Combinational Circuits

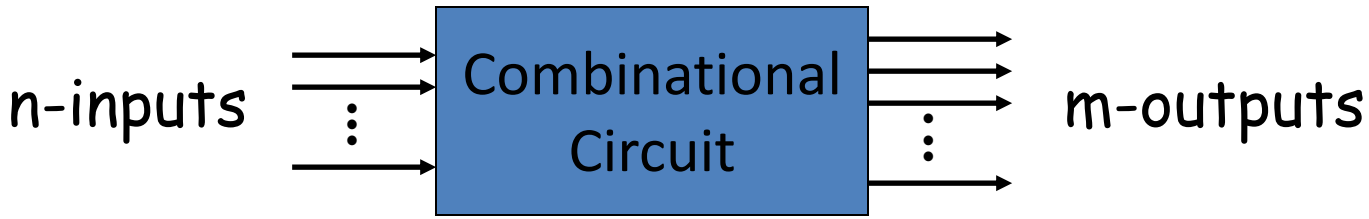
Hoda Roodaki
hroodaki@kntu.ac.ir

Combinational Circuits

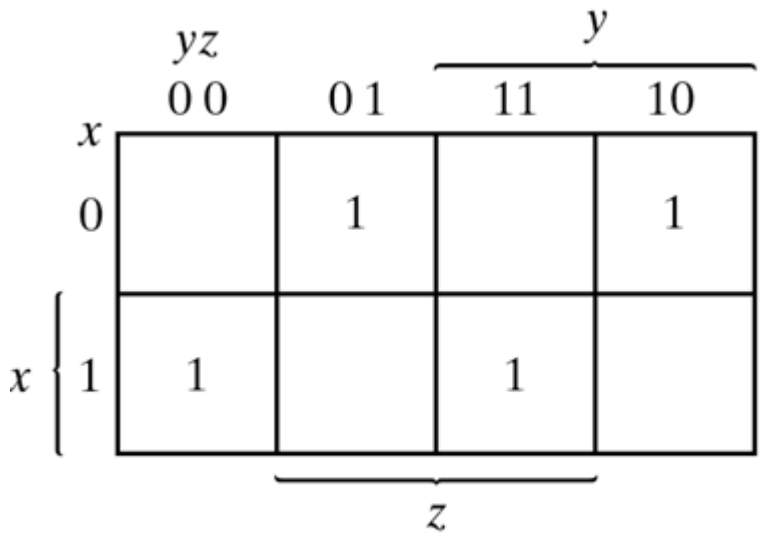
- A combinational circuit consists of logic gates whose outputs, at any time, are determined by combining the values of the inputs.
- For *n* input variables, there are 2^n possible binary input combinations.
- For each binary combination of the input variables, there is one possible output.

Combinational Circuits

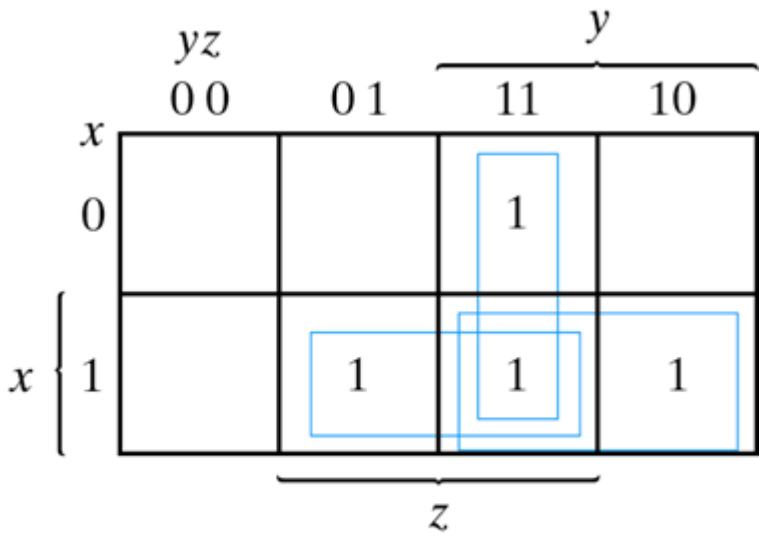
- A combinational circuit can be described by:
  1. A truth table that lists the output values for each combination of the input variables
  2. *m* Boolean functions, one for each output variable.



Full Adder

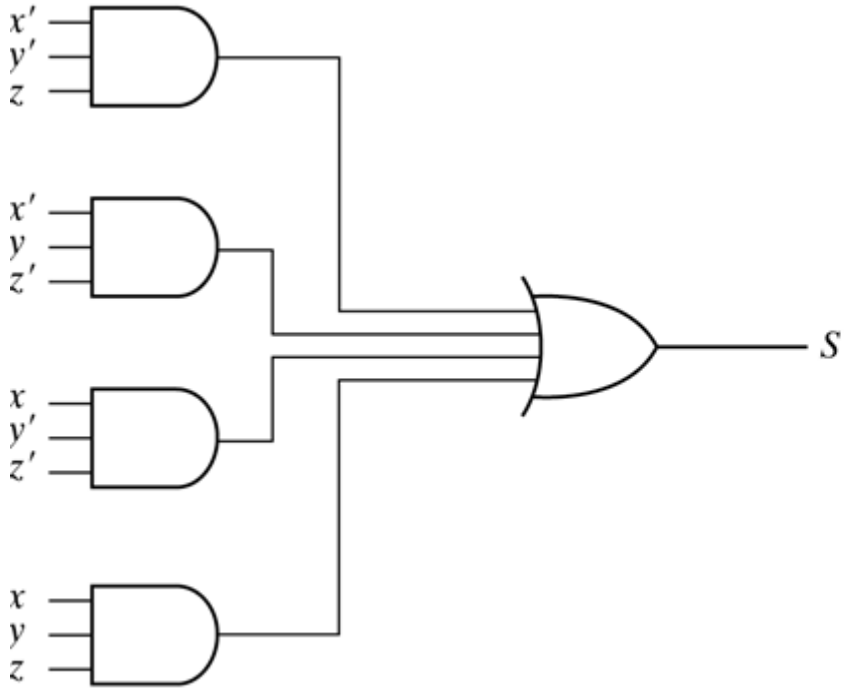


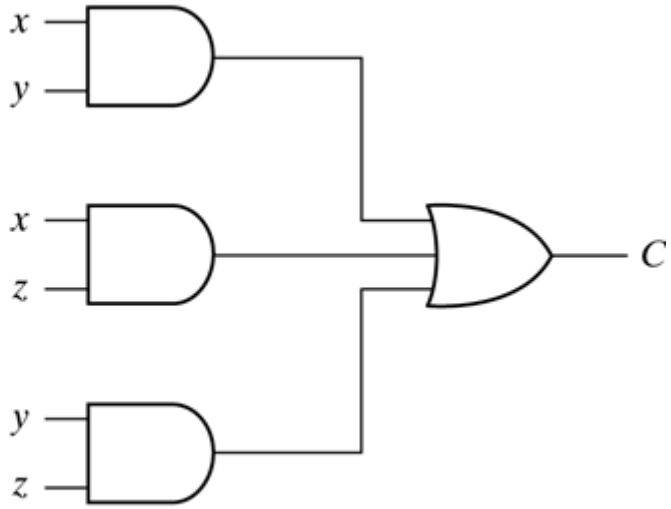
$$S = x'y'z + x'yz' + xy'z' + xyz$$



$$C = xy + xz + yz$$

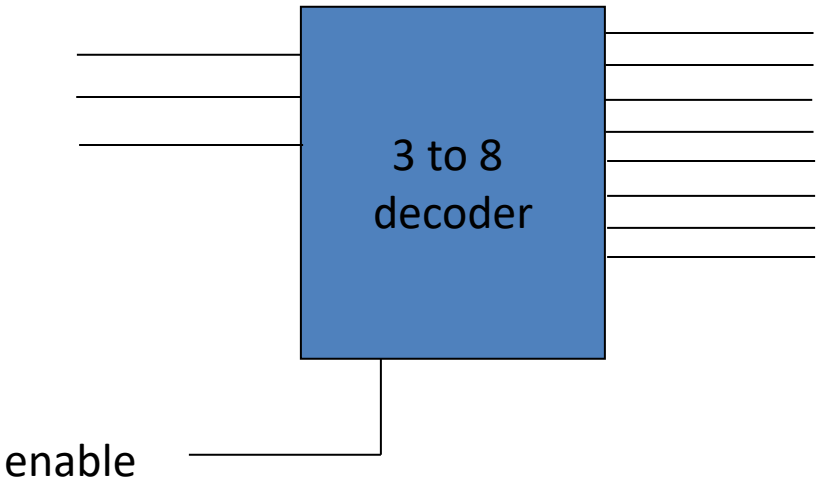
Full Adder





Decoder

- A combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines.
  - For example if the number of input is n=3 the number of output lines can be $m=2^3$.



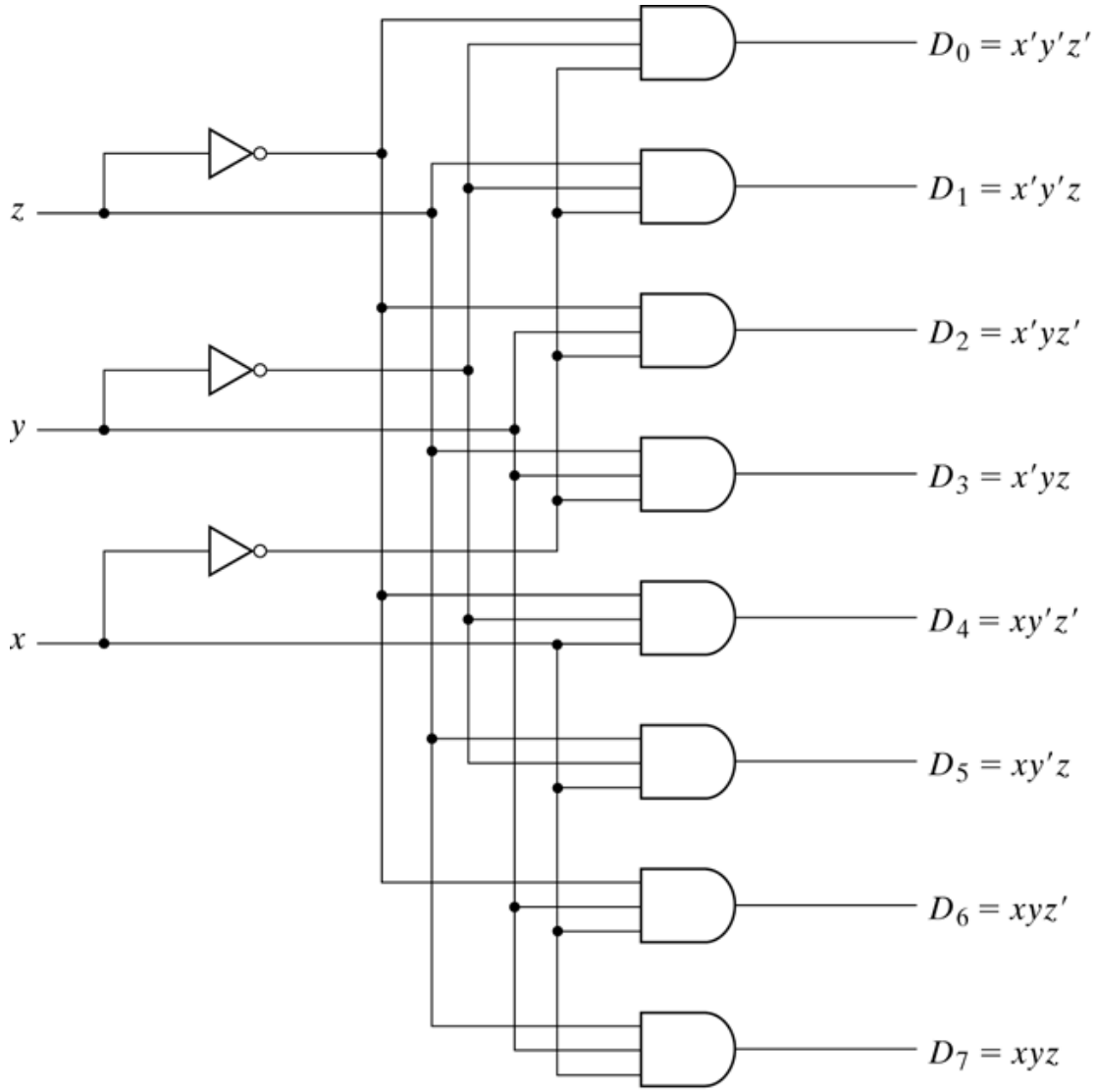
Decoder

- Decoders usually have an enable line
  - If enable=0 , decoder is off. It means all output lines are zero.
  - If enable=1, decoder is on and depending on input, the corresponding output line is 1, all other lines are 0.

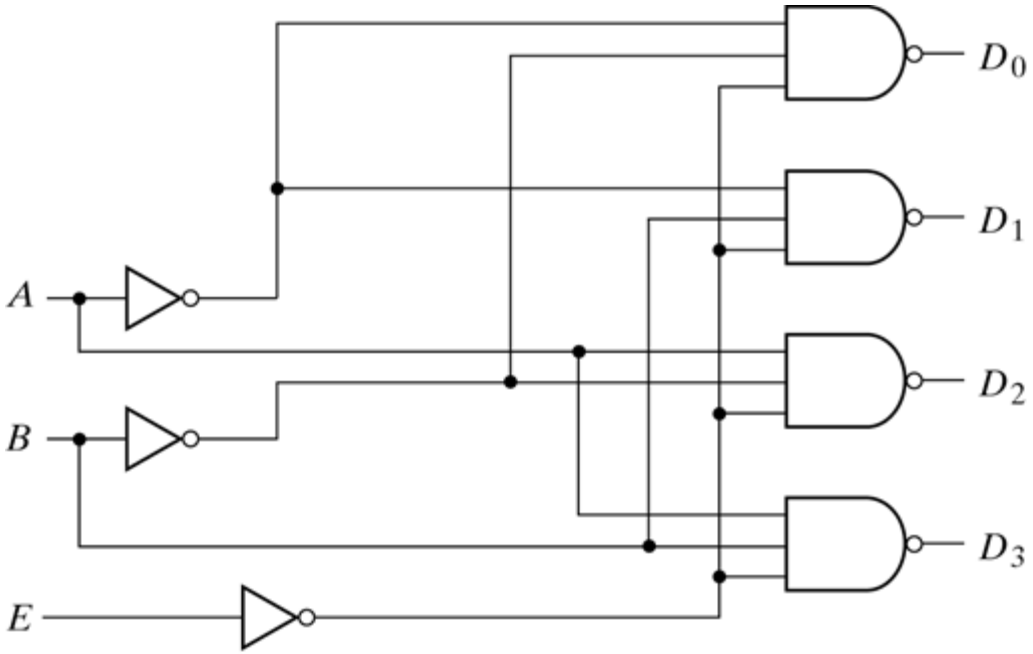
Decoder

E	a2	a1	a0	D7	D6	D5	D4	D3	D2	D1	D0
0	x	x	x	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1											
1		…….									
1		…….									
1											
1											
1	1	1	1	1	0	0	0	0	0	0	0

Decoder



Decoder



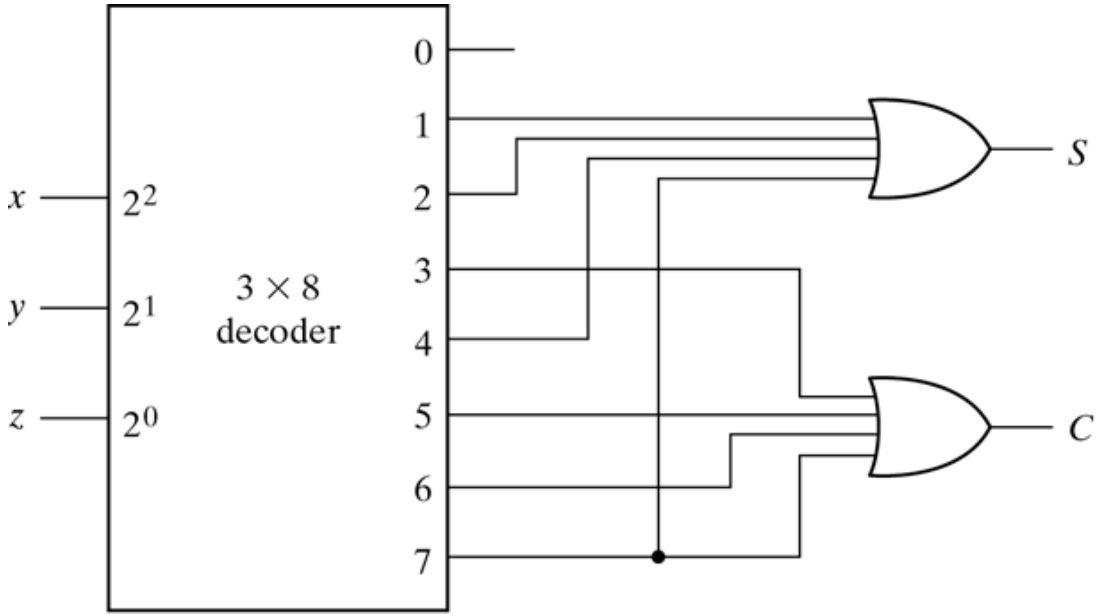
(a) Logic diagram

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(b) Truth table

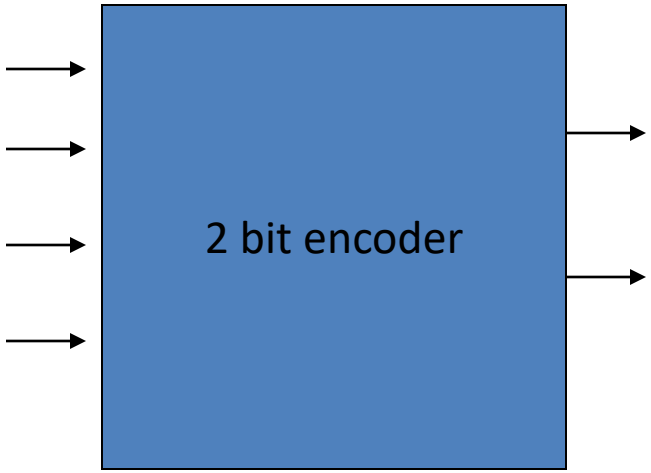
Decoder

- Decoder is use to implement any combinational circuits (f^n).
  - For example the truth table for full adder is s (x,y,z) = ∑ (1,2,4,7) and C(x,y,z)= ∑ (3,5,6,7).



Encoder

- Encoder is a digital circuit that performs the inverse operation of a decoder.
- Generates a unique binary code from several input lines.
- n bit encoder has 2^n input lines



Encoder

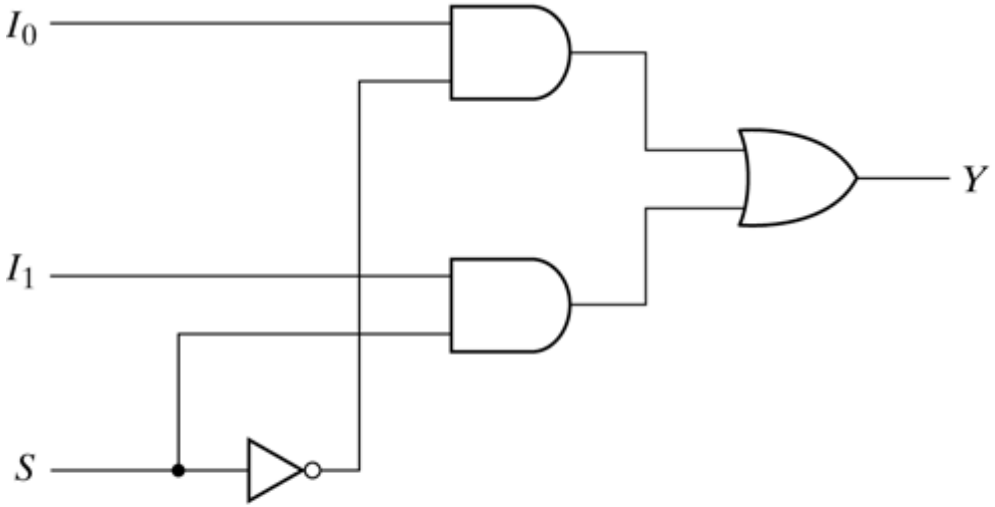
- If one of the four input lines is active encoder produces the binary code corresponding to that line.
- If more than one of the input lines will be activated, all the output is undefined.
  - We can consider don't care for these situations but in general we can solve this problem by using priority encoder.

Multiplexer

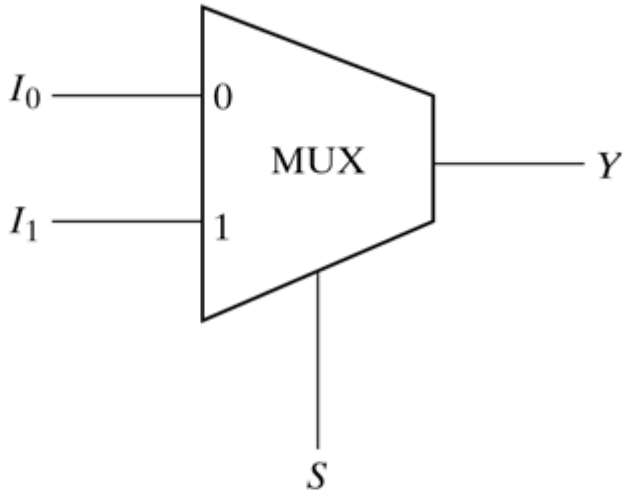
- It is a combinational circuit that selects binary information from one of the input lines and directs it to a single output line.
- Usually there are 2^n input lines and n selection lines whose bit combinations determine which input line is selected.
  - For example for 2-to-1 multiplexer if selection S is zero then I_0 has the path to output and if S is one I_1 has the path to output.

Multiplexer

2-to-1

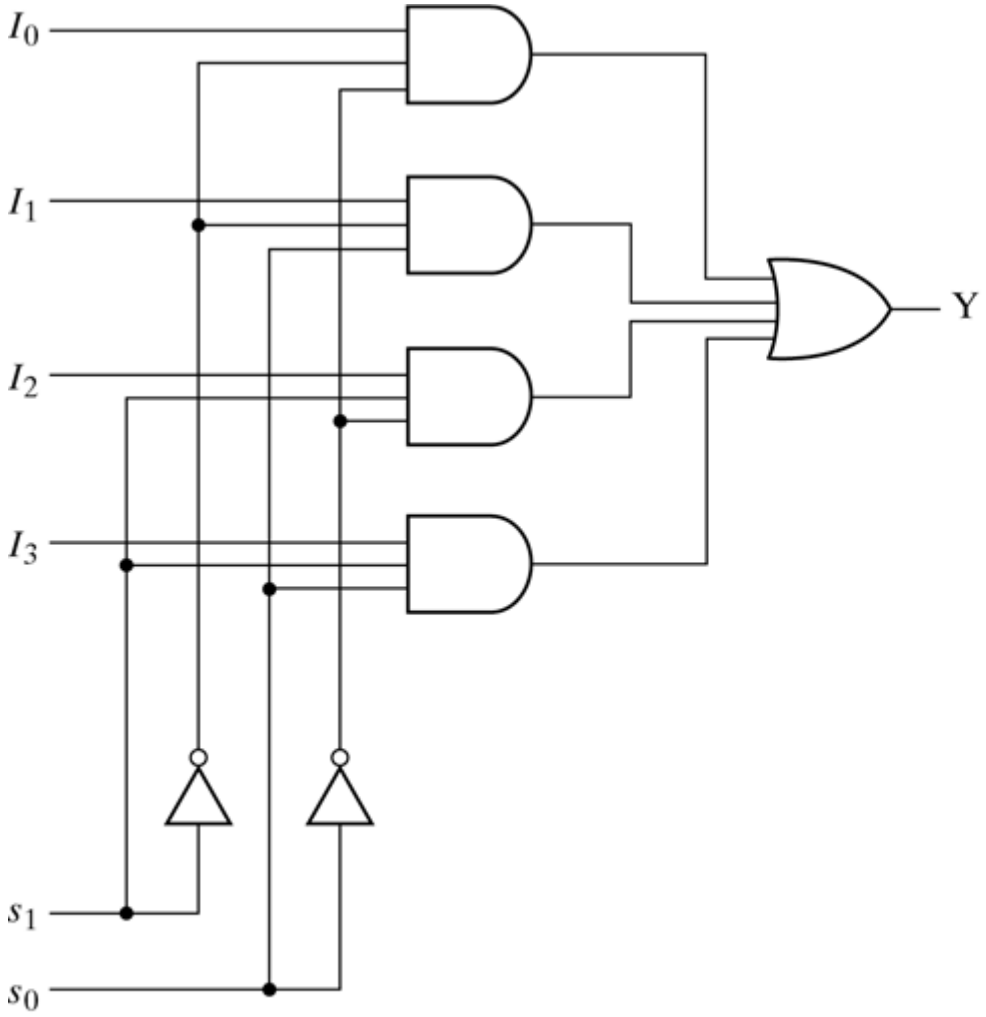


(a) Logic diagram



(b) Block diagram

Multiplexer



(a) Logic diagram

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) Function table

Demultiplexer

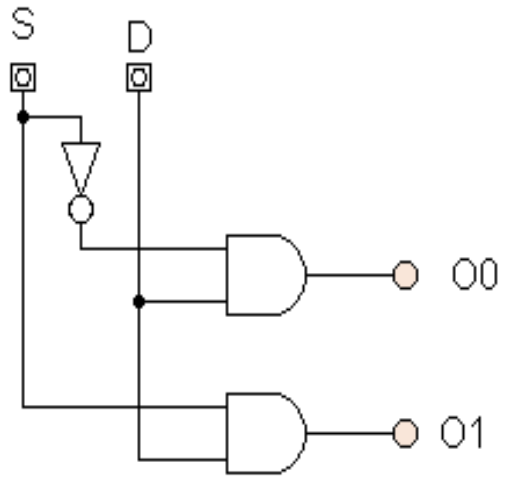
- A demultiplexer, sometimes abbreviated dmux, is a circuit that has one input and more than one output.
  - It is used when a circuit wishes to send a signal to one of many devices.

Demultiplexer

- Truth Table

S D	O_1 O_0
0 0	0 0
0 1	0 1
1 0	0 0
1 1	1 0

- Circuit



Basic Shifting

- Shift directions
  - Left (multiply by 2)
  - Right (divide by 2)
- Shift types
  - Logical (or unsigned)
  - Arithmetic (or signed)

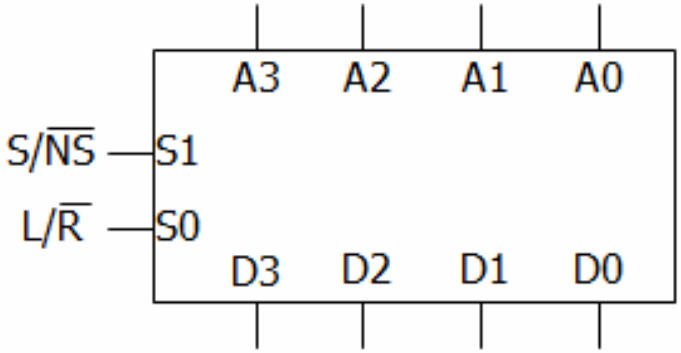
Logical Shift

- Shift Left
  - MSB: Shifted out
  - LSB: Shifted in with a “0”
  - Examples:
    - (11001011 << 1) = 10010110
    - (11001011 << 3) = 01011000
- Shift right
  - MSB: Shifted in with a “0”
  - LSB: Shifted out
  - Examples: (Some ISA use triple “>” for logical right shift)
    - (11001011 >> 1) = 01100101
    - (11001011 >> 3) = 00011001

Arithmetic Shift

- Shift left
  - MSB: Shifted out, however, be aware of overflow/underflow
  - LSB: Shifted in with a “0”
  - Examples:
    - (1100 << 1) = 1000
    - (1100 << 3) = 0000 (Incorrect!) $\Rightarrow$ Underflow
- Shift right
  - MSB: Retain “sign bit”
  - LSB: Shifted out
  - Examples:
    - (1100 >> 1) = 1110 (Retain sign bit)
    - (1100 >> 3) = 1111

4-bit Logical Shifter



S1	S0	D3	D2	D1	D0
0	X	A3	A2	A1	A0
1	0	0	A3	A2	A1
1	1	A2	A1	A0	0

$$D_3 = \overline{S_1} A_3 + S_1 S_0 A_2$$

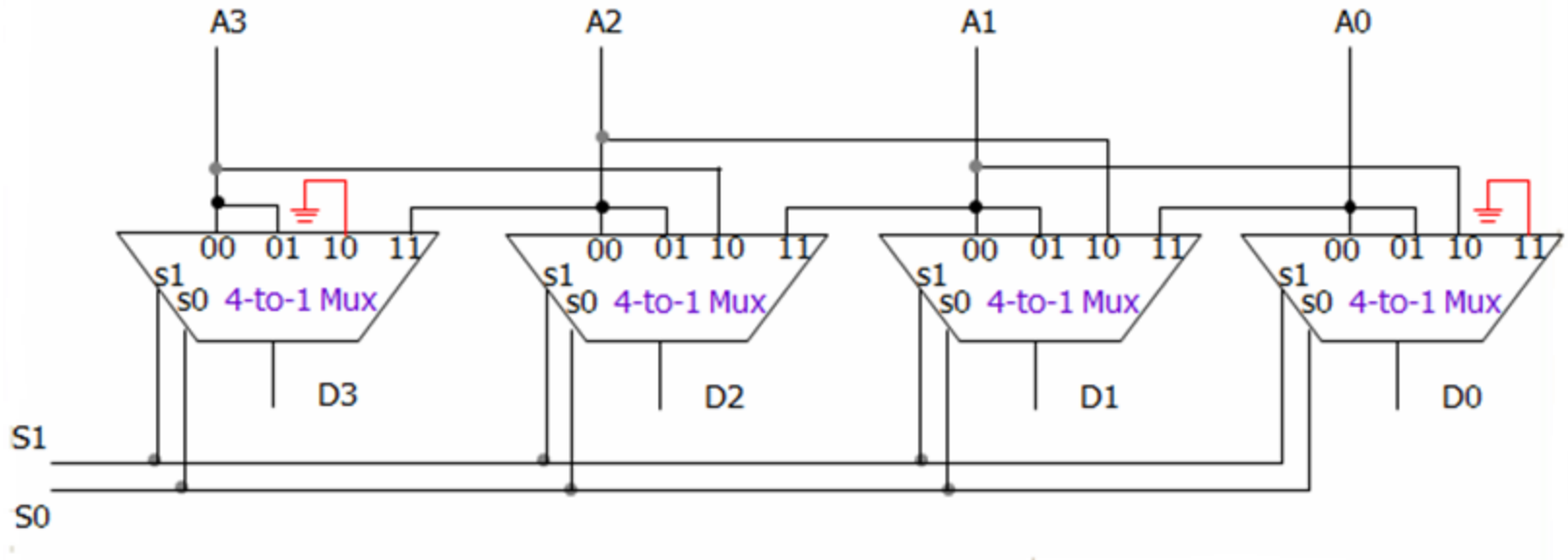
$$D_2 = \overline{S_1} A_2 + S_1 \overline{S_0} A_3 + S_1 S_0 A_1$$

$$D_1 = \overline{S_1} A_1 + S_1 \overline{S_0} A_2 + S_1 S_0 A_0$$

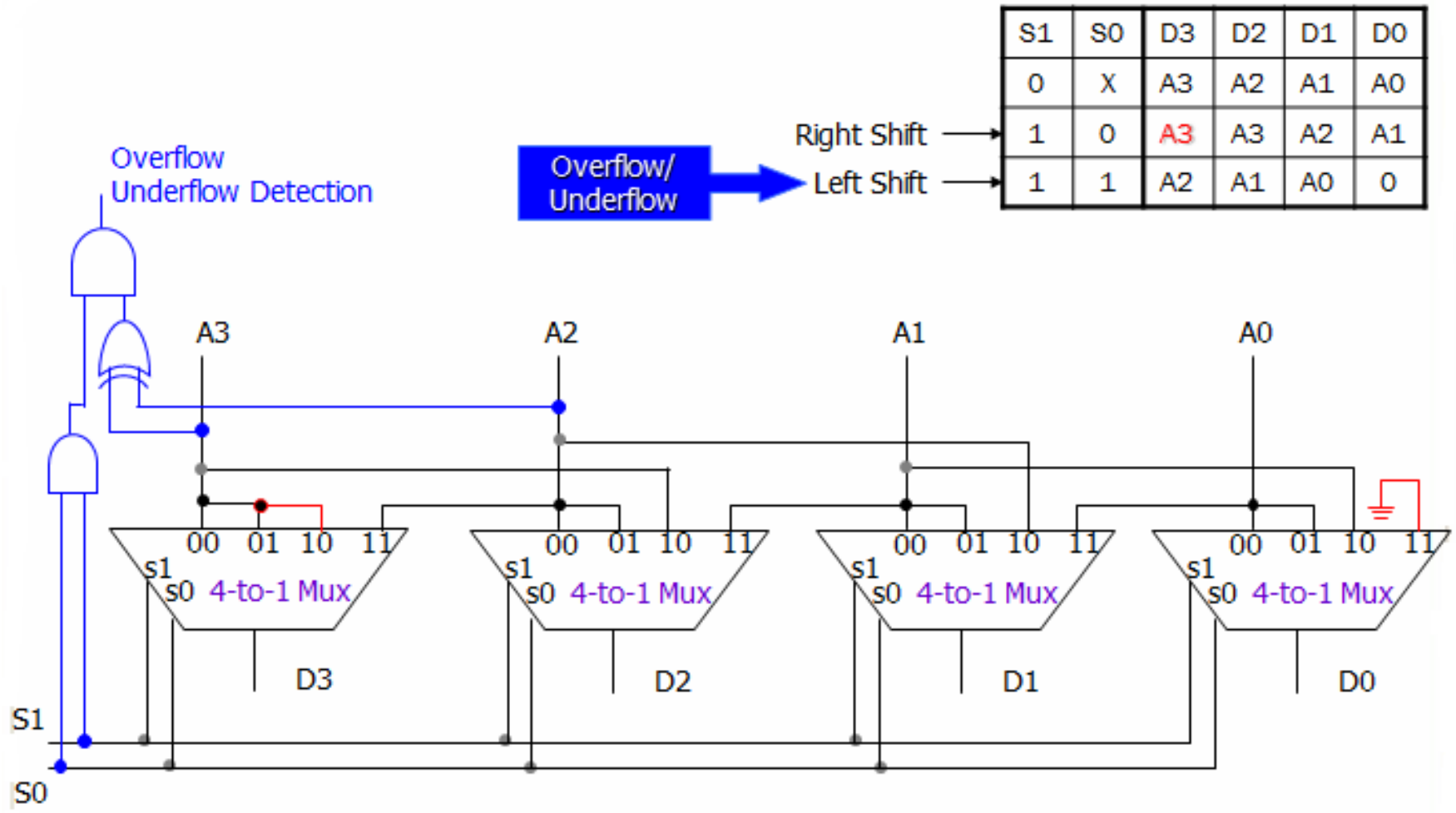
$$D_0 = \overline{S_1} A_0 + S_1 \overline{S_0} A_1$$

4-bit Logical Shifter using 4-to-1 Mux

	S1	S0	D3	D2	D1	D0
	0	X	A3	A2	A1	A0
Right Shift →	1	0	0	A3	A2	A1
Left Shift →	1	1	A2	A1	A0	0



4-bit Arithmetic Shifter w/ 4-to-1 Mux



S1	S0	D3	D2	D1	D0
0	X	A3	A2	A1	A0
1	0	A3	A3	A2	A1
1	1	A2	A1	A0	0

Sequential Circuits

- **Sequential logic** is a type of logic circuit whose output depends not only on the present value of its input signals but on the sequence of past inputs.
- In other words, the output state of a “sequential logic circuit” is a function of the following three states
  - the “present input”
  - the “past input”
  - and/or the “past output”.
- *Sequential Logic circuits* remember these conditions and stay fixed in their current state until the next clock signal changes one of the states, giving sequential logic circuits “Memory”.

Sequential Circuits

- The basic building blocks of combinational logic circuits are gates.
  - In particular, AND, OR, and NOT gates (however, there are also, XOR, NAND, NOR, XNOR gates too).
- The basic building blocks of sequential logic circuits are flip flops.
  - Flip flops are devices that use a clock. Each flip flop can store one bit.

Sequential Circuits

- Basically, a flip flop has two inputs. One input is a control input. For a D flip flop, the control input is labeled D.
- The other input is the clock.
- Flip-flops can be *positive edge-triggered flip flops*.
  - This means that the flip flops can only change output values when the clock is at a positive edge.
- There also can be negative edge triggered flip flops.
  - which change on a negative edge from the class notes on clock.
- When the clock is not at a positive or negative edge, then the output value is held. That is, it does not change.
- A flip flop also has two outputs, **Q** and **Q'**. The output is really the bit that's stored. Thus, the flip flop is always outputting the one bit of information.

Sequential Circuits

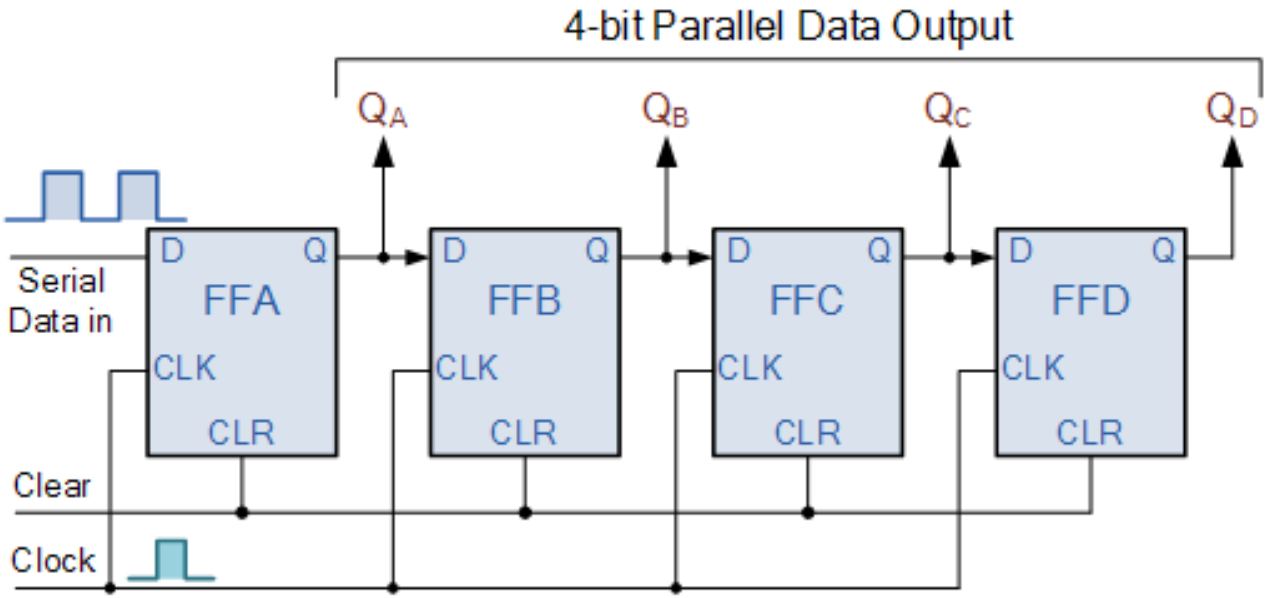
- **D Flip Flop Characteristic Table**

D	Q	Q^+	Operation
0	0	0	Reset
0	1	0	Reset
1	0	1	Set
1	1	1	Set

Shift Register

- The **Shift Register** is another type of sequential logic circuit that can be used for the storage or the transfer of data in the form of binary numbers.
- This sequential device loads the data present on its inputs and then moves or “shifts” it to its output once every clock cycle, hence the name “shift register”.

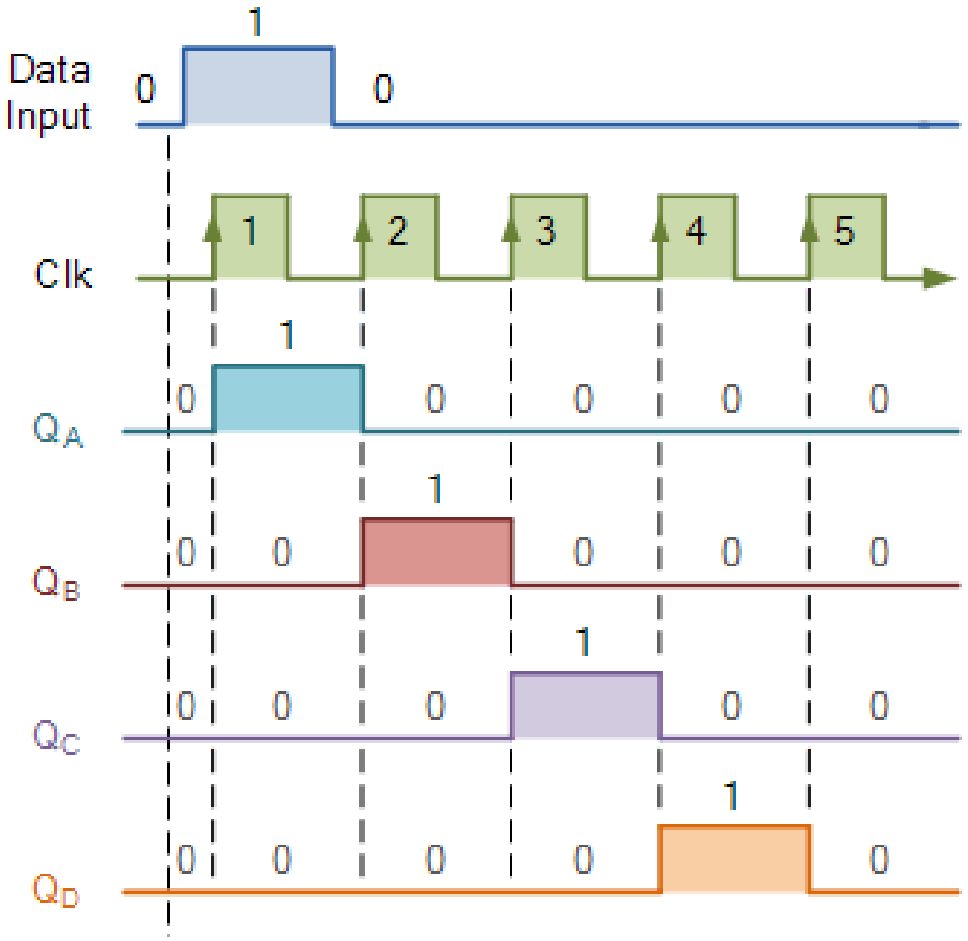
Serial-in to Parallel-out (SIPO) Shift Register



Serial-in to Parallel-out (SIPO) Shift Register

- Lets assume that all the flip-flops have just been RESET (CLEAR input) and that all the outputs Q_A to Q_D are at logic level “0”.
- If a logic “1” is connected to the DATA input pin of FFA then on the first clock pulse the output of FFA and therefore the resulting Q_A will be set HIGH to logic “1” with all the other outputs still remaining LOW at logic “0”. Assume now that the DATA input pin of FFA has returned LOW again to logic “0” giving us one data pulse or 0-1-0.
- The second clock pulse will change the output of FFA to logic “0” and the output of FFB and Q_B HIGH to logic “1” as its input D has the logic “1” level on it from Q_A. The logic “1” has now moved or been “shifted” one place along the register to the right as it is now at Q_A.
- When the third clock pulse arrives this logic “1” value moves to the output of FFC (Q_C) and so on until the arrival of the fifth clock pulse which sets all the outputs Q_A to Q_D back again to logic level “0” because the input to FFA has remained constant at logic level “0”.
- The effect of each clock pulse is to shift the data contents of each stage one place to the right, and this is shown in the following table until the complete data value of 0-0-0-1 is stored in the register. This data value can now be read directly from the outputs of Q_A to Q_D.
  - Then the data has been converted from a serial data input signal to a parallel data output. T
  - he truth table and following waveforms show the propagation of the logic “1” through the register from left to right as follows.

Serial-in to Parallel-out (SIPO) Shift Register



Parallel-in to Serial-out (PISO) Shift Register

- The Parallel-in to Serial-out shift register acts in the opposite way to the serial-in to parallel-out one above. The data is loaded into the register in a parallel format in which all the data bits enter their inputs simultaneously, to the parallel input pins P_A to P_D of the register. The data is then read out sequentially in the normal shift-right mode from the register at Q representing the data present at P_A to P_D.
- This data is outputted one bit at a time on each clock cycle in a serial format. It is important to note that with this type of data register a clock pulse is not required to parallel load the register as it is already present, but four clock pulses are required to unload the data.

Parallel-in to Serial-out (PISO) Shift Register