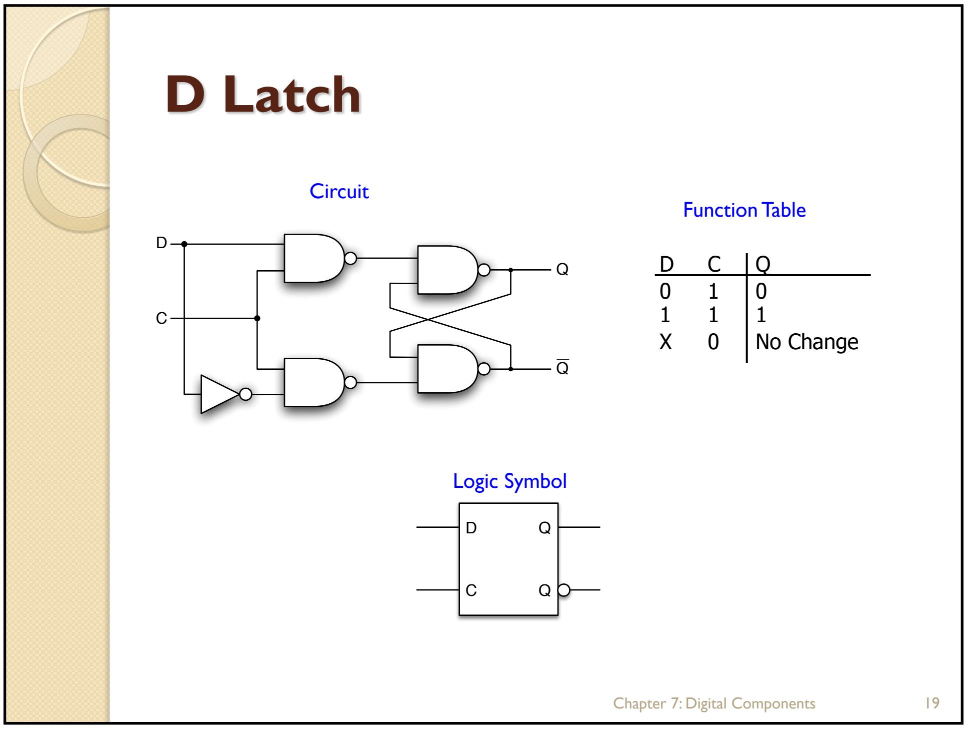
S-R Latch with Enable

Circuit

Function Table

C	S	R	Q
0	X	X	No Change
1	0	0	No Change
1	0	1	0
1	1	0	1
1	1	1	Not allowed

Logic Symbol



D Latch

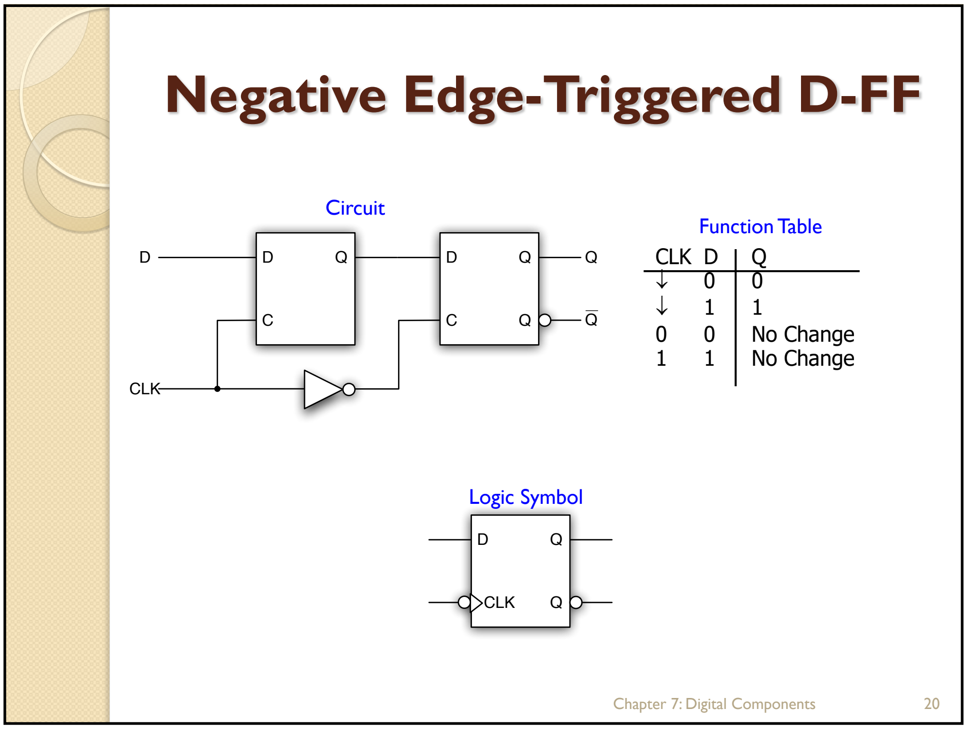
Circuit

Function Table

D	C	Q
0	1	0
1	1	1
X	0	No Change

Logic Symbol

19



Negative Edge-Triggered D-FF

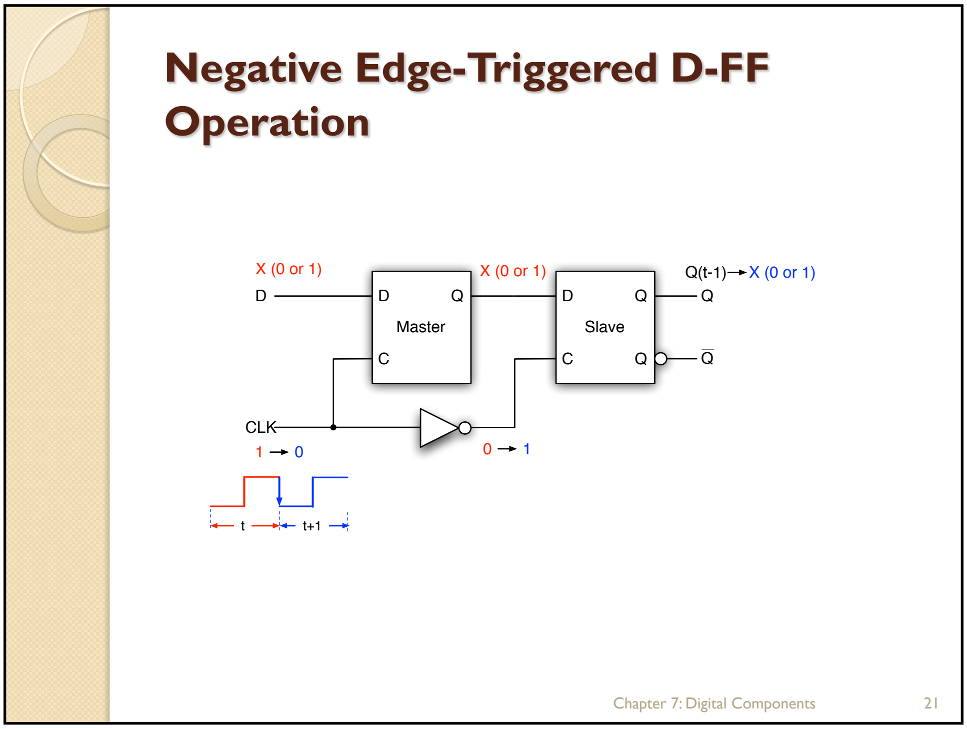
Circuit

Function Table

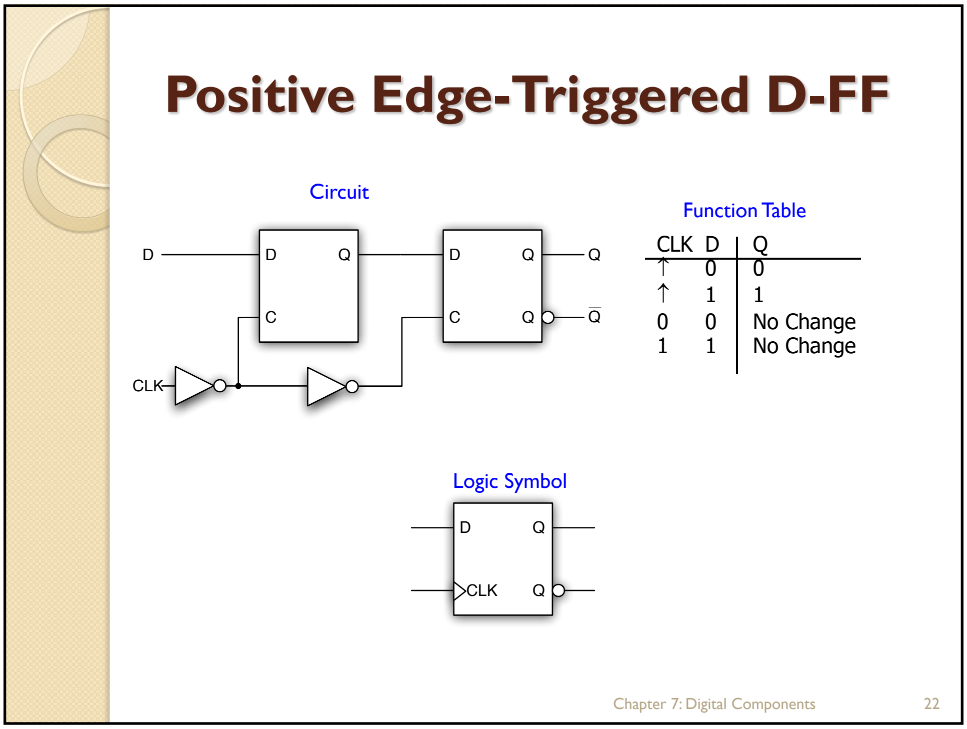
CLK	D	Q
↓	0	0
↓	1	1
0	0	No Change
1	1	No Change

Logic Symbol

20



Negative Edge-Triggered D-FF Operation



Positive Edge-Triggered D-FF

Circuit

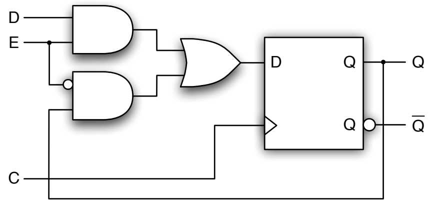
Function Table

CLK	D	Q
↑	0	0
↑	1	1
0	0	No Change
1	1	No Change

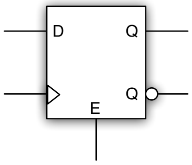
Logic Symbol

Edge-Triggered D-FF /w Enable

Circuit



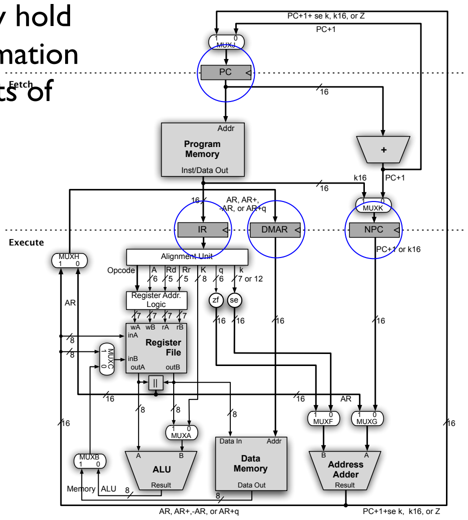
Logic Symbol



7.4 Registers

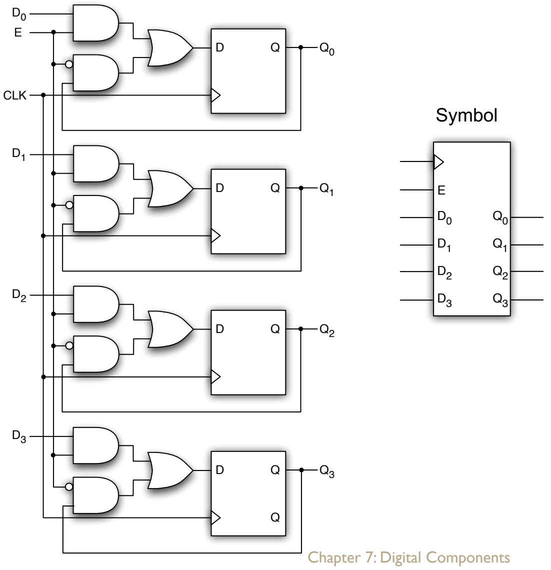
Registers

- Used to temporary hold and separate information among various parts of the datapath.

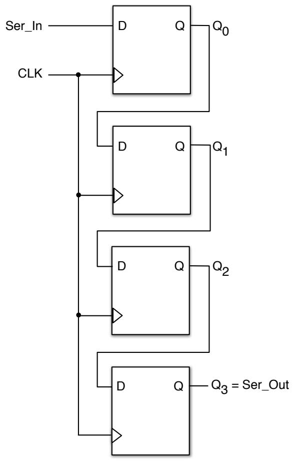


n-bit Register /w Load Enable

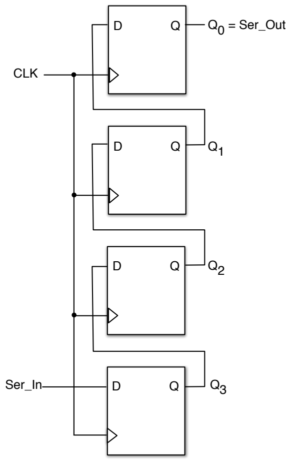
A set of commonly clocked D flip-flops



Shift Registers

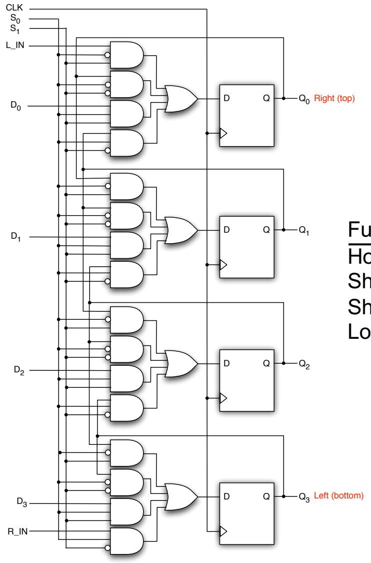


Shift Left (down)



Shift Right (up)

Bi-Directional Shift Register /w Parallel Load

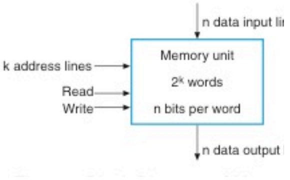


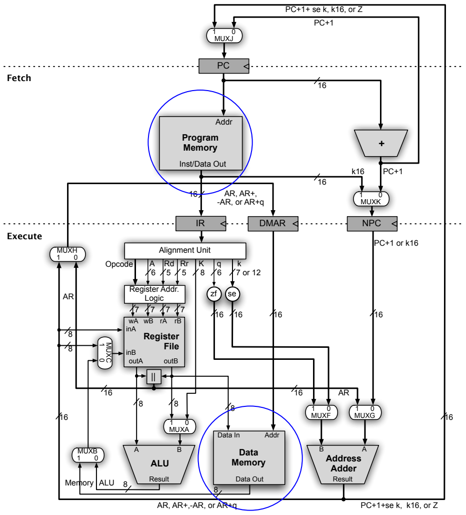
	Control Input		Next State			
Function	S_1	S_0	Q_3	Q_2	Q_1	Q_0
Hold	0	0	Q_3	Q_2	Q_1	Q_0
Shift right	0	1	R_IN	Q_3	Q_2	Q_1
Shift left	1	0	Q_2	Q_1	Q_0	L_IN
Load	1	1	D_3	D_2	D_1	D_0

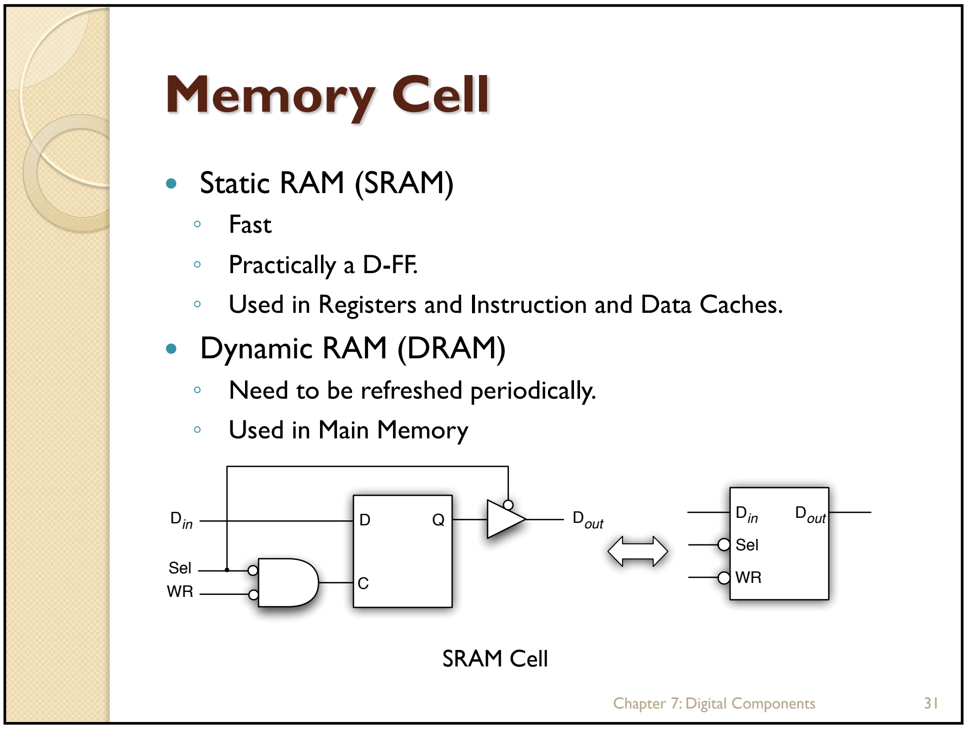
7.5 Memory

Memory

Program and Data memories





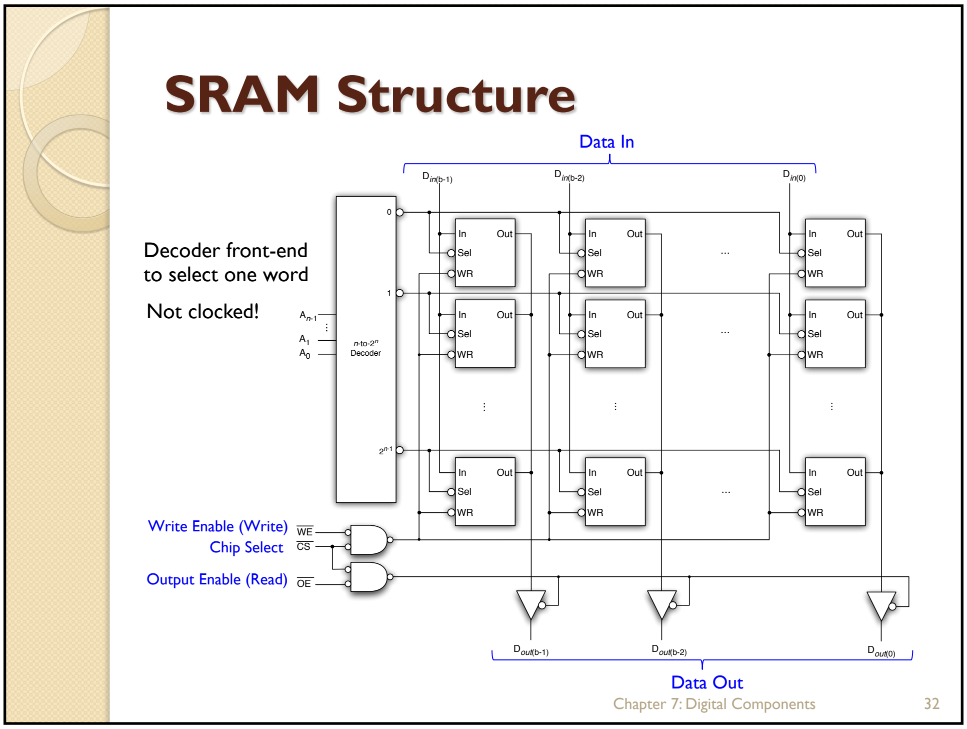


Memory Cell

- Static RAM (SRAM)
  - Fast
  - Practically a D-FF.
  - Used in Registers and Instruction and Data Caches.
- Dynamic RAM (DRAM)
  - Need to be refreshed periodically.
  - Used in Main Memory

D_{in}
Sel
WR
D
Q
C
D_{out}
D_{in}
D_{out}
Sel
WR

SRAM Cell



SRAM Structure

Decoder front-end to select one word

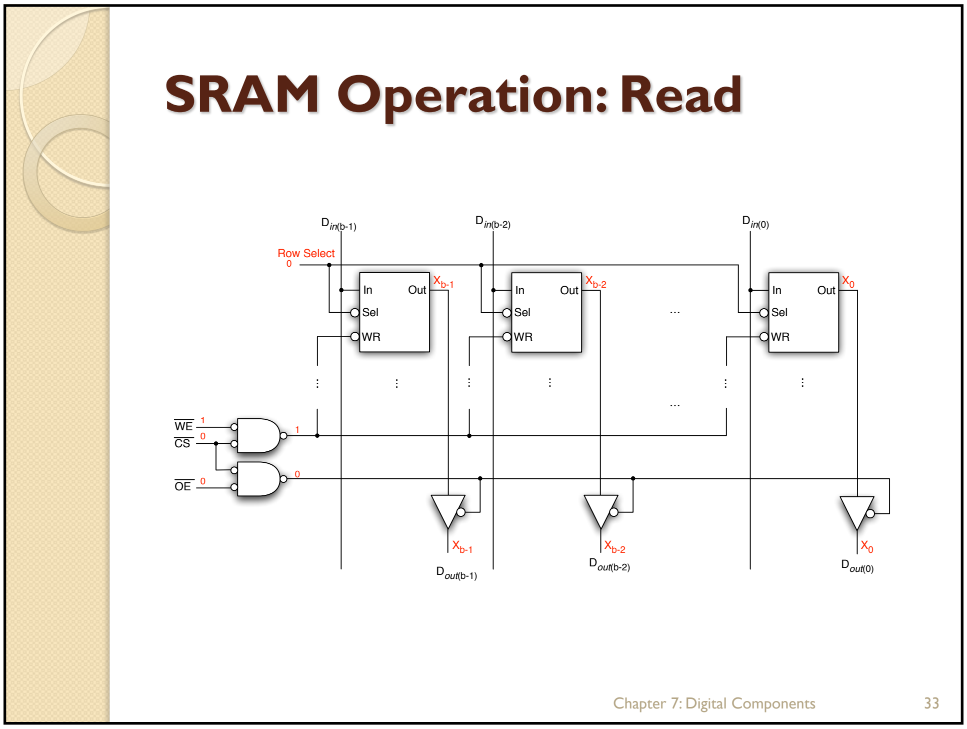
Not clocked!

Write Enable (Write)

Chip Select

Output Enable (Read)

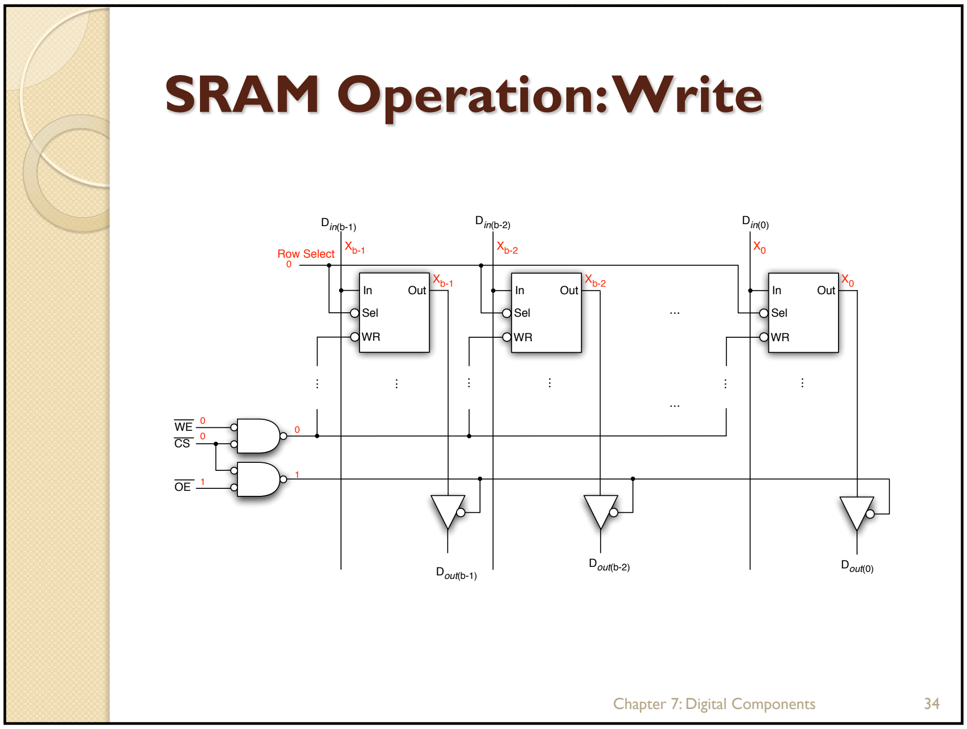
Data In
$D_{in(b-1)}$
$D_{in(b-2)}$
$D_{in(0)}$
0
1
2^{n-1}
A_{n-1}
A_1
A_0
n-to-2^n Decoder
In
Out
Sel
WR
...
$\overline{WE}$
$\overline{CS}$
$\overline{OE}$
$D_{out(b-1)}$
$D_{out(b-2)}$
$D_{out(0)}$
Data Out



SRAM Operation: Read

Chapter 7: Digital Components 33

33

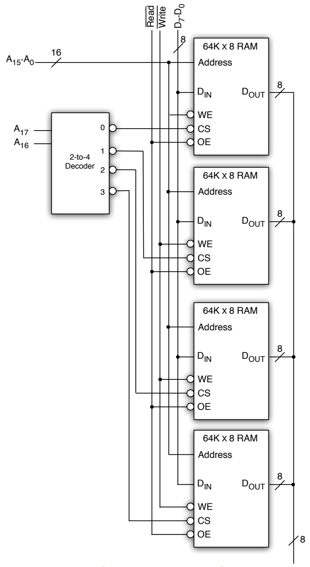


SRAM Operation: Write

Chapter 7: Digital Components 34

34

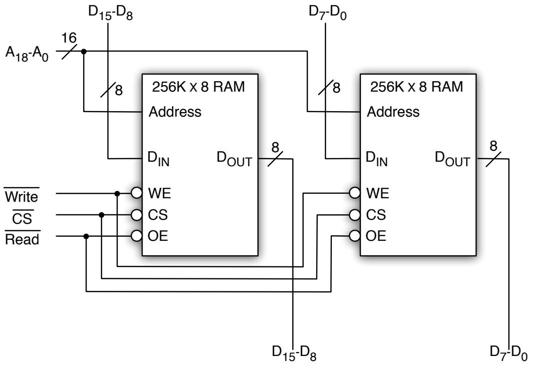
Building Larger Memory



256K x 8 RAM
Using four 64K x 8 RAM

Building Wider Memory

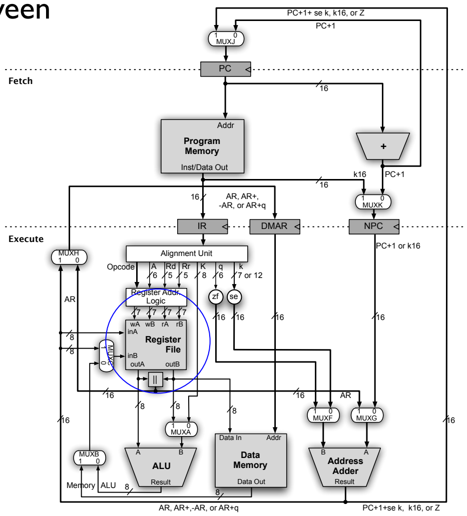
256K x 16 RAM
Using two 256K x 8 RAMs

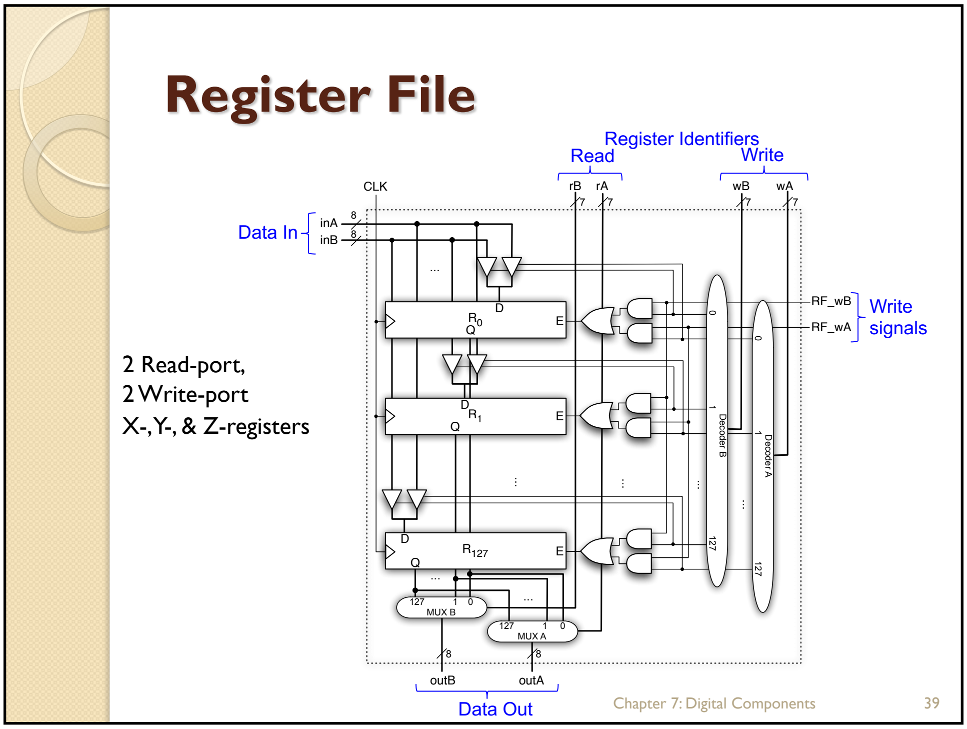


7.6 Register File

Register File

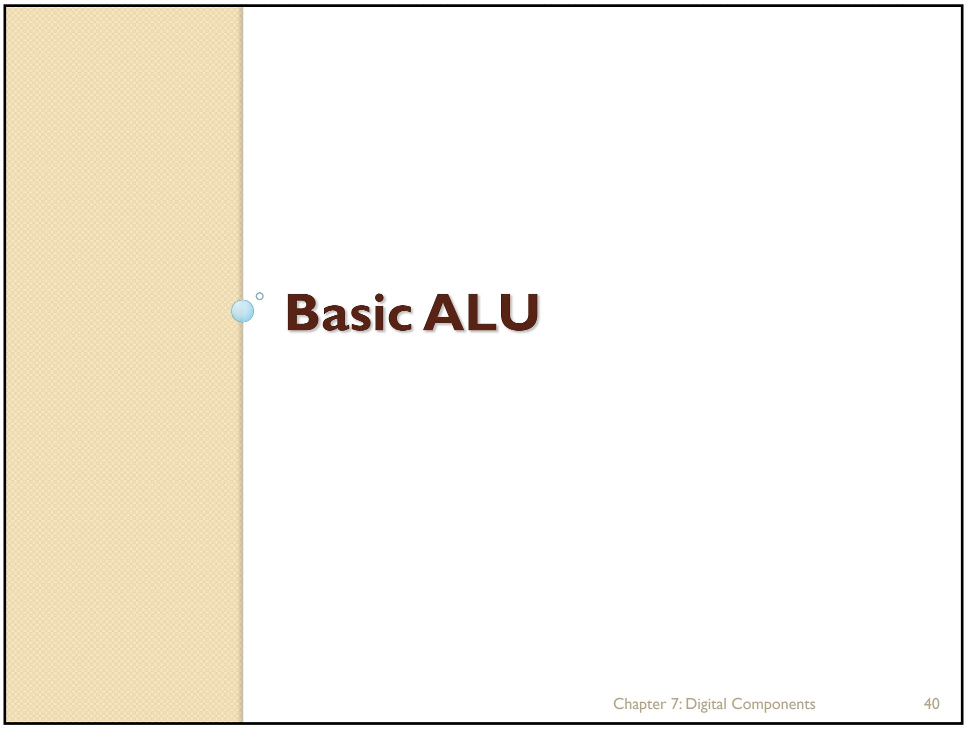
Storage buffer between ALU and Memory



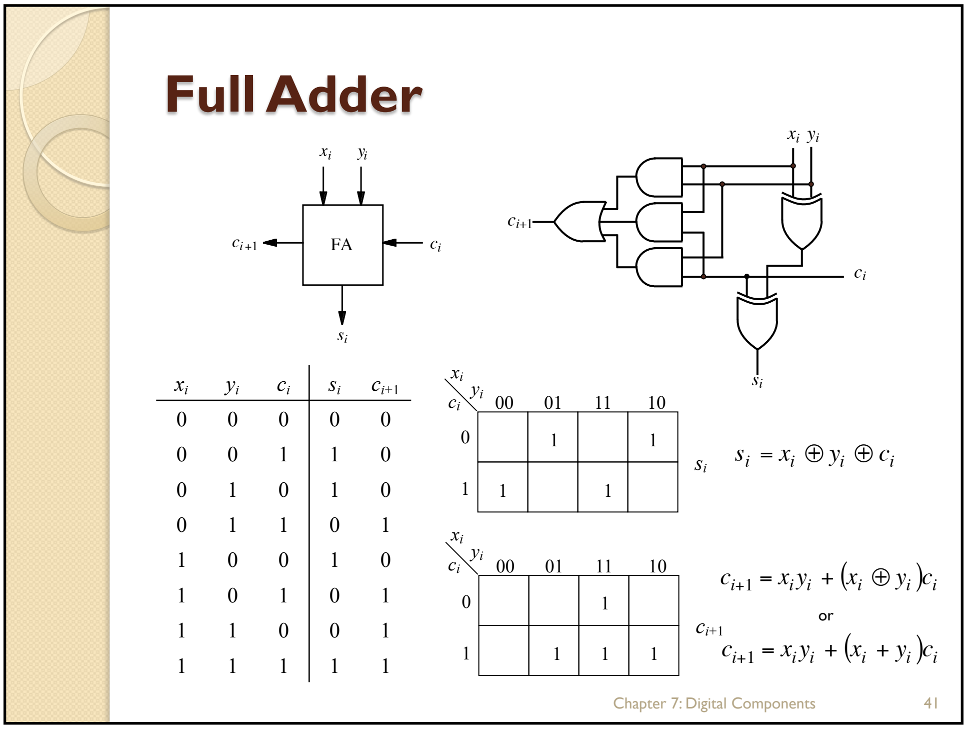


Register File

2 Read-port,
2 Write-port
X-,Y-, & Z-registers



Basic ALU



Full Adder

x_i	y_i	c_i	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

s_i

c_i \ $x_i y_i$	00	01	11	10
0		1		1
1	1		1	

$$s_i = x_i \oplus y_i \oplus c_i$$

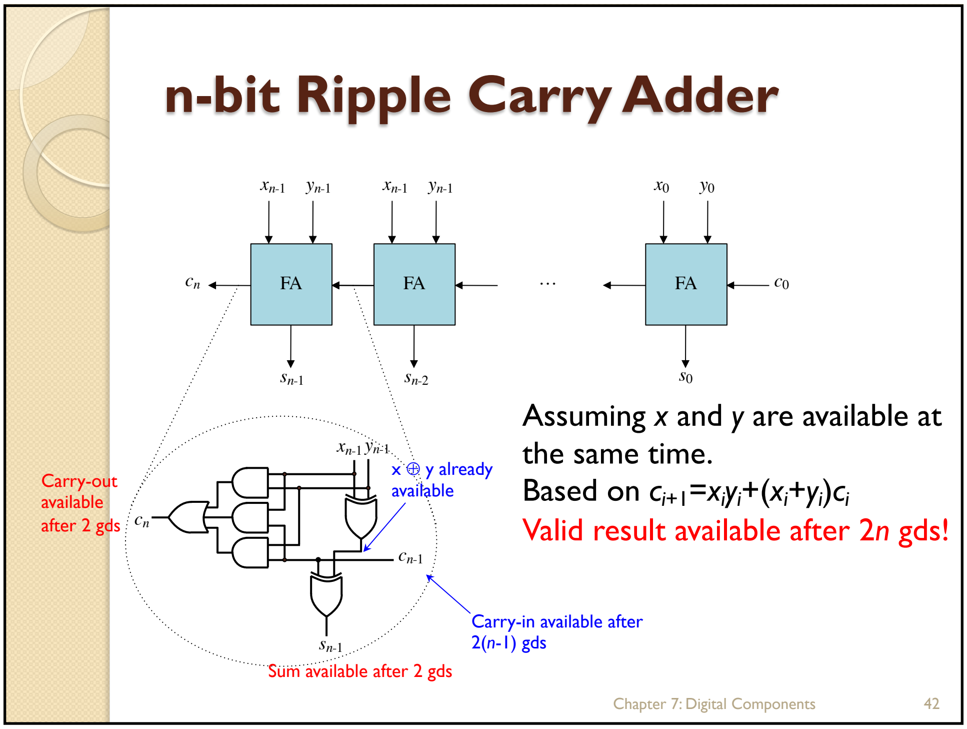
c_{i+1}

c_i \ $x_i y_i$	00	01	11	10
0			1	
1		1	1	1

$$c_{i+1} = x_i y_i + (x_i \oplus y_i)c_i$$

or

$$c_{i+1} = x_i y_i + (x_i + y_i)c_i$$



n-bit Ripple Carry Adder

Carry-out available after 2 gds

x ⊕ y already available

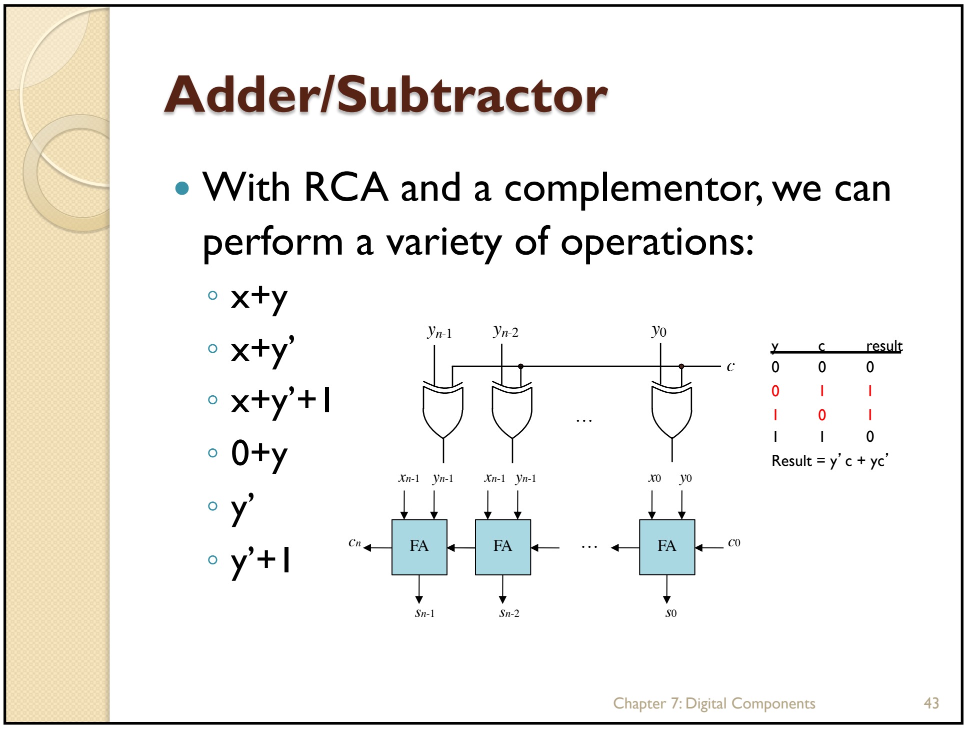
Carry-in available after 2(*n*-1) gds

Sum available after 2 gds

Assuming *x* and *y* are available at the same time.

Based on $c_{i+1}=x_iy_i+(x_i+y_i)c_i$

Valid result available after 2*n* gds!



Adder/Subtractor

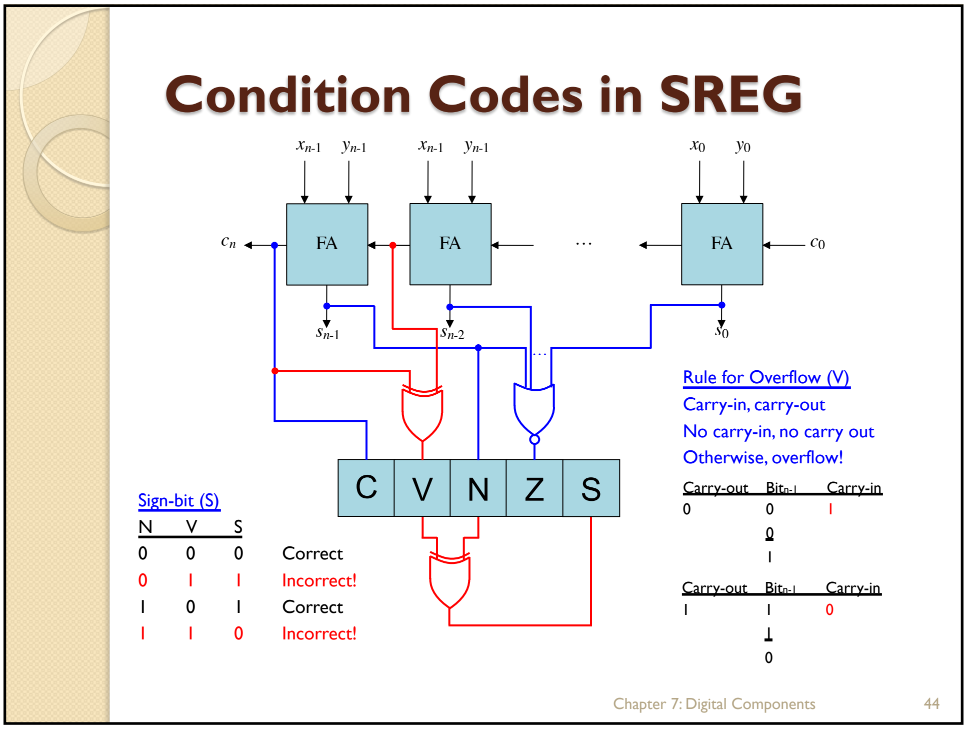
- With RCA and a complementor, we can perform a variety of operations:
  - x+y
  - x+y'
  - x+y'+1
  - 0+y
  - y'
  - y'+1

y	c	result
0	0	0
0	1	1
1	0	1
1	1	0

Result = y' c + yc'

Chapter 7: Digital Components 43

43



Condition Codes in SREG

Sign-bit (S)

N	V	S	
0	0	0	Correct
0	1	1	Incorrect!
1	0	1	Correct
1	1	0	Incorrect!

Rule for Overflow (V)

Carry-in, carry-out

No carry-in, no carry out

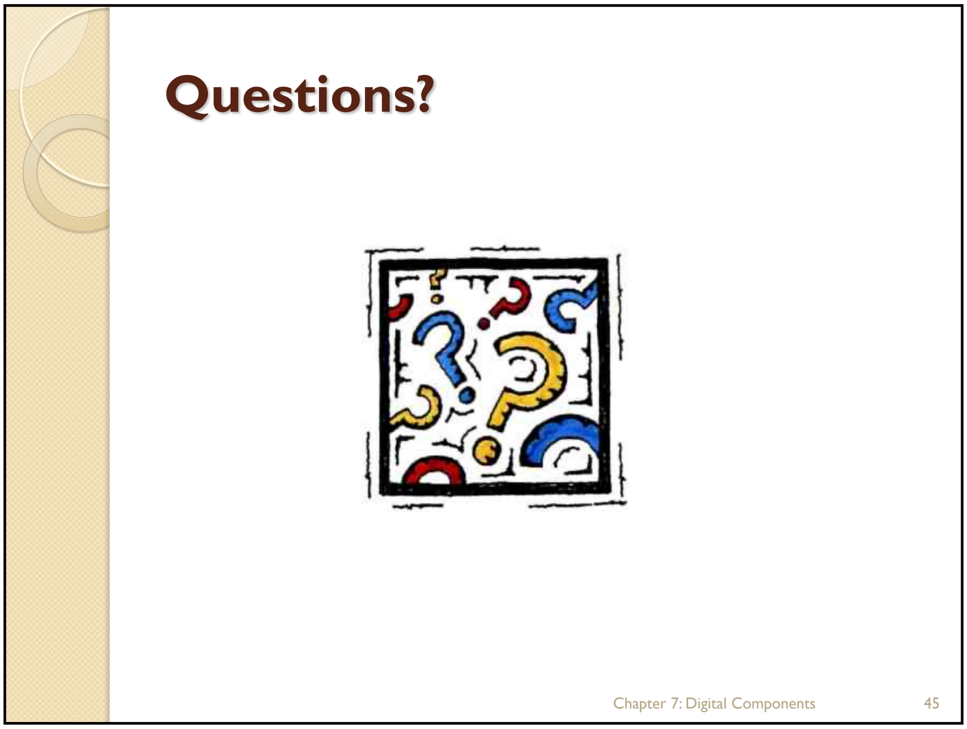
Otherwise, overflow!

Carry-out	Bit_{n-1}	Carry-in
0	0	1
	0	
	1	

Carry-out	Bit_{n-1}	Carry-in
1	1	0
	1	
	0	

Chapter 7: Digital Components 44

44



Questions?

45