CS2100 COMPUTER ORGANISATION

http://www.comp.nus.edu.sg/~cs2100/

Lecture #19

Sequential Logic

https://app.sli.do/event/aanrHmK6Geu3scZbeLh6wx

6.3 Flip-flop Excitation Tables (1/2)

- *Analysis*: Starting from a circuit diagram, derive the state table or state diagram.
- *Design*: Starting from a set of specifications (in the form of state equations, state table, or state diagram), derive the logic circuit.
- *Characteristic tables* are used in analysis.
- *Excitation tables* are used in design.

6.3 Flip-flop Excitation Tables (1/2)

- *Excitation tables*: given the required transition from present state to next state, determine the flip-flop input(s).

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

SR Flip-flop

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

D Flip-flop

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

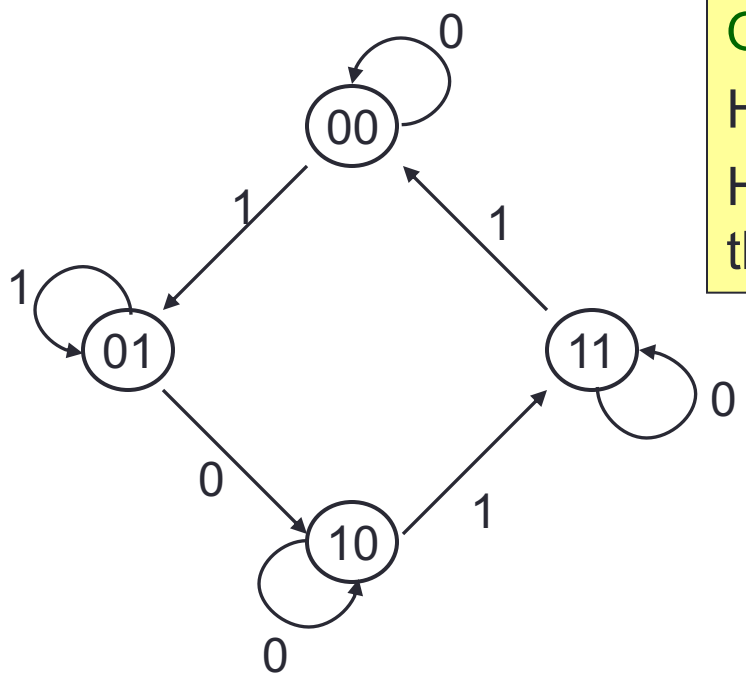
T Flip-flop

6.4 Sequential Circuits: Design

- Design procedure:
  - Start with circuit specifications – description of circuit behaviour, usually a state diagram or state table.
  - Derive the state table.
  - Perform state reduction if necessary.
  - Perform state assignment.
  - Determine number of flip-flops and label them.
  - Choose the type of flip-flop to be used.
  - Derive circuit excitation and output tables from the state table.
  - Derive circuit output functions and flip-flop input functions.
  - Draw the logic diagram.

6.4 Design: Example #1 (1/5)

- Given the following state diagram, design the sequential circuit using *JK* flip-flops.



Questions:

How many flip-flops are needed?

How many input variable are there?

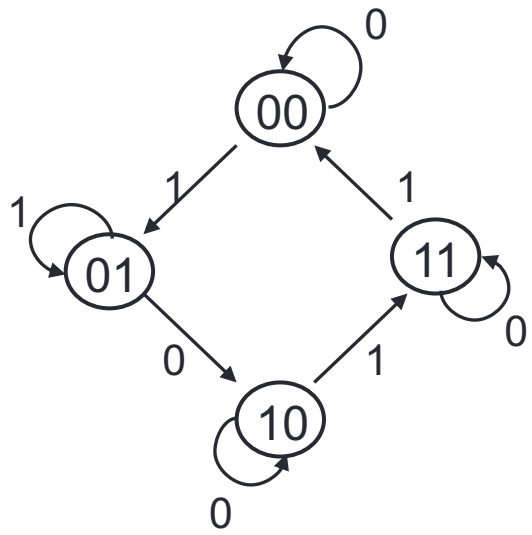
Answers:

Two flip-flops.
Let's call them *A* and *B*.

One input variable.
Let's call it *x*.

6.4 Design: Example #1 (2/5)

- Circuit state/excitation table, using *JK* flip-flops.



Present State	Next State	
	x=0	x=1
AB	A^+B^+	A^+B^+
00	00	01
01	10	01
10	10	11
11	11	00

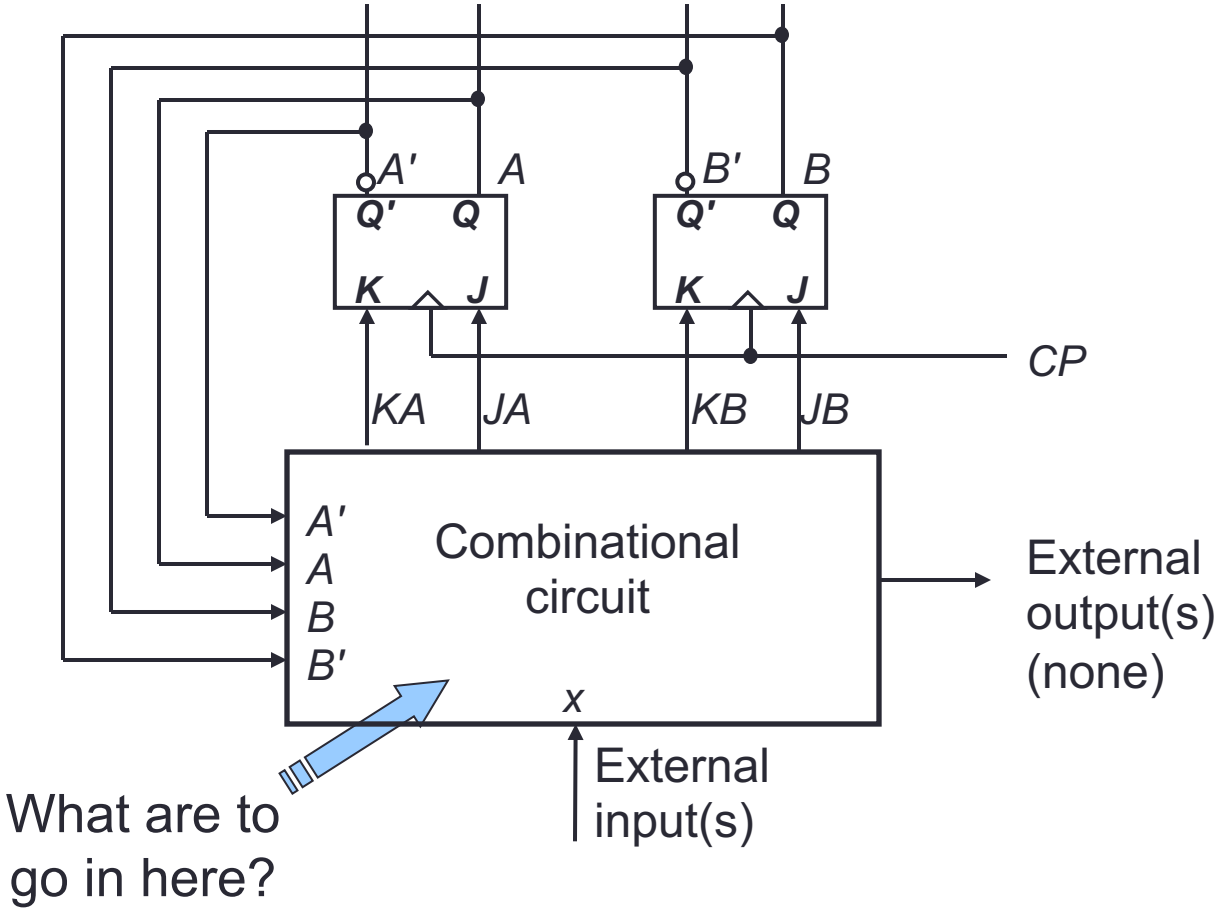
Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop's excitation table.

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

6.4 Design: Example #1 (3/5)

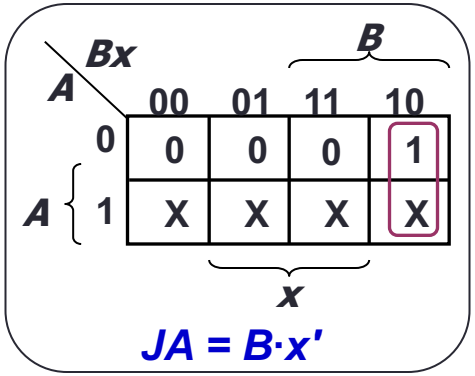
- Block diagram.

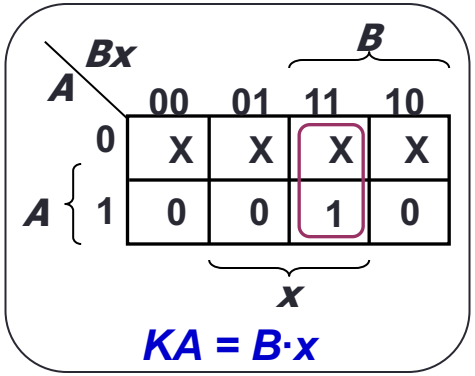


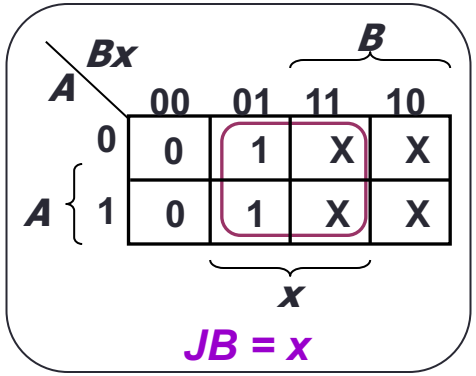
6.4 Design: Example #1 (4/5)

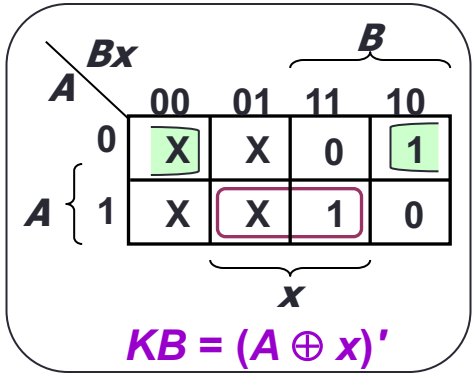
- From state table, get flip-flop input functions.

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1









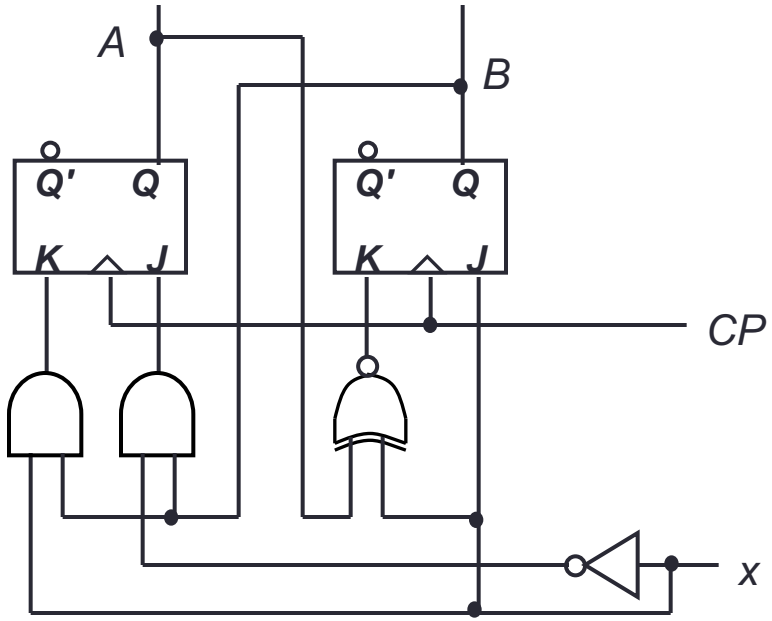
6.4 Design: Example #1 (5/5)

- Flip-flop input functions:

  $JA = B \cdot x'$ $JB = x$

  $KA = B \cdot x$ $KB = (A \oplus x)'$

- Logic diagram:



6.4 Design: Example #2 (1/3)

- Using *D* flip-flops, design the circuit based on the state table below. (Exercise: Design it using *JK* flip-flops.)

Present state		Input	Next state		Output
A	*B*	*x*	A^+	B^+	*y*
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

6.4 Design: Example #2 (2/3)

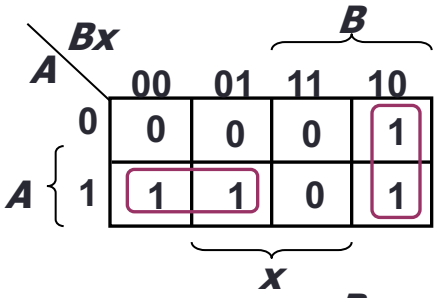
- Determine expressions for flip-flop inputs and the circuit output *y*.

Present state		Input	Next state		Output
A	*B*	*x*	A^+	B^+	*y*
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

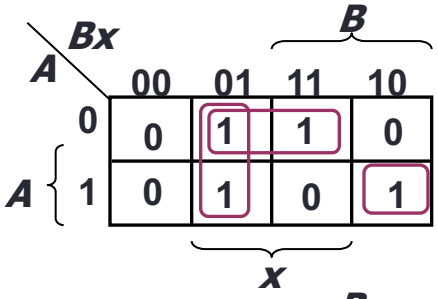
$DA(A,B,x) = \Sigma\, m(2,4,5,6)$

$DB(A,B,x) = \Sigma\, m(1,3,5,6)$

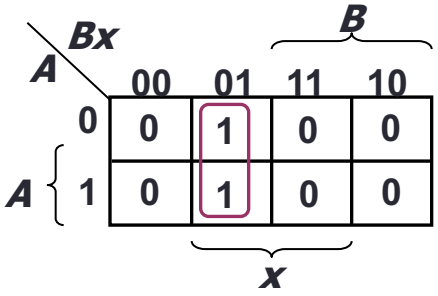
$y(A,B,x) = \Sigma\, m(1,5)$



$DA = A \cdot B' + B \cdot x'$



$DB = A' \cdot x + B' \cdot x + A \cdot B \cdot x'$



$y = B' \cdot x$

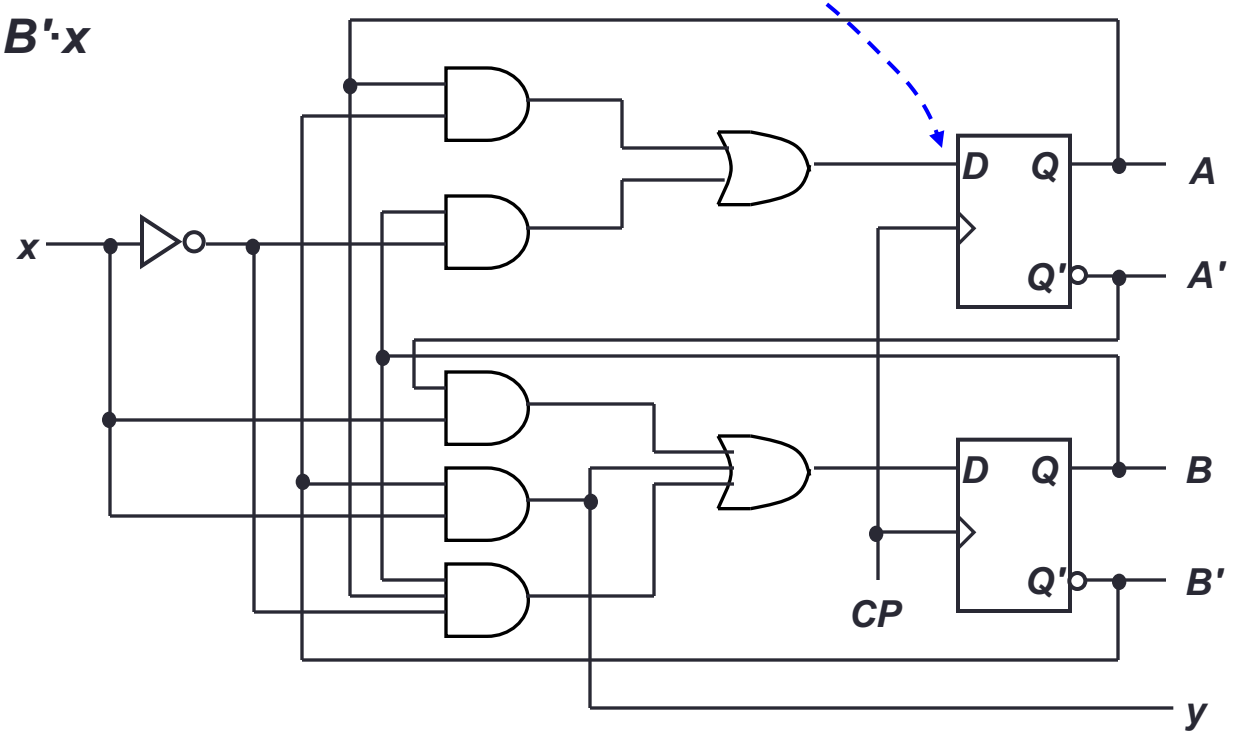
6.4 Design: Example #2 (3/3)

- From derived expressions, draw logic diagram:

$DA = A \cdot B' + B \cdot x'$

$DB = A' \cdot x + B' \cdot x + A.B \cdot x'$

$y = B' \cdot x$



6.4 Design: Example #3 (1/4)

- Design involving unused states.

Present state			Input	Next state			Flip-flop inputs						Output
A	B	C	x	A^+	B^+	C^+	SA	RA	SB	RB	SC	RC	y
0	0	1	0	0	0	1	0	X	0	X	X	0	0
0	0	1	1	0	1	0	0	X	1	0	0	1	0
0	1	0	0	0	1	1	0	X	X	0	1	0	0
0	1	0	1	1	0	0	1	0	0	1	0	X	0
0	1	1	0	0	0	1	0	X	0	1	X	0	0
0	1	1	1	1	0	0	1	0	0	1	0	1	0
1	0	0	0	1	0	1	X	0	0	X	1	0	0
1	0	0	1	1	0	0	X	0	0	X	0	X	1
1	0	1	0	0	0	1	0	1	0	X	X	0	0
1	0	1	1	1	0	0	X	0	0	X	0	1	1

Given these

Derive these

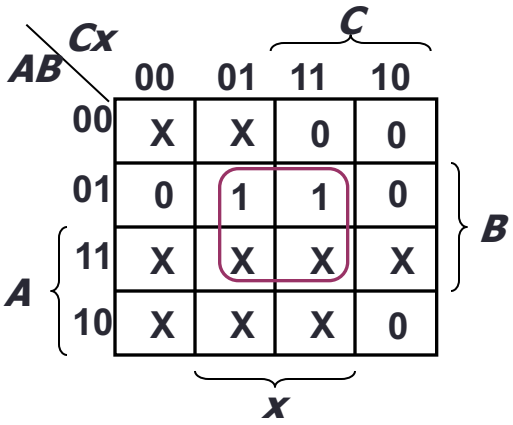
Are there other unused states?

Unused state 000:

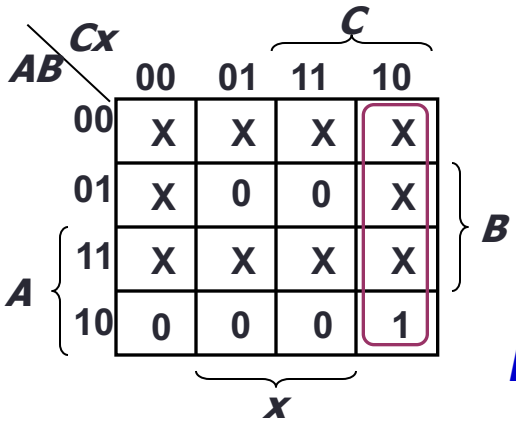
0	0	0	0	X	X	X	X	X	X	X	X	X	X
0	0	0	1	X	X	X	X	X	X	X	X	X	X

6.4 Design: Example #3 (2/4)

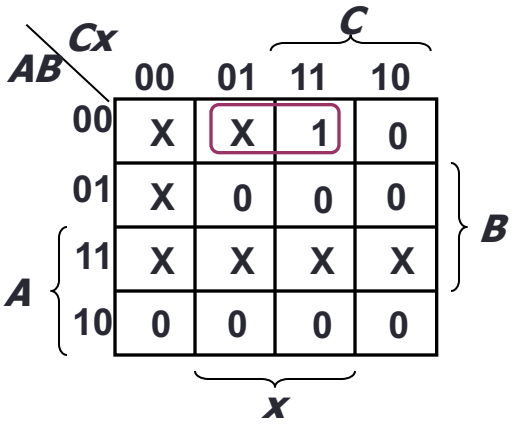
- From state table, obtain expressions for flip-flop inputs.



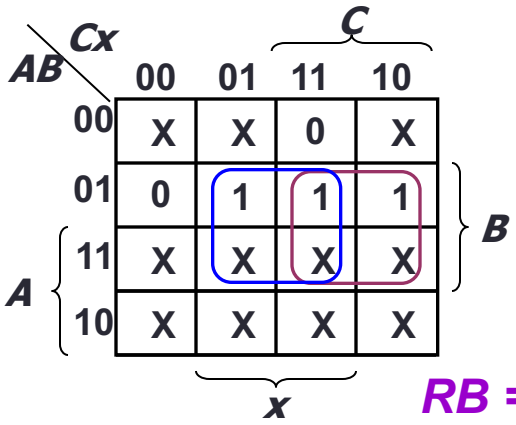
$SA = B \cdot x$



$RA = C \cdot x'$



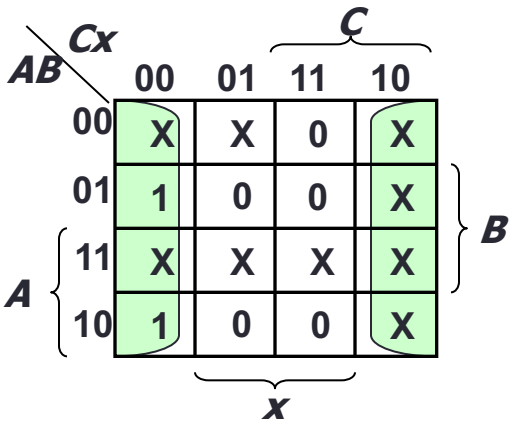
$SB = A' \cdot B' \cdot x$



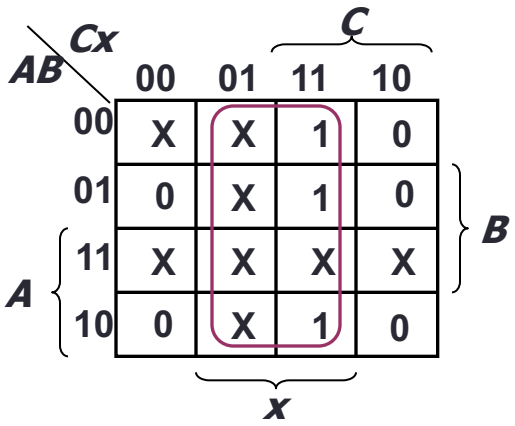
$RB = B \cdot C + B \cdot x$

6.4 Design: Example #3 (3/4)

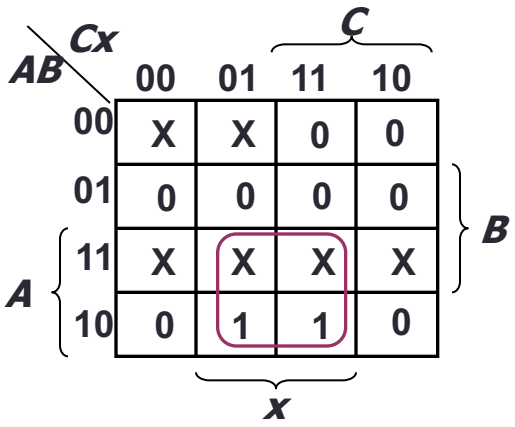
- From state table, obtain expressions for flip-flop inputs (cont'd).



$SC = x'$



$RC = x$

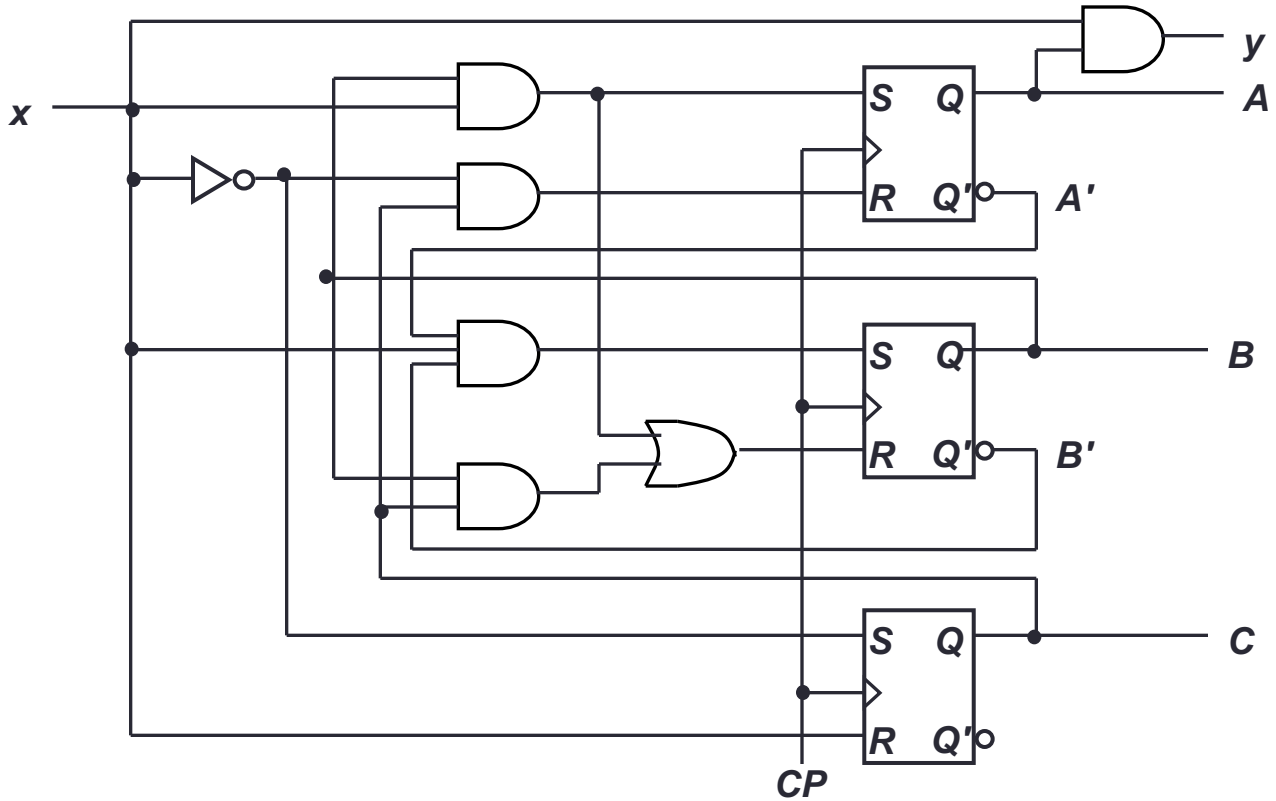


$y = A \cdot x$

6.4 Design: Example #3 (4/4)

- From derived expressions, draw the logic diagram:

$SA = B \cdot x$	$SB = A' \cdot B' \cdot x$	$SC = x'$	$y = A \cdot x$
$RA = C \cdot x'$	$RB = B \cdot C + B \cdot x$	$RC = x$	

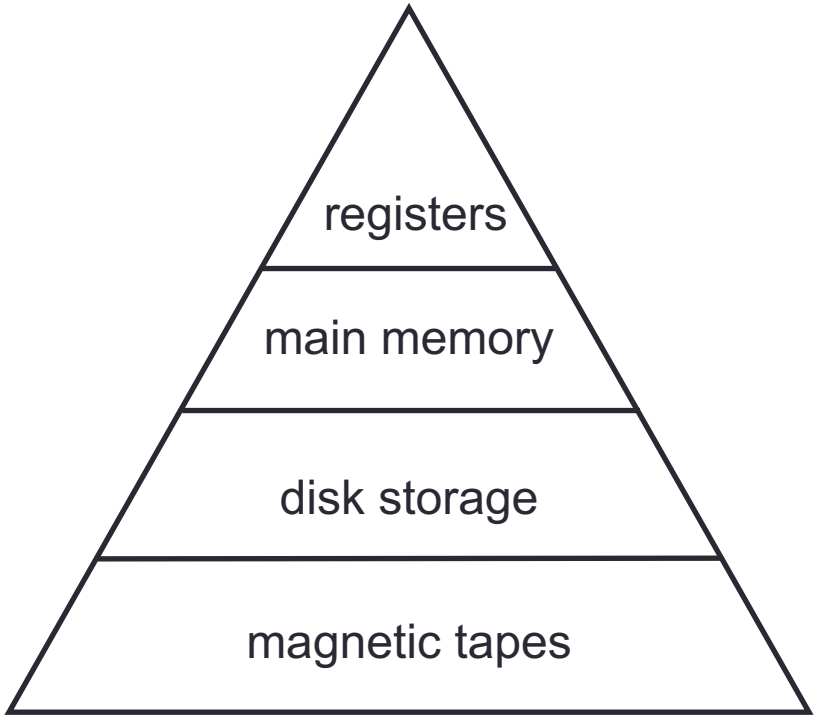


7. Memory (1/4)

- Memory stores programs and data.
- Definitions:
  - 1 byte = 8 bits
  - 1 word: in multiple of bytes, a unit of transfer between main memory and registers, usually size of register.
  - 1 KB (kilo-bytes) = 2^{10} bytes; 1 MB (mega-bytes) = 2^{20} bytes; 1 GB (giga-bytes) = 2^{30} bytes; 1 TB (tera-bytes) = 2^{40} bytes.
- Desirable properties: fast access, large capacity, economical cost, non-volatile.
- However, most memory devices do not possess all these properties.

7. Memory (2/4)

Memory hierarchy

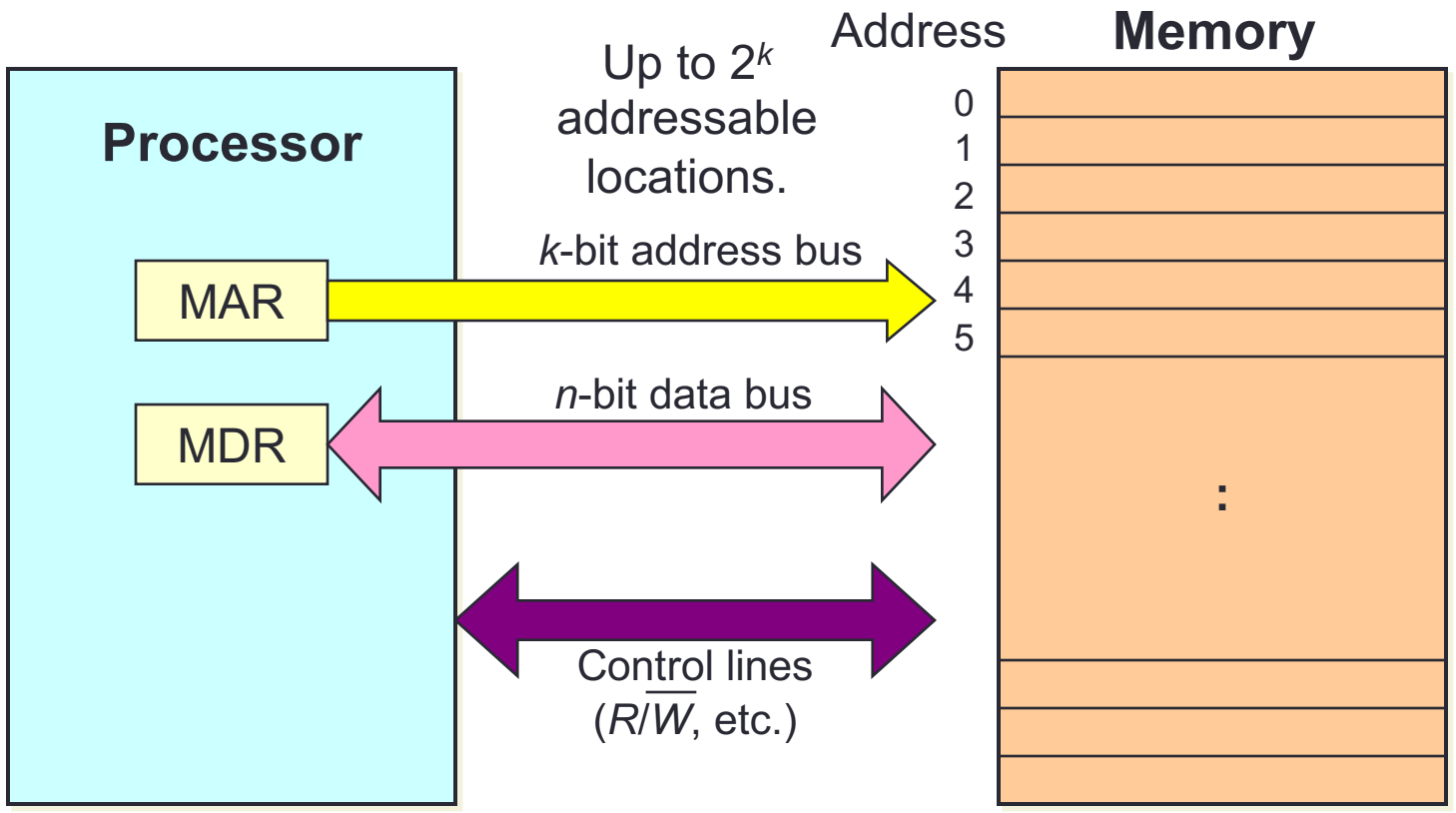


Fast, expensive (small numbers), volatile

Slow, cheap (large numbers), non-volatile

7. Memory (3/4)

Data transfer

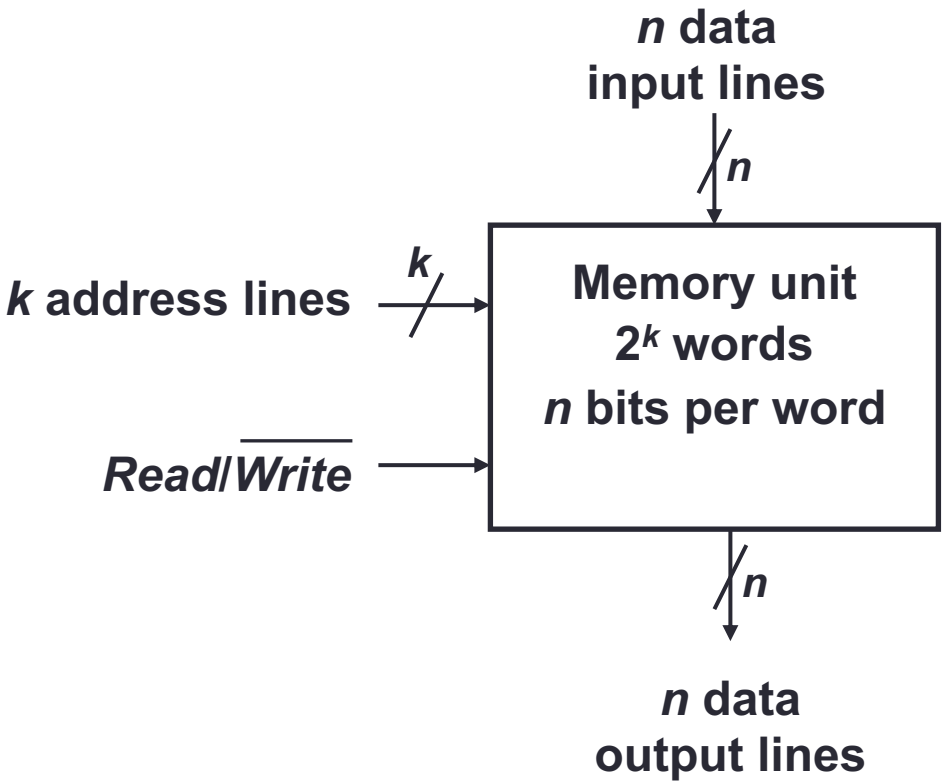


7. Memory (4/4)

- A memory unit stores binary information in groups of bits called *words*.
- The data consists of *n* lines (for *n*-bit words). Data input lines provide the information to be stored (*written*) into the memory, while data output lines carry the information out (*read*) from the memory.
- The address consists of *k* lines which specify which word (among the 2^k words available) to be selected for reading or writing.
- The control lines *Read* and *Write* (usually combined into a single control line *Read/Write*) specifies the direction of transfer of the data.

7.1 Memory Unit

- Block diagram of a memory unit:



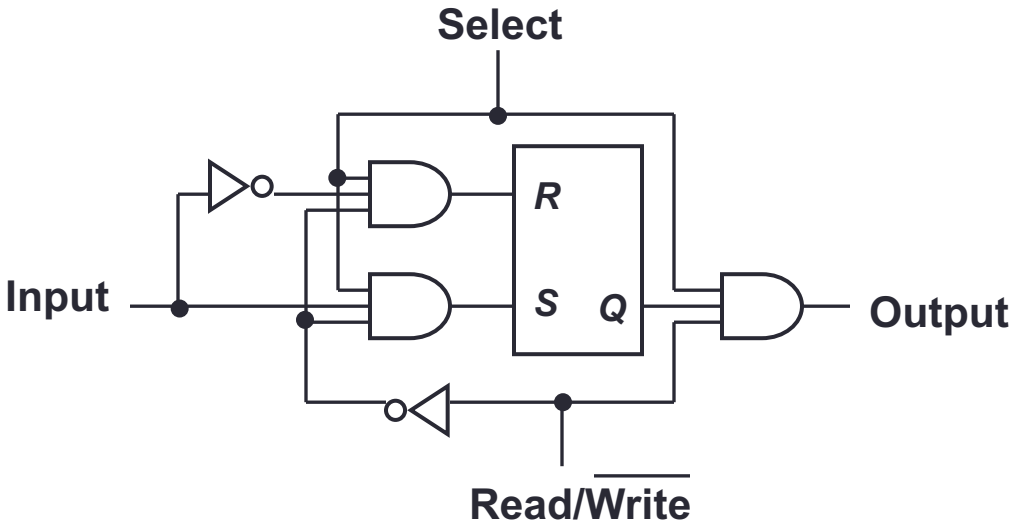
7.2 Read/Write Operations

- Write operation:
  - Transfers the address of the desired word to the address lines.
  - Transfers the data bits (the word) to be stored in memory to the data input lines.
  - Activates the *Write* control line (set *Read/Write* to 0).
- Read operation:
  - Transfers the address of the desired word to the address lines.
  - Activates the *Read* control line (set *Read/Write* to 1).

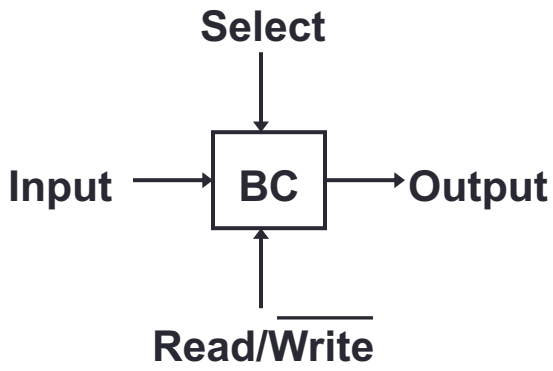
Memory Enable	*Read/Write*	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

7.3 Memory Cell

- Two types of RAM
  - Static RAMs use flip-flops as the memory cells.
  - Dynamic RAMs use capacitor charges to represent data. Though simpler in circuitry, they have to be constantly refreshed.
- A single memory cell of the static RAM has the following logic and block diagrams:



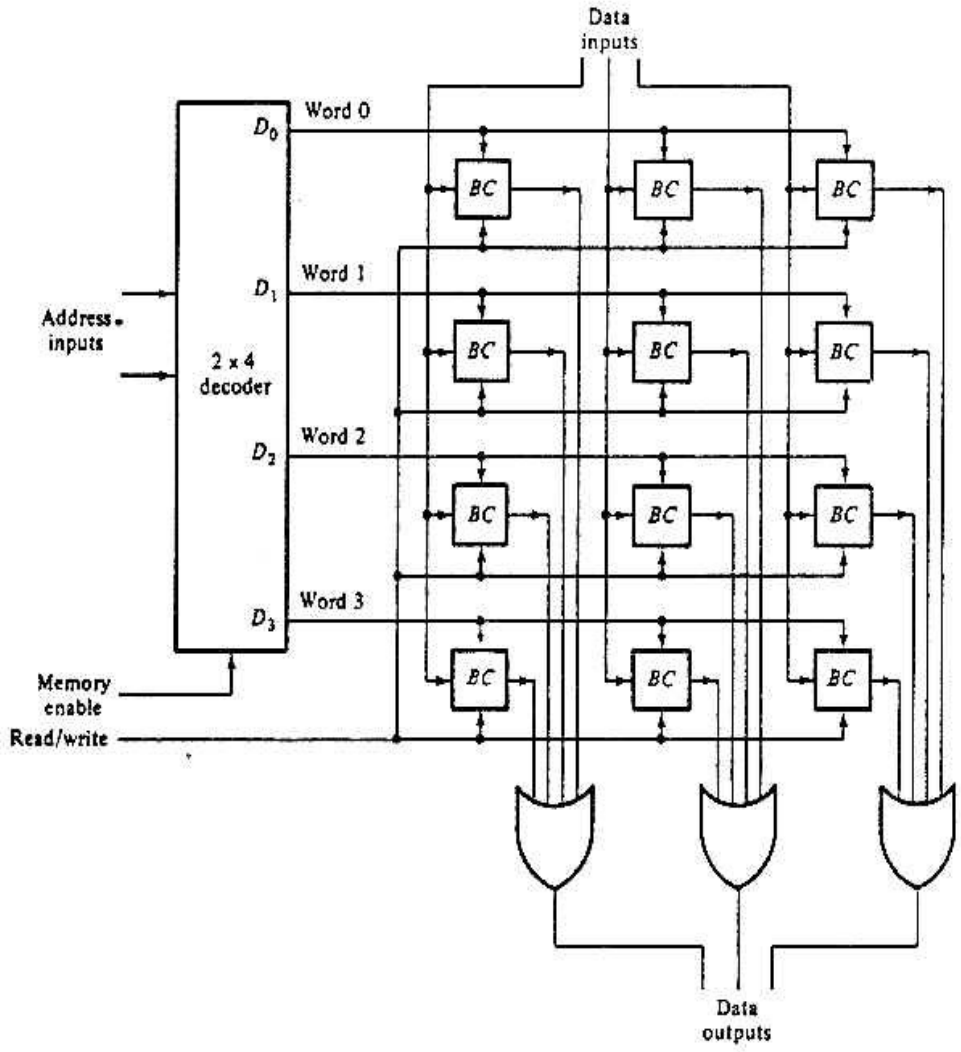
Logic diagram



Block diagram

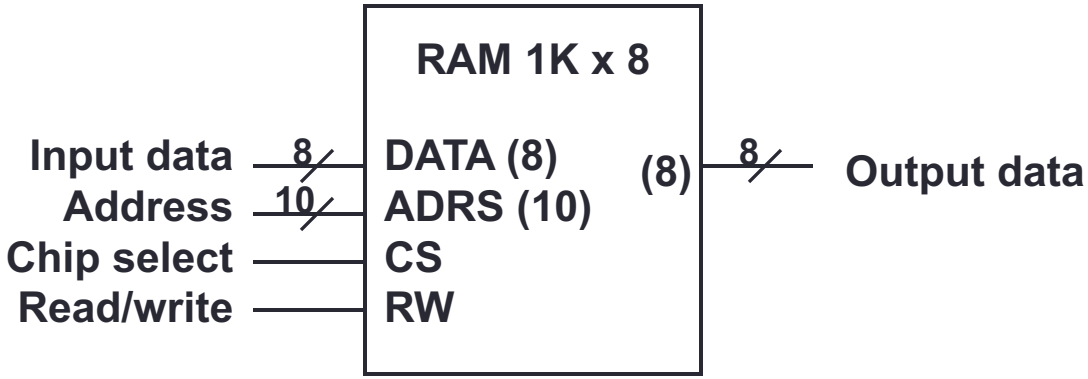
7.4 Memory Arrays (1/4)

- Logic construction of a 4×3 RAM (with decoder and OR gates):



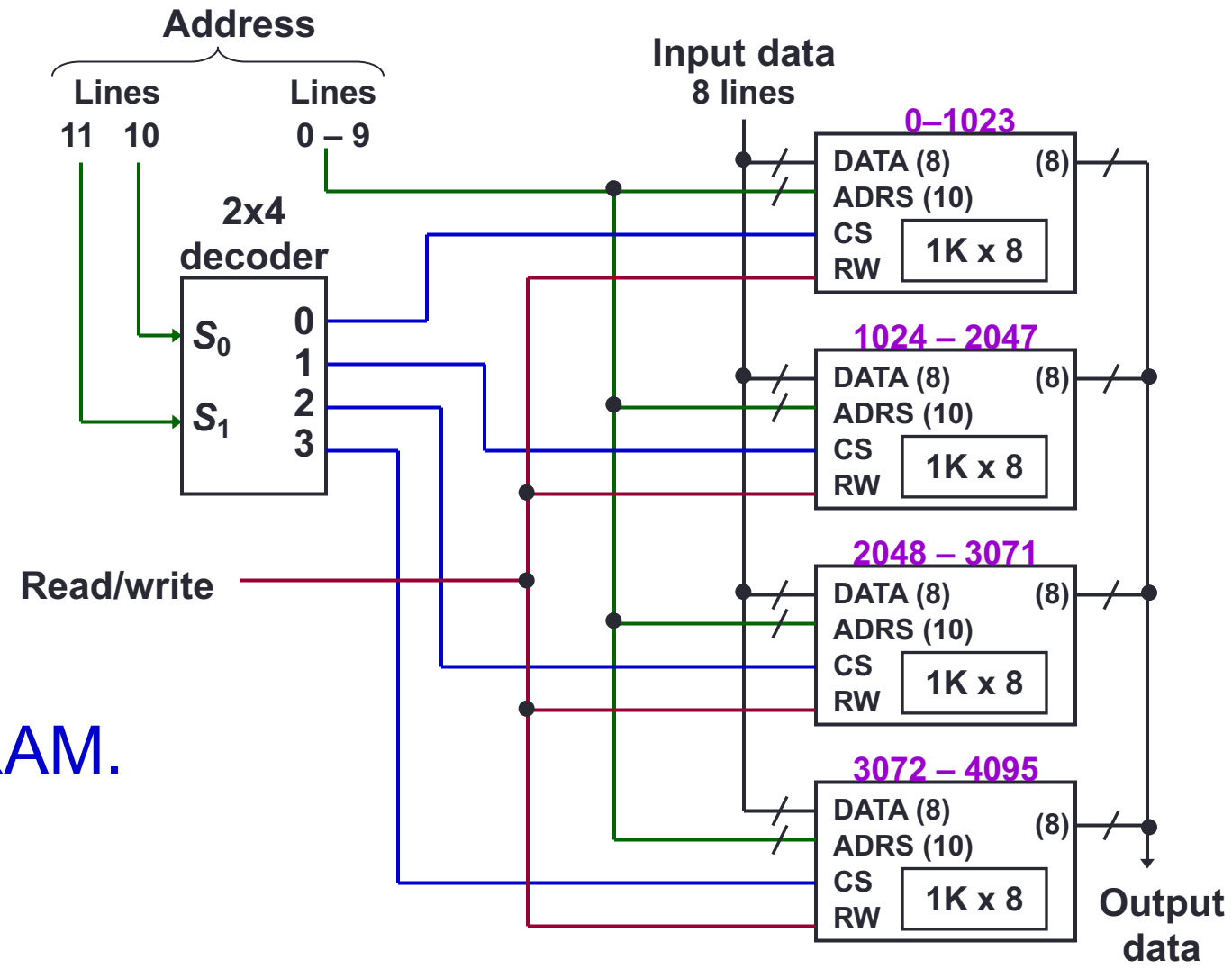
7.4 Memory Arrays (2/4)

- An array of RAM chips: memory chips are combined to form larger memory.
- A 1K × 8-bit RAM chip:



Block diagram of a 1K x 8 RAM chip

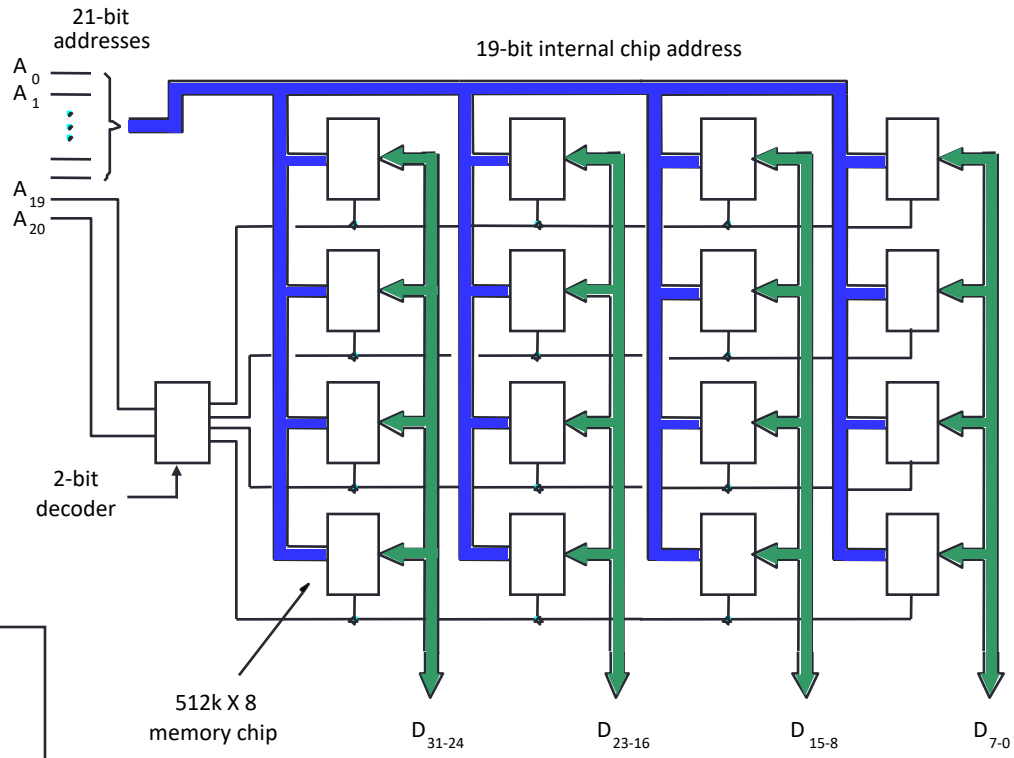
7.4 Memory Arrays (3/4)

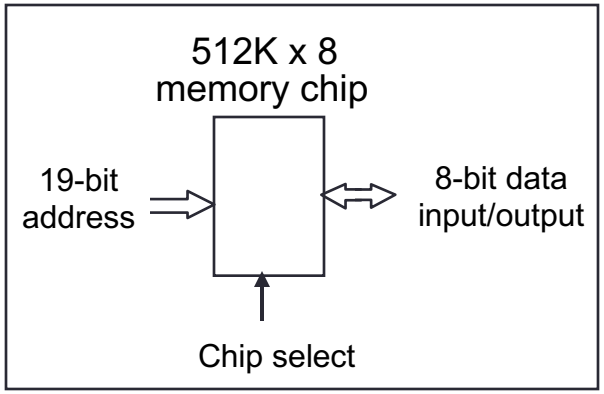


- 4K × 8 RAM.

7.4 Memory Arrays (4/4)

Read/write control line not included in this diagram.





- 2M × 32 memory module
  - Using 512K × 8 memory chips.

End of File