EE103 Digital Systems 1

Wong Chin Hong

106

Maynooth International Engineering College
Fuzhou University

So far … !

- We've used Karnaugh Maps to minimise logic ...
- We've implemented circuits using NAND only or NOR only gates ...
- We've looked in detail at the SR, D, JK and T flipflops ...
- We've analysed and designed a counter ...

TODAY, we are going to look at Registers …

BUT FIRST, let's finish off the counter design from the last lecture …

Designing a Counter

- Finally, we need to use NAND implementation. Note that, by default, complements are available from the flipflops.

- Converting to NAND gives:

$A_J = B$

$A_K = BCD = \overline{\overline{BCD}}$ ②

$B_J = CD = \overline{\overline{CD}}$ ①

$B_K = \overline{A} + CD = \overline{A.\overline{CD}}$ $\boxed{\gamma}$

$C_J = D + \overline{A}B = \overline{\overline{D}.\overline{\overline{A}B}}$ $\boxed{\gamma}$

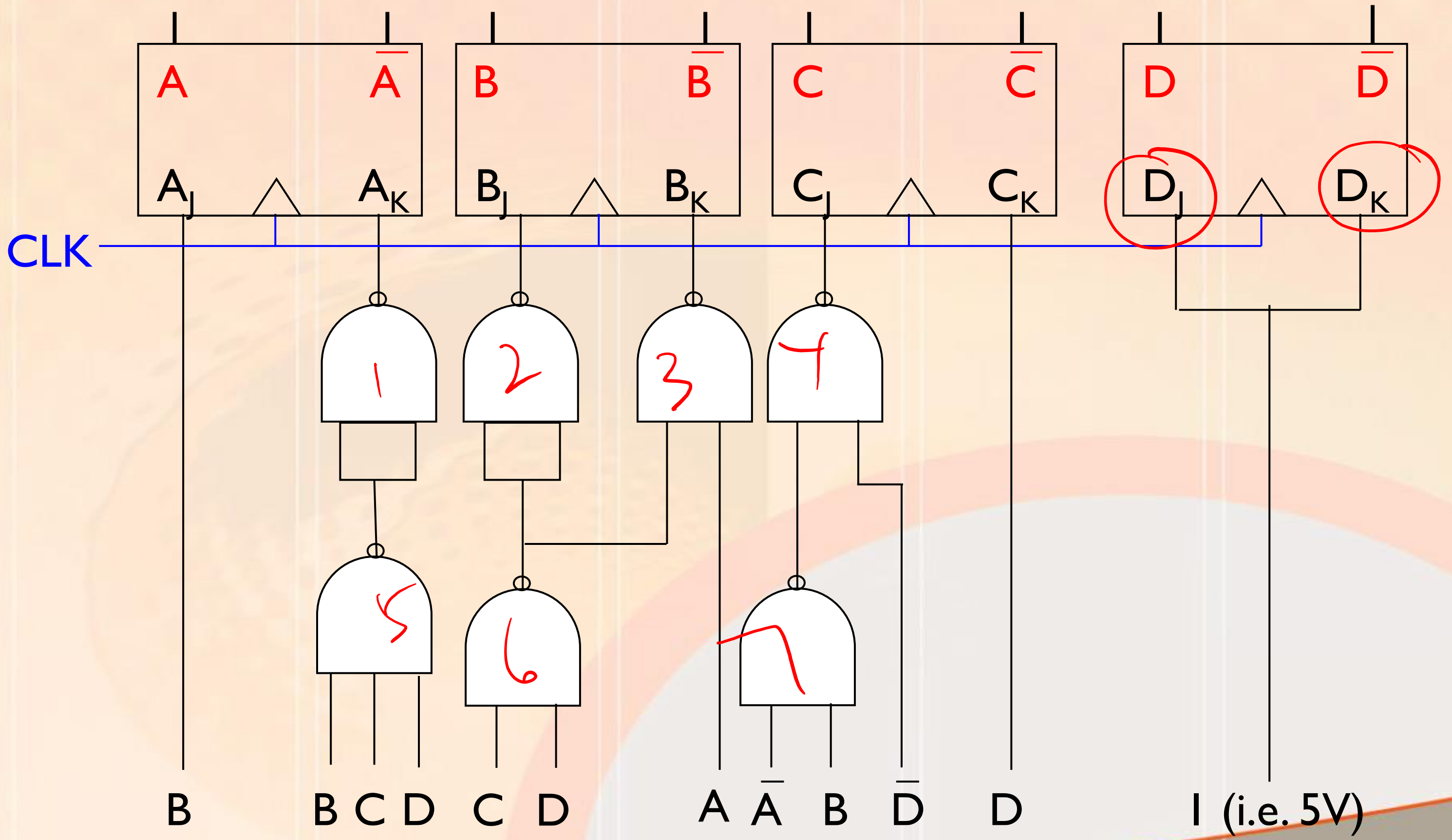
$C_K = D$

$D_J = 1$

$D_K = 1$

Designing a Counter

- This requires a total of 7 NAND gates, leading to the following circuit design:



Designing a Counter

- In the above diagram, all the variables (A, B, C, D) and their complements are generated by the outputs of the flipflops. For ease of viewing, these connections are omitted from the diagram.

- As is good practice in engineering design, we need to **carry out a self check of our design**.

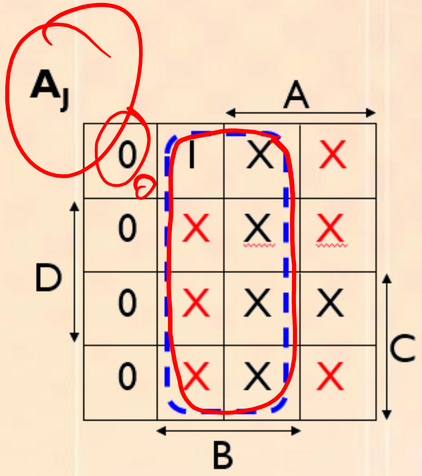
- More importantly, we **need to determine what the unused states do in our design**, as these were not allowed for at the outset.

- Recall the concise form of the JK state table:

J	K	$Q(\tau)$
0	0	Q
0	1	**0**
1	0	**1**
1	1	$\overline{Q}$

Designing a Counter

- Carrying out a full self check of our circuit design we get the following table:

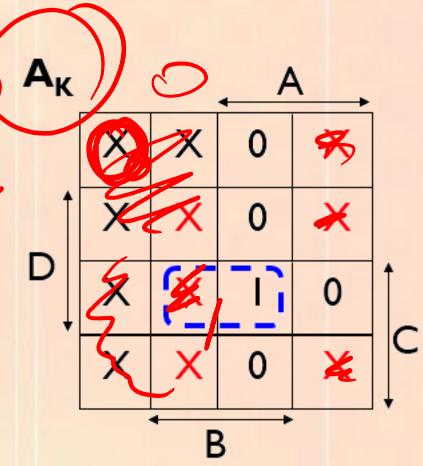


0	1	X	X
0	X	X	X
0	X	X	X
0	X	X	X

A_J = B

0 0	1 4	1 12	0 8
0 1	1 5	1 13	0 9
0 3	1 7	1 15	0 11
0 2	1 6	1 14	0 10

Minterm	Count	ABCD	Flipflop Inputs A_J A_K	B_J B_K	C_J C_K	D_J D_K
0	0	0000	0 0	0 1	0 0	1 1
	1	0001	0 0			



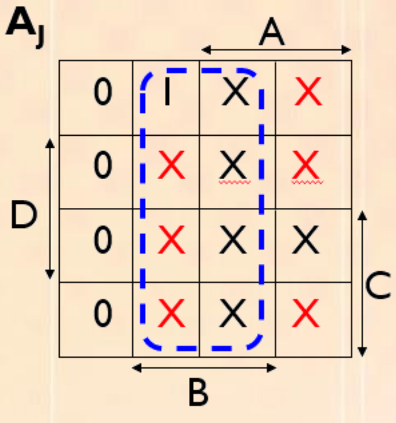
X	X	0	X
X	X	0	X
X	X	1	0
X	X	0	X

A_K = BCD

Designing a Counter

- Carrying out a full self check of our circuit design we get the following table:

A_J

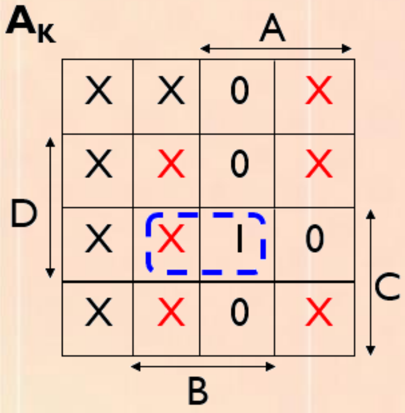


0	1	X	X
0	X	X	X
0	X	X	X
0	X	X	X

$A_J = B$

			Flipflop Inputs			
Minterm	Count	ABCD	A_J A_K	B_J B_K	C_J C_K	D_J D_K
0	0	0000	0 0	0 1	0 0	1 1
1	1	0001	0 0	0 1	1 1	1 1
2	2	0010	0 0	0 1	0 0	1 1
3	3	0011	0 0	1 1	1 1	1 1
4	4	0100	1 0	0 1	1 0	1 1
11	5	1011	0 0	1 1	1 1	1 1
12	6	1100	1 0	0 0	0 0	1 1
13	7	1101	1 0	0 0	1 1	1 1
14	8	1110	1 0	0 0	0 0	1 1
15	9	1111	1 1	1 1	1 1	1 1
0	0	0000				

A_K



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

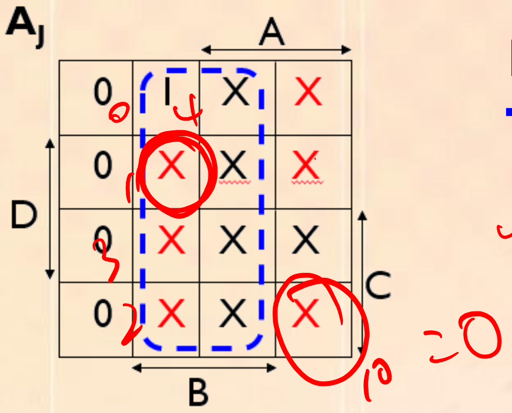
$A_K = BCD$

Designing a Counter

J	K	Q(t)
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

- Checking the unused states …

A_{J} 8-5 = 1



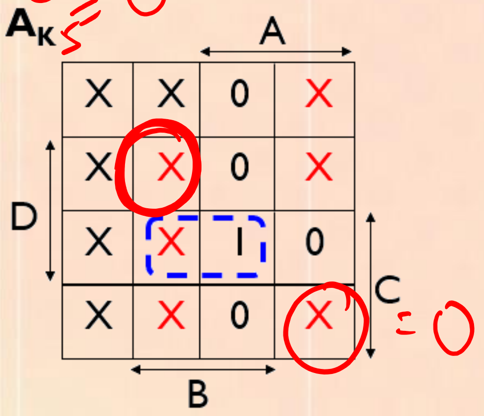
0	1	X	X
0	X	X	X
0	X	X	X
0	X	X	X

A_J = B

Minterm	Count	ABCD	A_J A_K	B_J B_K	C_J C_K	D_J D_K
5	-	0101	1 0	0 1	1 1	1 1
10		1010	0 0	✓	✓	✓
		1				

Flipflop Inputs

A_{K5} = 0

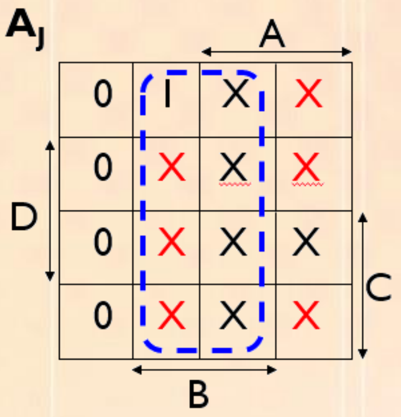


X	X	0	X
X	X	0	X
X	X	1	0
X	X	0	X

A_K = BCD

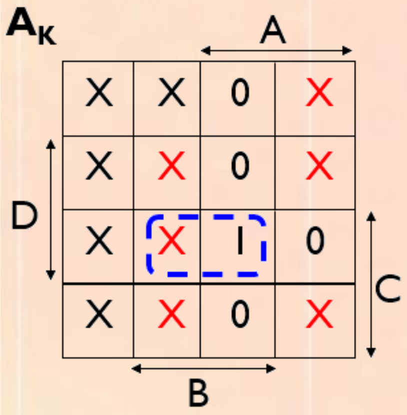
Designing a Counter

- Checking the unused states …



A_J = B

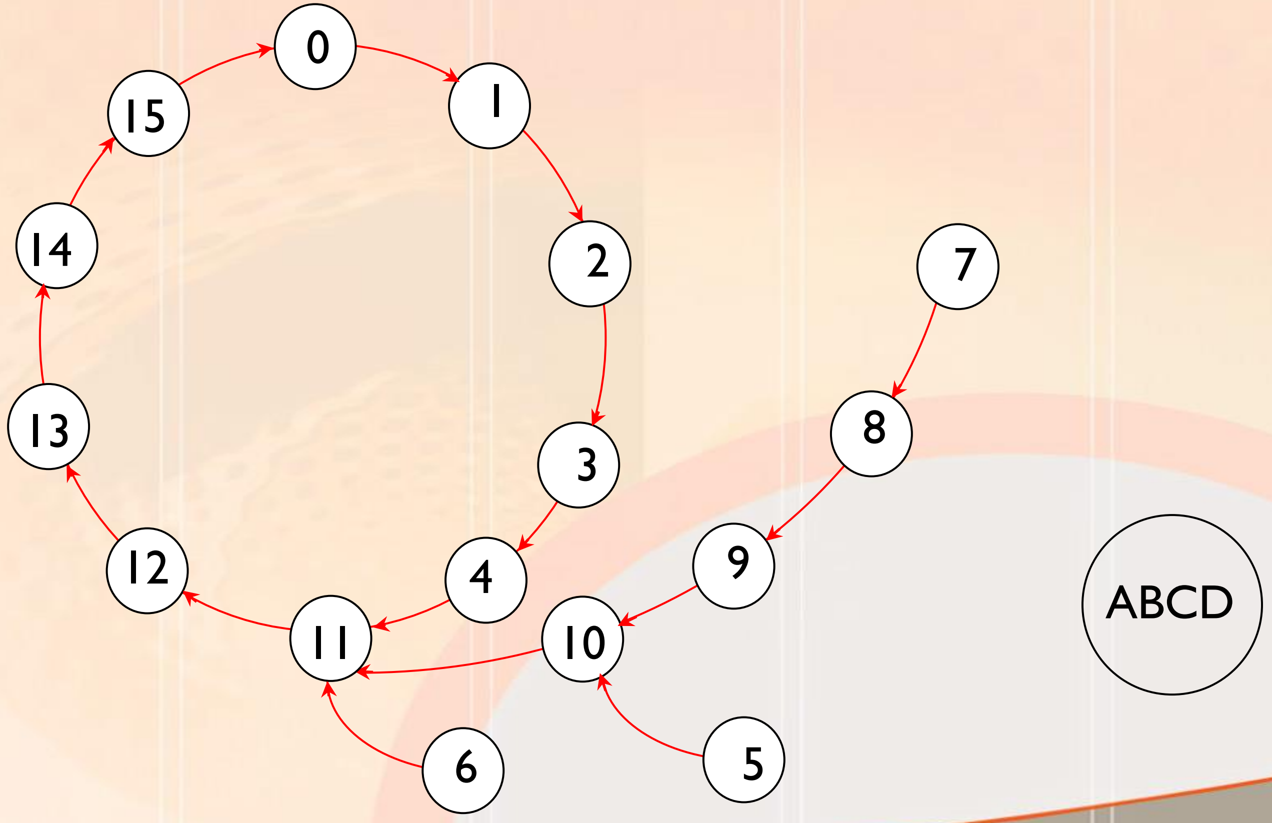
Minterm	Count	ABCD	Flipflop Inputs			
			A_J A_K	B_J B_K	C_J C_K	D_J D_K
5	-	0101	1 0	0 1	1 1	1 1
10	-	1010	0 0	0 0	0 0	1 1
11	5	1011				
6	-	0110	1 0	0 1	1 0	1 1
11	5	1011				
7	-	0111	1 1	1 1	1 1	1 1
8	-	1000	0 0	0 0	0 0	1 1
9	-	1001	0 0	0 0	1 1	1 1
10	-	1010				



A_K = BCD

Designing a Counter

- Finally, although not required for the question, the behaviour of the designed counter can be summarized by the following state diagram (expressed using minterms):



Registers

- A **register** is a **set of binary storage cells**, each storing one bit of information.

- It consists of an array of flipflops connected together with a common clock.

- While the actual content of a register is binary information, the interpretation of this information will vary considerably from one application to another.

Registers

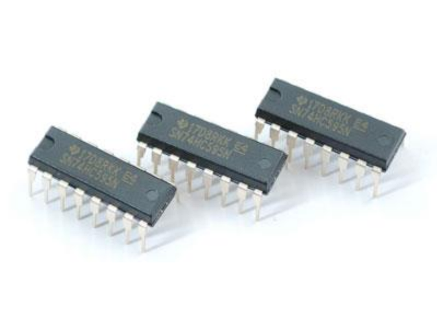
- An *n*-bit register consists of *n* flipflops and is capable of storing binary information containing *n* bits.

- In addition to flipflops, registers may also contain some combinational logic which can perform certain data processing tasks.

- Typically, the flipflops are used to store the information and the combinational circuitry controls when and how the information is transferred into the register.

Shift Registers

- A shift register is one that moves (or shifts) its information sideways (either left to right or right to left).
- The flipflops are cascaded together (i.e. in series) with the output of one connected to the input of the next.
- The flipflops are controlled by a common clock. On each clock pulse the information is shifted by one bit to either the left or to the right.



Shift Registers

- Note – a given binary number **multiplied by 2 corresponds to a shift to the LEFT** by one position and the new least significant bit (LSB) is set to 0.

- For example, consider the binary number: **0100**

- **Shift to the left** by one and insert 0 for the LSB gives:

1000

- This is the same as multiplying by 2, i.e.:

0100 x 2 = 1000

Shift Registers

- Similarly – a given binary number **divided by 2 corresponds to a shift to the RIGHT** by one position and the new most significant bit (MSB) is set to 0.

- For example, consider the binary number: **1000**

- **Shift to the right** by one and insert 0 for the MSB gives:

0100

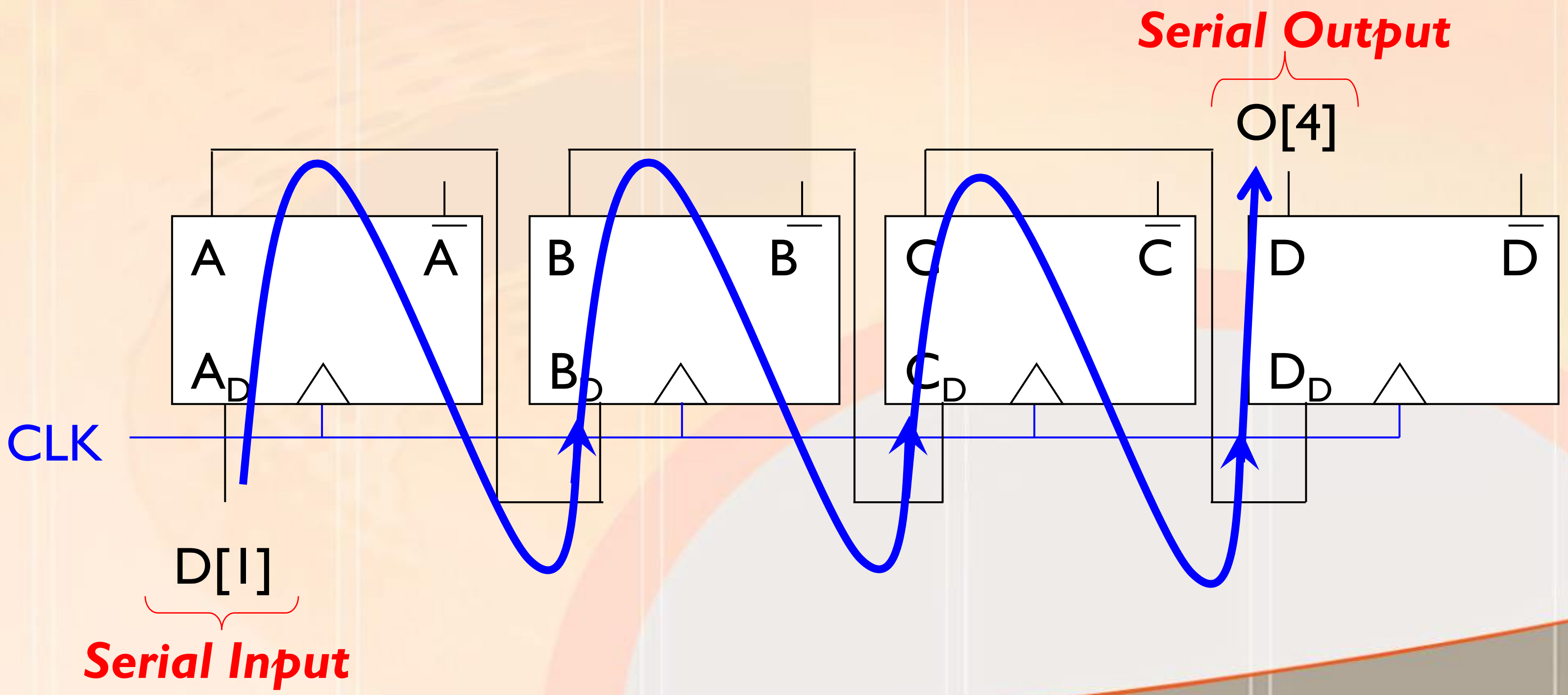
- This is the same as dividing by 2, i.e.:

1000 ÷ 2 = 0100

Serial & Parallel Data Entry

- There are two ways by which data can be entered into registers, namely serial and parallel entry.

- **Serial data entry** is where the data is entered one bit at a time:

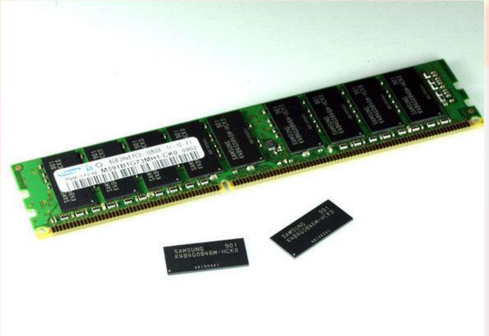


Serial & Parallel Data Entry

- On each clock pulse, one bit of information is entered into the flipflop A. On subsequent clock pulses, this bit gets shifted from one flipflop to the next

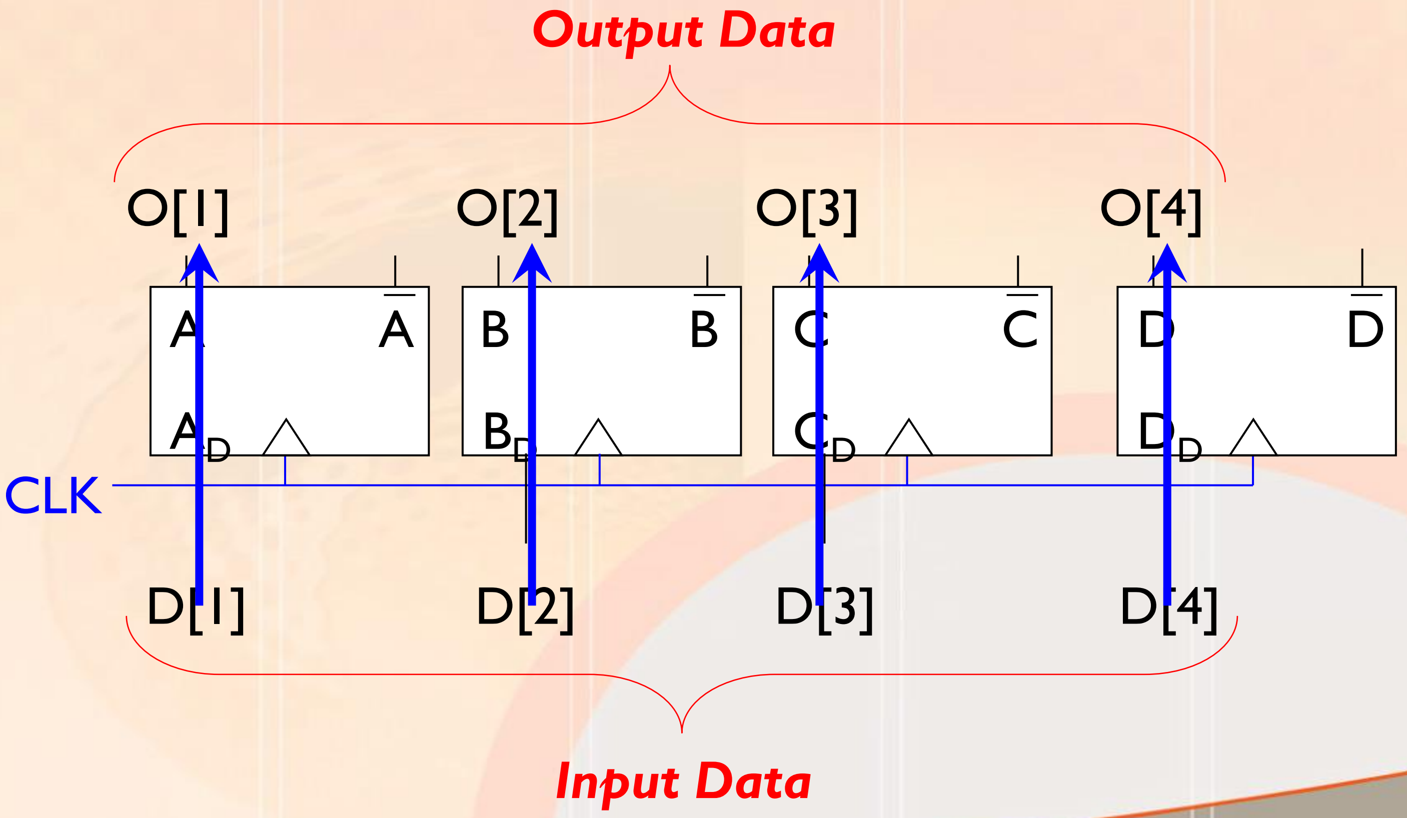
- The circuit shown is an example of a ***4-bit shift register***.

- **Parallel data entry** is where multiple data bits are entered to the register at the same time.



Serial & Parallel Data Entry

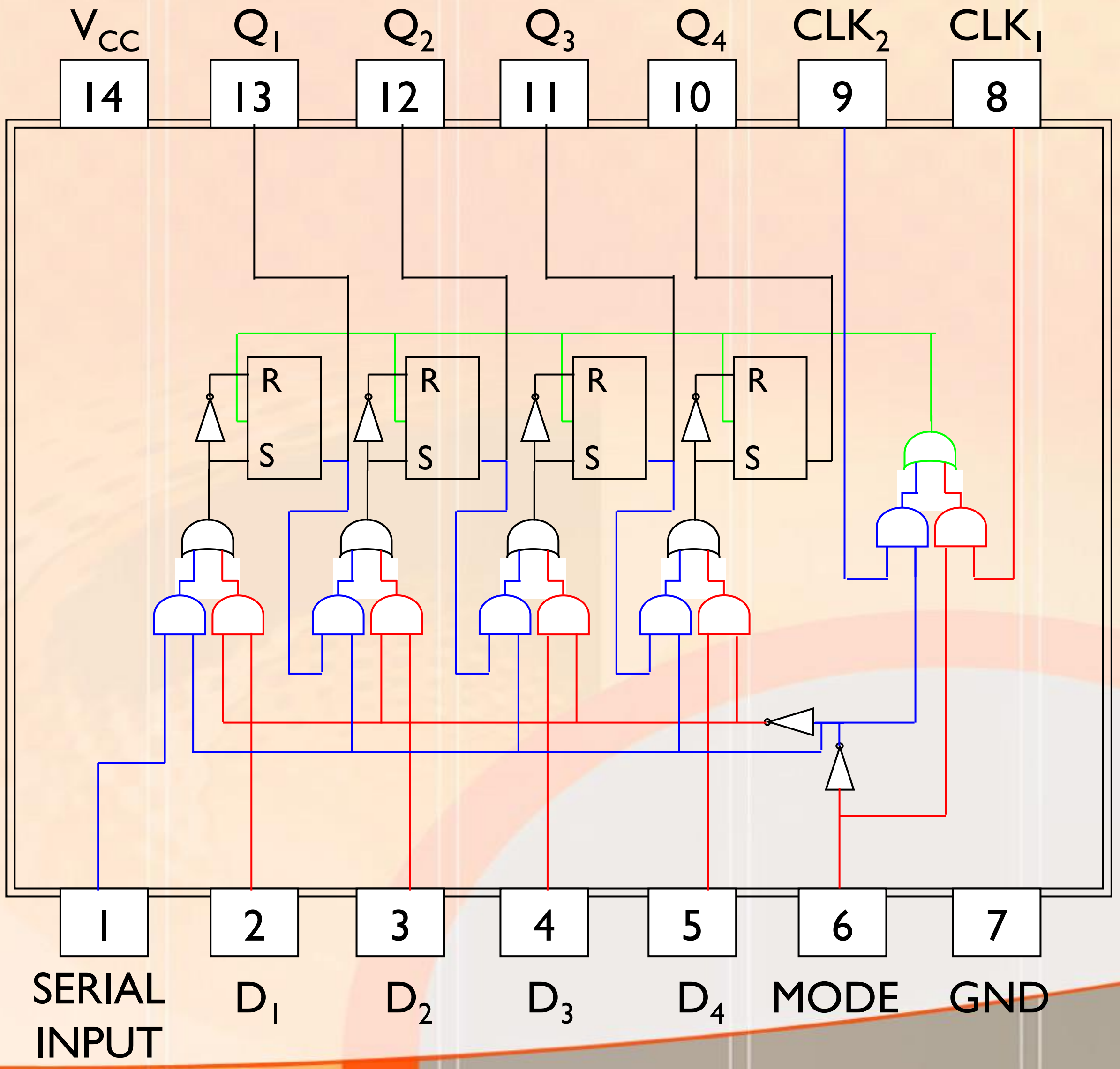
- The circuit below illustrates how 4-bits of information are transferred in parallel on a single clock pulse:



Universal Shift Register

- Both the serial and parallel data entry modes can be incorporated into a single IC known as the ***Universal Shift Register, as shown in the following diagram:***

Universal Shift Register



Universal Shift Register

- This device allows for easy conversion from serial to parallel.
- For example, a 4-bit serial word could be clocked into the shift registers in 4 cycles.
- When the full word is entered the mode can be changed to parallel and the parallel word can then be clocked out in one cycle.