1007ICT / 1807ICT / 7611ICT Computer Systems & Networks

3C. Digital Logic and Digital Circuits

Dr. Sven Venema

Dr. Vallipuram Muthukkumarasamy

Last Lecture:

Topics Covered:

- Logic unit, Selection logic, Decoder logic

- Multiplexing and Demultiplexing

- Half and Full adders

Lecture Content

- Learning objectives
- Arithmetic logic unit
- Binary multiplication and division
- Shifting
- Sequential Logic
- Data latches, S-R Latch
- Clocks and synchronisation
- Registers, Buses, Computer memory

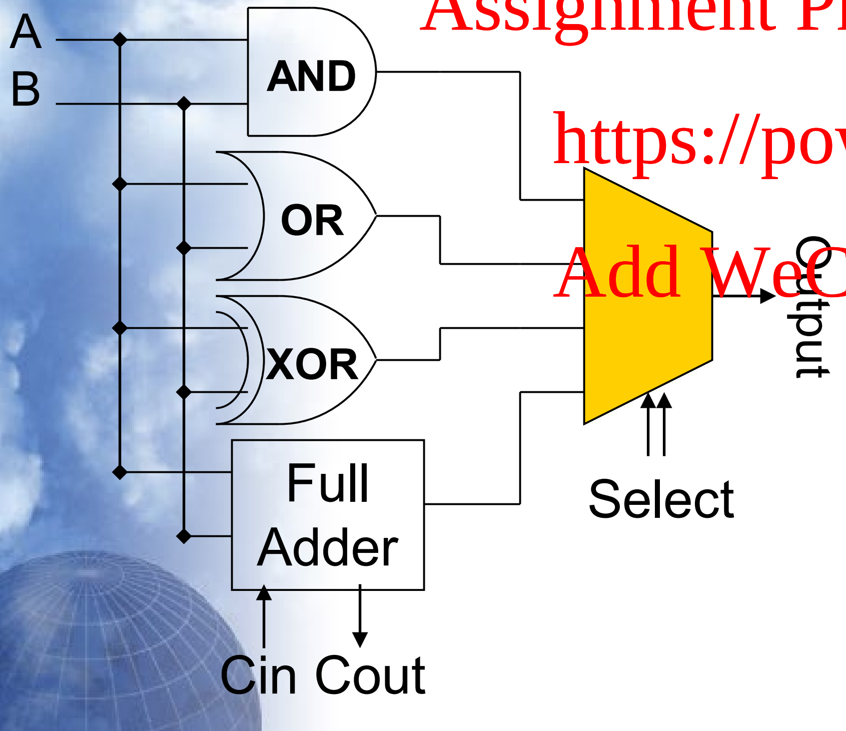
Learning Objectives

At the end of this lecture you will have gained an understanding of:

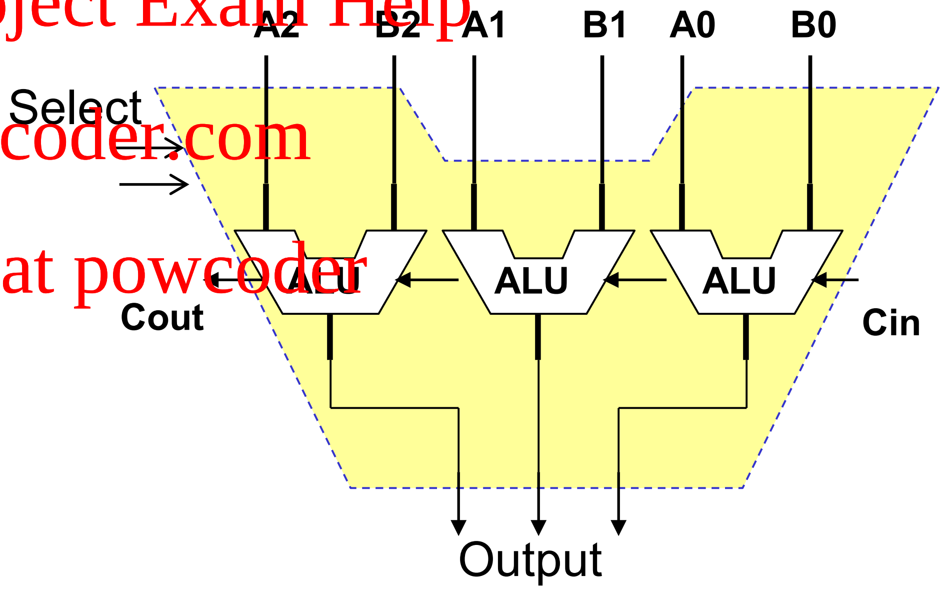
- Arithmetic logic unit
- Binary multiplication and division
- Shifting
- Sequential Logic
- Data latches, S-R Latch
- Clocks and synchronisation
- Registers, Buses, Computer memory

ALU – Arithmetic Logic Unit (Section 7.2)

- The ALU is a general processing element
- It puts everything we have learnt together
- ALUs are combined in parallel for multi-bit versions



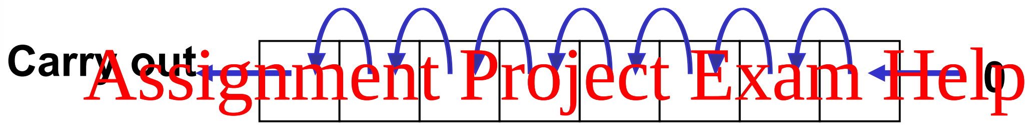
1 bit ALU



3 bit ALU

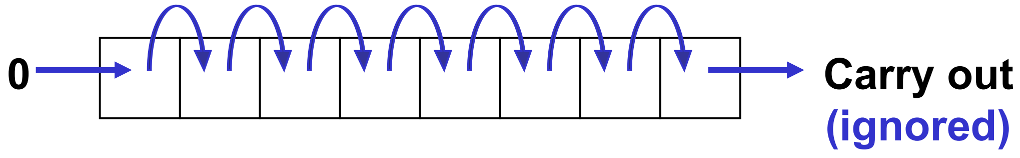
Multiplication and Division By 2

- To multiply by 2 in binary, shift bits left and insert a zero at the right hand side.



$0001_2 = 1_{10}$
$0010_2 = 2_{10}$
$0100_2 = 4_{10}$

- To divide by 2 in binary, shift bits right and insert a zero at the left hand side.



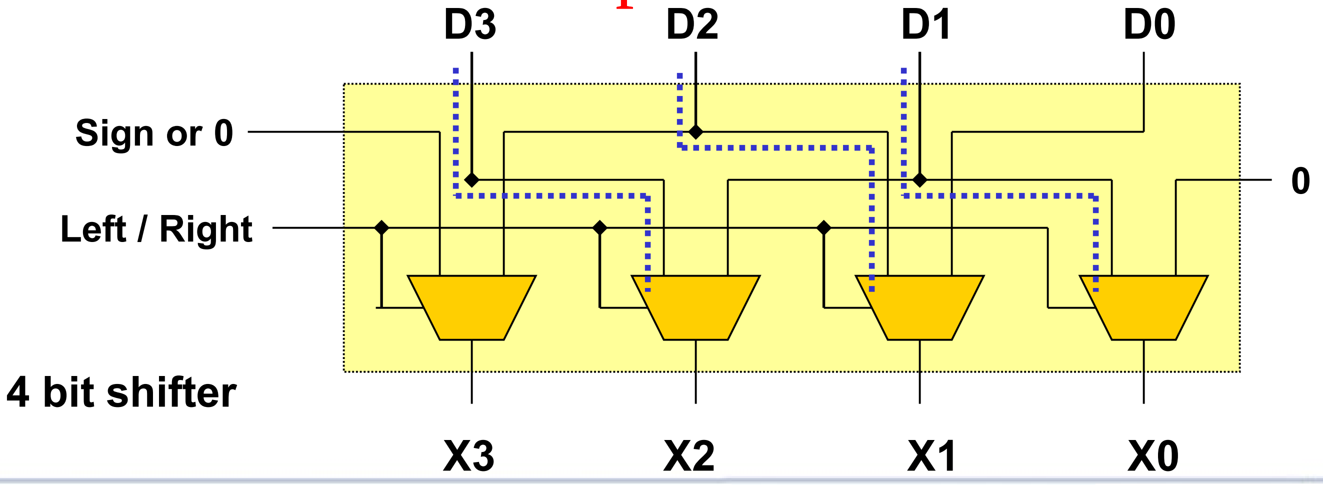
$0111_2 = 7_{10}$
$0011_2 = 3_{10}$
$0001_2 = 1_{10}$

Shifters

- We can use multiplexors to shift bits left and right.
- We need to select each output bit to be the input bit on its left or right side

Inputs	D4	D3	D2	D1	D0
Shift Left					
Outputs	X3	X3	X2	X1	X0

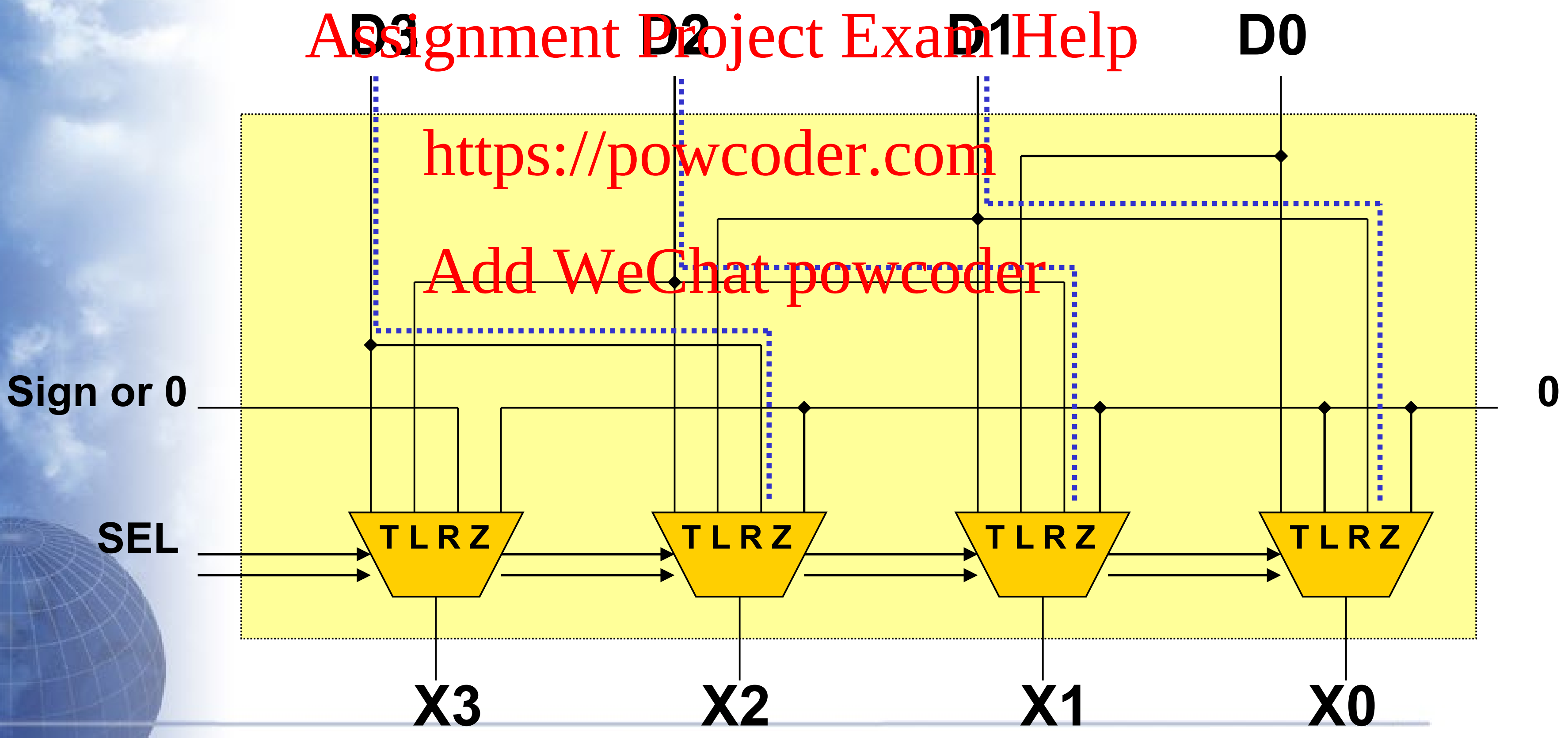
Inputs	D4	D3	D2	D1	D0
Shift Right					
Outputs	X3	X3	X2	X1	X0



Shifters

- What if we also want the shifter to not shift or to set the output to zero? Use a 4-input multiplexor

SEL	Out
00	Zero
01	Right
10	Left
11	Thru



Arbitrary Multiplication

- To multiply by other numbers say A * B – we could add A to itself B times, but it's faster to use shifts and adds like this…
- The powers of 2 you need to add to get the multiplicand determines the number combination of shifts and addition we need to perform on the multiplier (ie the 1s in its binary value)

- Let << denote the binary left shift operator:
- 10 x 5 = (10 x 4) + (10 x 1) = 110010_2

5 = 101_2
= $100_2 + 1_2$
= 4 + 1

Mult.	Shifts	Binary	Hex
10 x 4	1010 << 2	101000	28
10 x 1	1010 << 0	001010	0A

- 10 x 7 = (10 x 4) + (10 x 2) + (10 x 1) = 1000110_2

7 = 111_2
= $100_2 + 10_2 + 1_2$
= 4 + 2 + 1

Mult.	Shifts	Binary	Hex
10 x 4	1010 << 2	101000	28
10 x 2	1010 << 1	010100	14
10 x 1	1010 << 0	001010	0A

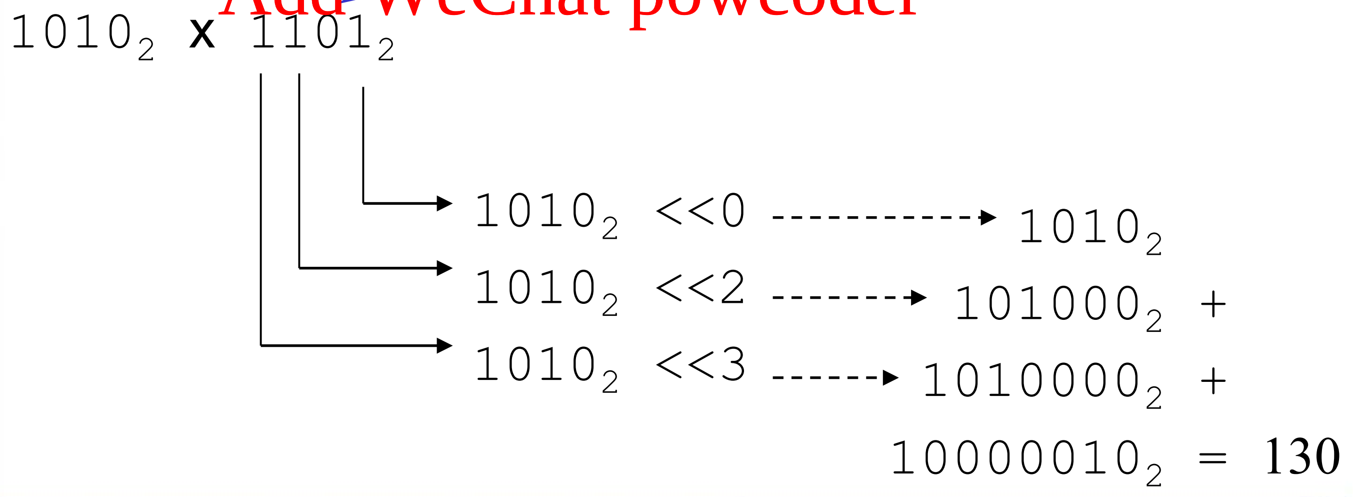
Multiplication Example

Multiply Sum ← A x B

- Check every bit position of B so that if:
  - **bit position 0 of B is 1, add A to the sum.**
  - **bit position 1 of B is 1, add A << 1 to the sum.**
  - **bit position 2 of B is 1, add A << 2 to the sum.**
  - **bit position 3 of B is 1, add A << 3 to the sum.**
  - **etc..**

*Note: **1010_2** << 1 is not added as bit position 1 of 1101_2 is 0.*

$10 \times 13 = 130$



Arbitrary Division

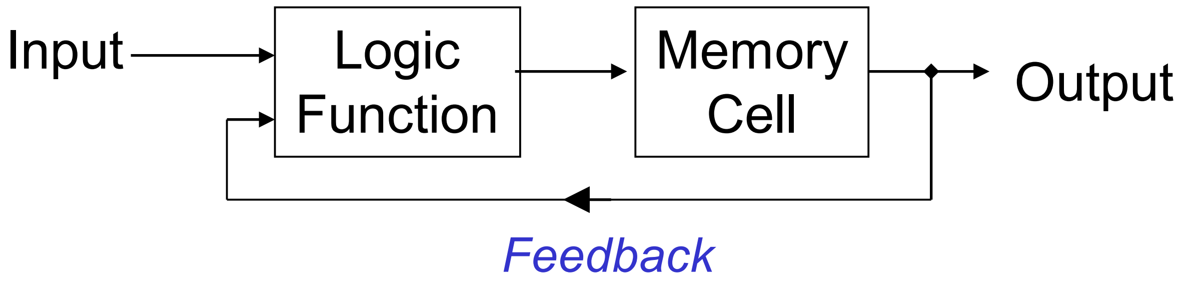
- For arbitrary division we can use basic binary long division.

```
          8 4 2 1       2 1

13 / 3  = 1 1 0 1   /   1 1

        = 4 Remainder 1
```

```
              1 0 0    →   Answer: 4
          _____________
1 1   |     1 1 0 1

          - 1 1   ↓  ↓
            ___
              0 0 1    →   Remainder: 1
```

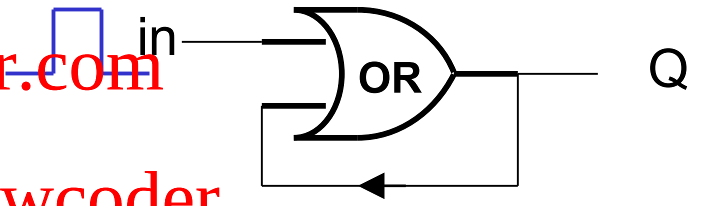
Sequential Logic (Section 2.5)

- Previously we looked at ***combinatorial*** logic which produces an output as some combination of the input values.
- ***Sequential*** logic produces an output that depends not only on the inputs but on previous inputs as well.

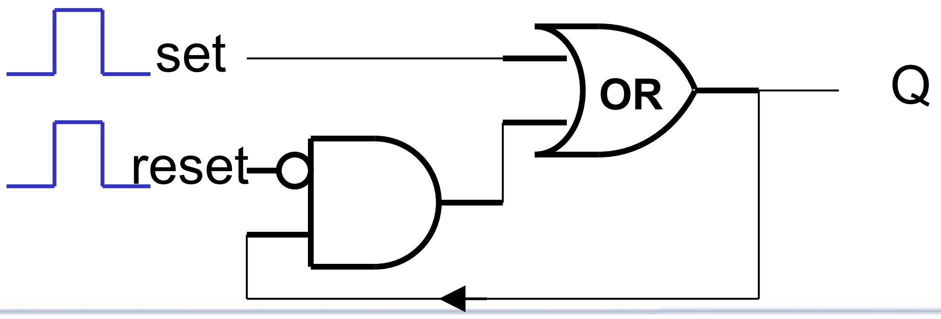


Data Latches

- None of the logic functions we saw before can store any data… bits come in and go straight out again.
- A ***Latch*** is a logic function that can store bits
- Consider the following logic:

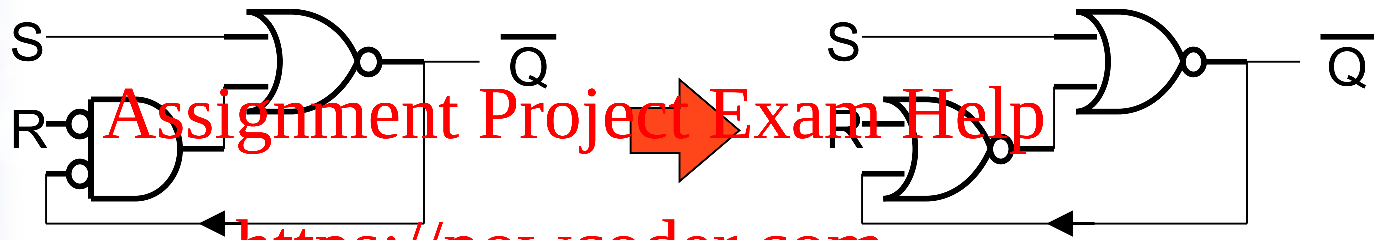


- This can store a ‘1’ bit but not a ‘0’ bit. We need another input to tell it to store a ‘0’bit, like this.

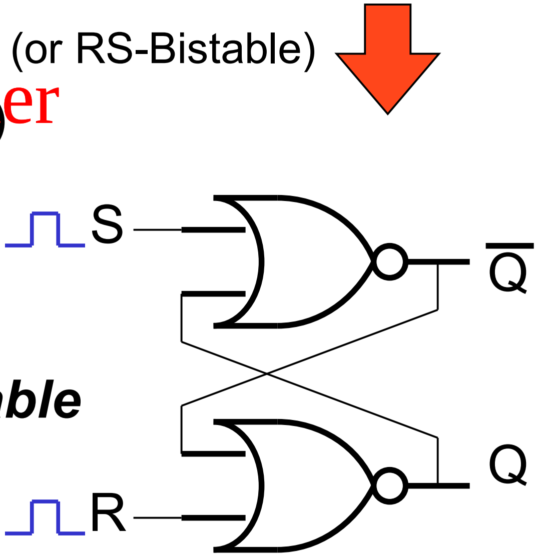


S-R Latch (RS-Bistable)

- Let’s rearrange the logic a bit as follows using the rules of logic (using DeMorgan’s Theorem) :

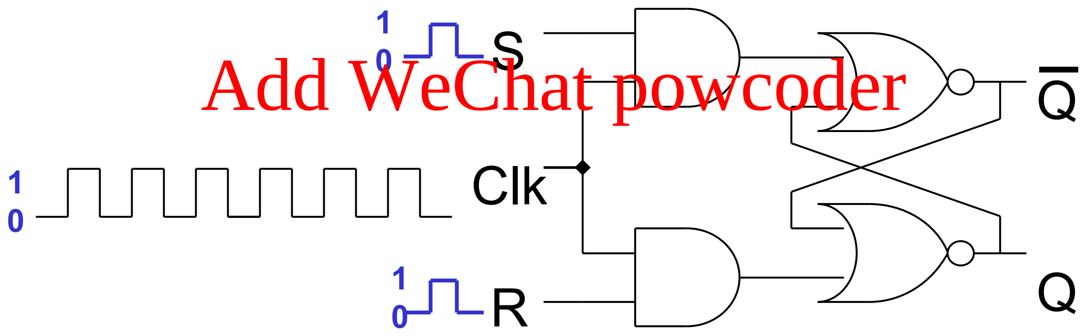


- The result is called the SR-latch (or RS-Bistable)
  - Both inputs are normally OFF (at 0)
  - The S input sets the data state to 1
  - The R input sets the data state to 0
  - Having both to ON at once is illegal
  - When both are OFF the latch is ***stable***
    - Either Q = 0 and $\overline{Q}$ = 1
    - Or Q = 1 and $\overline{Q}$ = 0



Clocks and Synchronisation

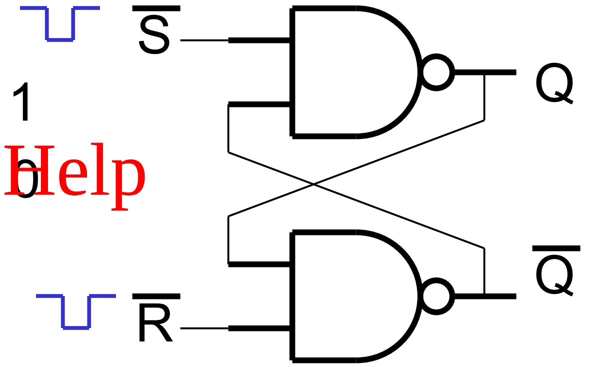
- The main problem with latches is that after the inputs change the output is unstable for a short time.
- If we directly connect the output of a latch to the input of another circuit then the circuit may be unstable
- A clock signal can control when a latch can load its inputs after they are stable (ie when the clock is high)



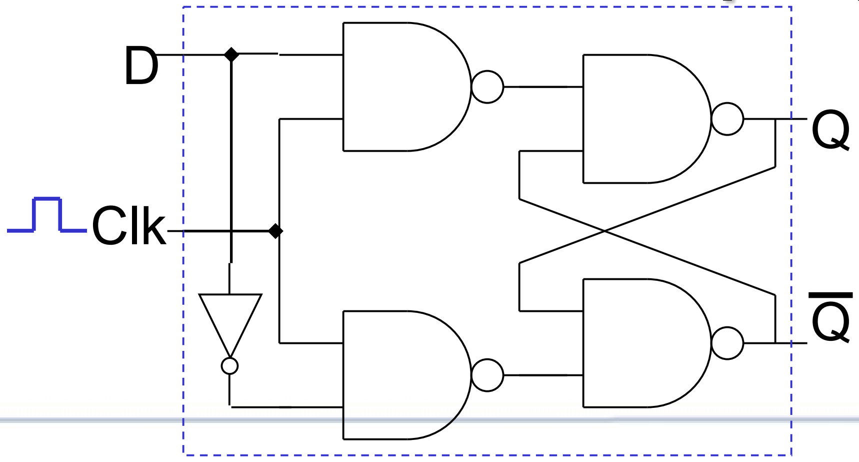
- Clocked latches are sometimes called ***Flip-Flops;***
- The faster the clock – the faster that data can be stored/read into a latch

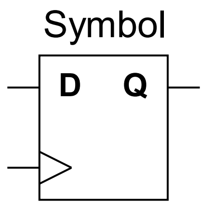
Data Latch (Level-Triggered D-Type bistable)

- We can use DeMorgan's Theorem again to convert the Latch to use NAND gates that is set using OFF inputs.
  - Both inputs are normally ON
  - Setting S to OFF sets the data state to 1
  - Setting R to OFF sets the data state to 0
  - Having both to OFF at once is illegal
  - When both are ON the latch is ***stable***



- A simple modification of the clocked version lets us create the useful ***D-Latch*** or ***D-FlipFlop***



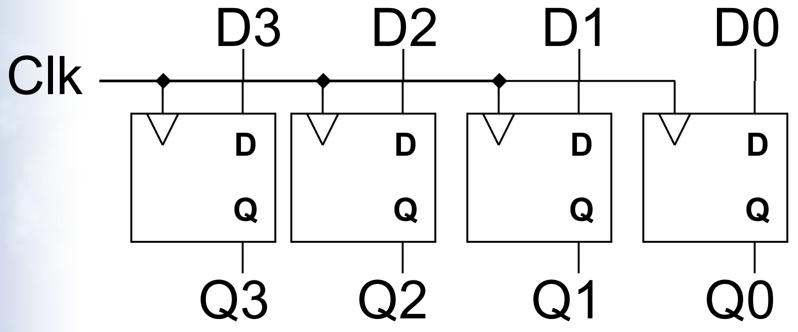


Bits / Bytes / Words

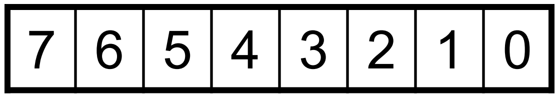
- If computer memory was only addressed one bit at a time, it would take quite a long time to retrieve enough data to any serious processing.
- Most computers do not process individual bits but instead group them together into multiples of 8 eg 8, 16, 32, 64 etc.
- These groups are described as ***words***, but word length varies depending on the machine.
- An 8 bit word is called a ***Byte***.
- A 4 bit word is called a ***Nibble*** so 1 byte = 2 nibbles
- There are no special names for other sized words.

Registers

- Data Latches (Level-Triggered D-Type bistables) allow us to store 1 bit of information but we can group them in parallel
- This is called a ***register*** to store data consisting of multiple bits.
- The **wordsize** is the number of bits a register can store



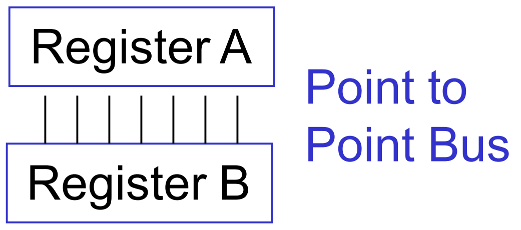
Simple 4 bit register using D-Latches

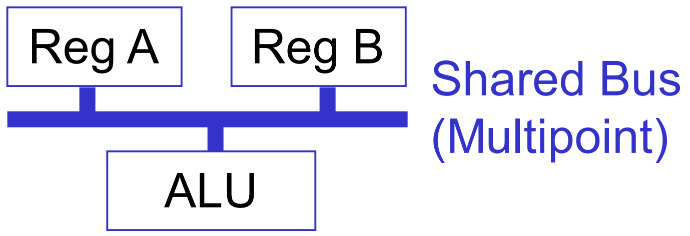


8 bit register

Buses

- We can connect logic gates together with lines that convey single bits of information between them.
- With registers we have multiple input and output lines to convey N-bits of data to and from the register
- A **bus** is a set of lines that simultaneously conveys a set of bits between components.
- Two types of buses are common in computer systems: *point-to-point*, and *multipoint* buses.





Multipoint Buses use special gates with "tristate" outputs that can be connected together

Computer Memory (Section 3.3)

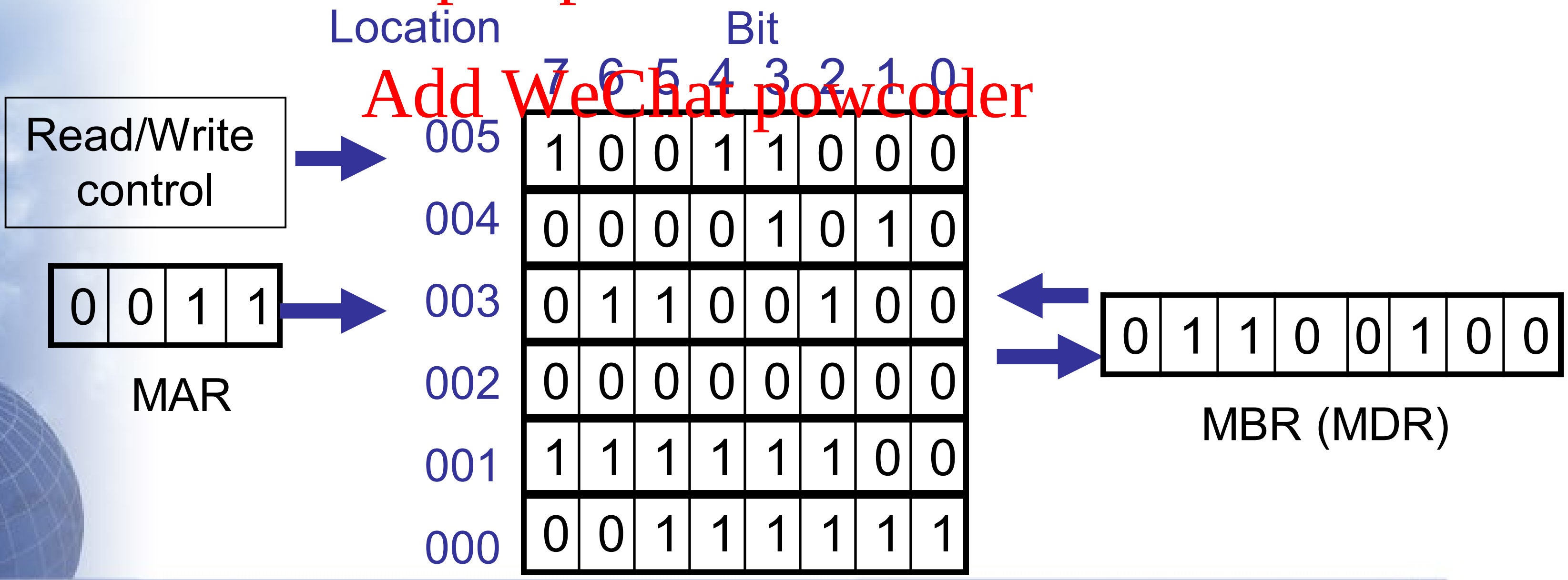
- We can use an array of registers in series to create a memory bank.
- Every register location in the memory bank is given a unique address that is used so that we can select it and access the data stored in it.
- An 8 bit machine would use a bank of 8 bit registers and access 8 bit words from memory at a time.

Fetching word at address 003 would return the value $01100100_2 = 64_{16} = 100_{10}$

Location	Bit 7	6	5	4	3	2	1	0
005	1	0	0	1	1	0	0	0
004	0	0	0	0	1	0	1	0
003	0	1	1	0	0	1	0	0
002	0	0	0	0	0	0	0	0
001	1	1	1	1	1	1	0	0
000	0	0	1	1	1	1	1	1

Computer Memory

- A memory bank needs some way of selecting memory addresses
- A special register called a memory address register (***MAR***) which is internal to the CPU contains the physical location of the next memory address that will be selected for reading/writing.
- Another internal register called the Memory Buffer Register (***MBR***) [or Memory Data Register (***MDR***)] holds the value that was read /written to the selected memory address.



Summary

Have considered:

- Arithmetic logic unit
- Binary multiplication and division
- Shifting
- Sequential Logic
- Data latches, S-R Latch
- Clocks and synchronisation
- Registers, Buses, Computer memory

Next….

- Processors