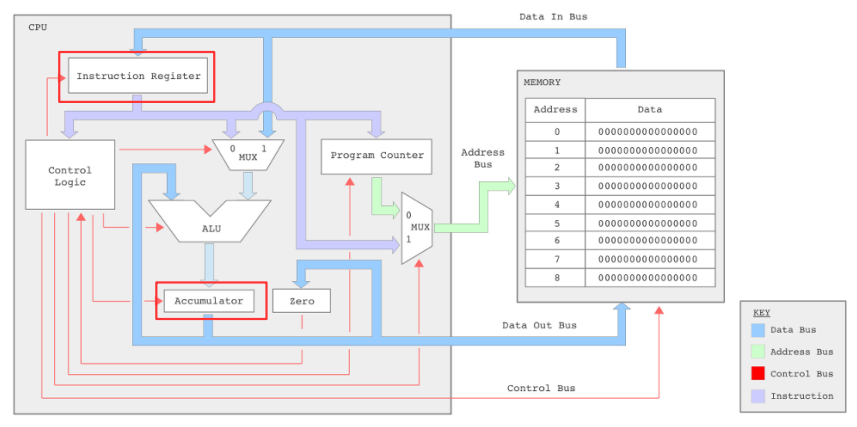
Systems and Devices 1
Lec 4 : Sequential Logic

Before we get started ...

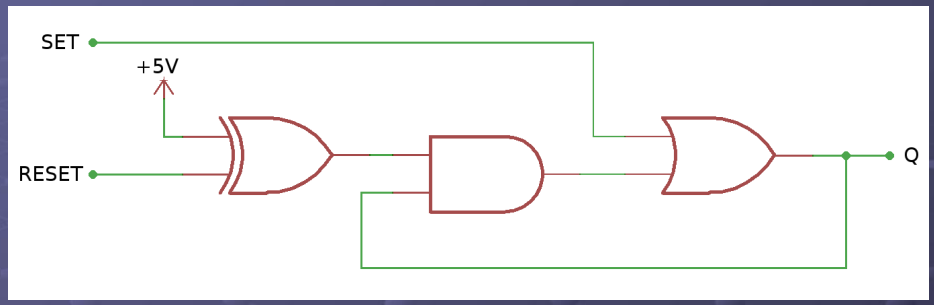
- Combinational logic : outputs are a function of the present inputs.
  - The same input will always produce the same output.
- Sequential logic : outputs are dependent on the present inputs and past inputs.
  - The same input may produce different outputs as these logic circuits have an internal state (memory).
- A key building block of any computer is memory, the ability to store the results of past calculations, data …
  - Note, memory performance and a computer's architecture are just as important as the data processing hardware i.e. ALU is only as fast as you can pass data to it

SimpleCPU_v1a



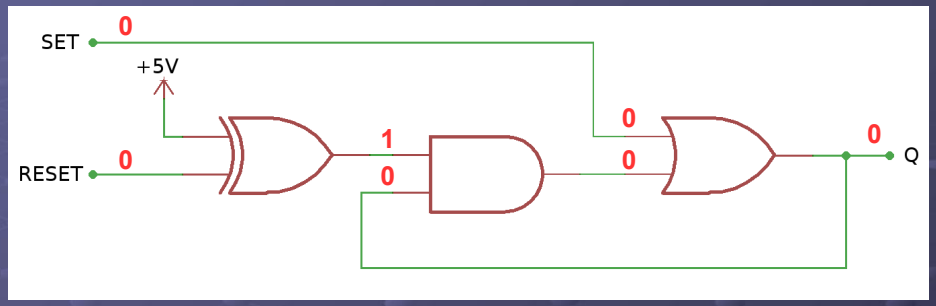
- Block diagram
  - Registers : to execute a program a computer needs to remember what its is doing and the data it is processing.

Flip-Flop



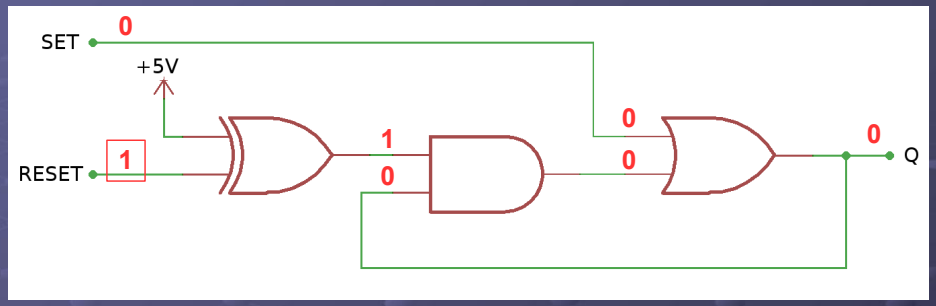
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



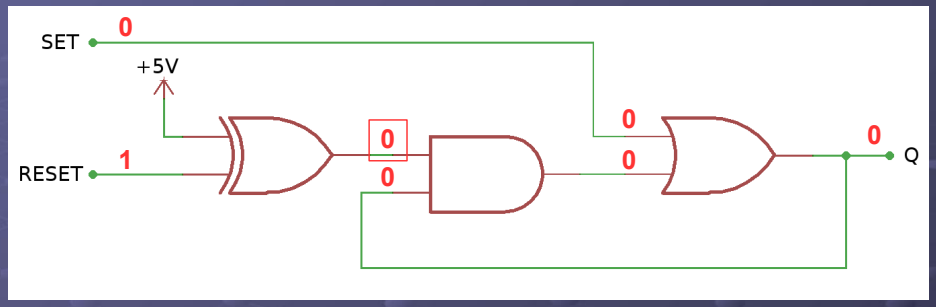
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



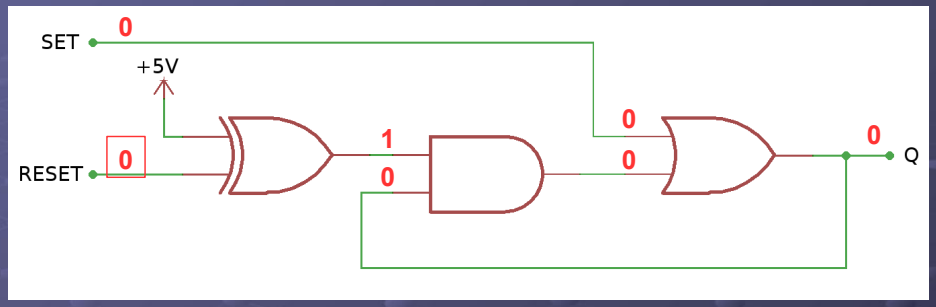
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



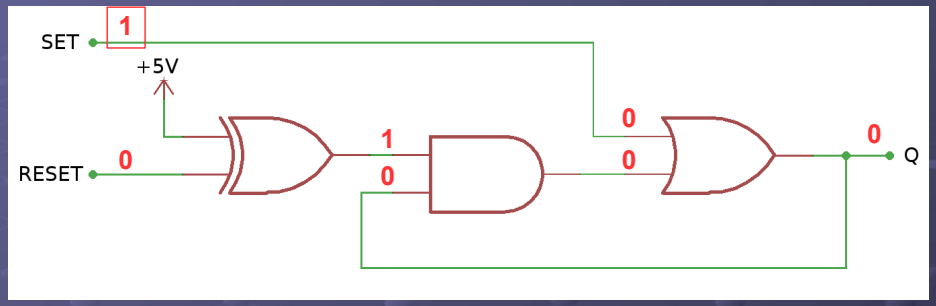
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



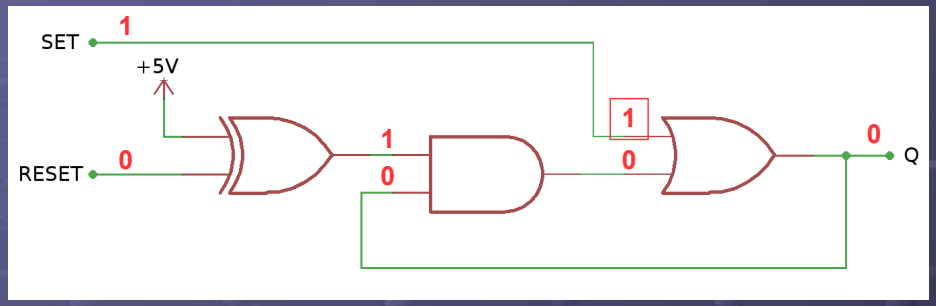
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



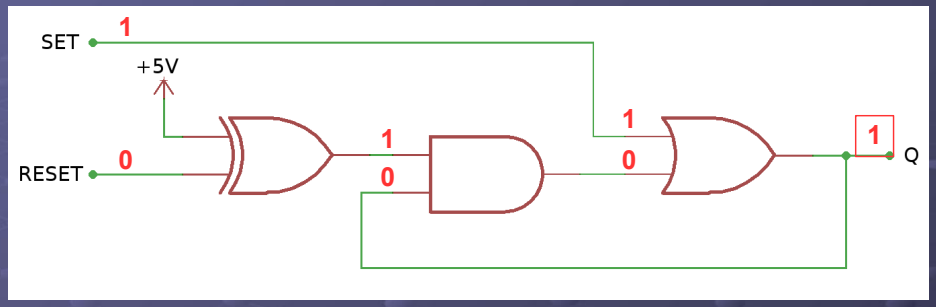
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



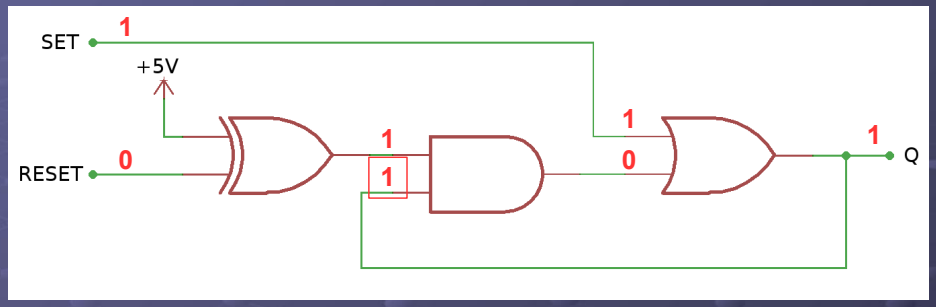
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



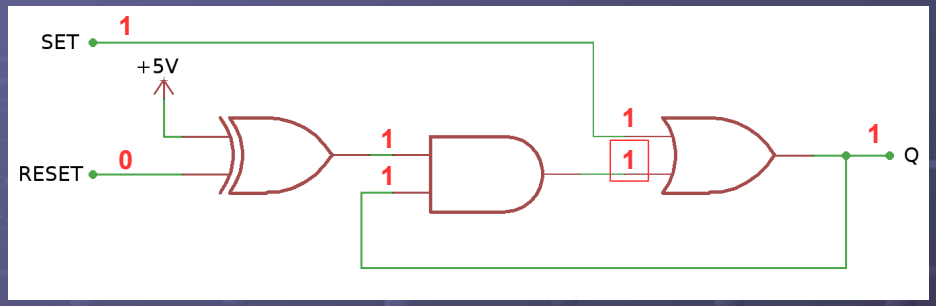
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



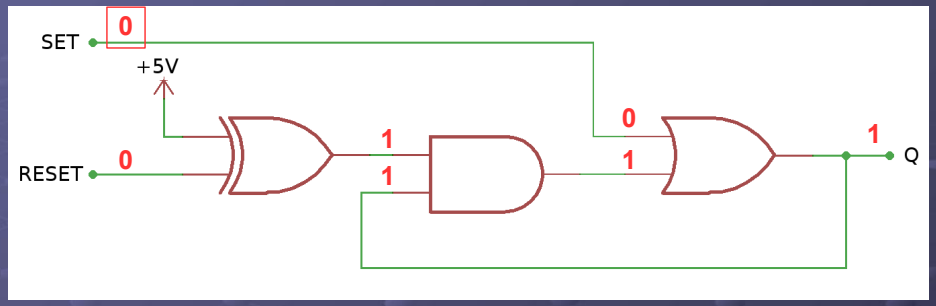
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



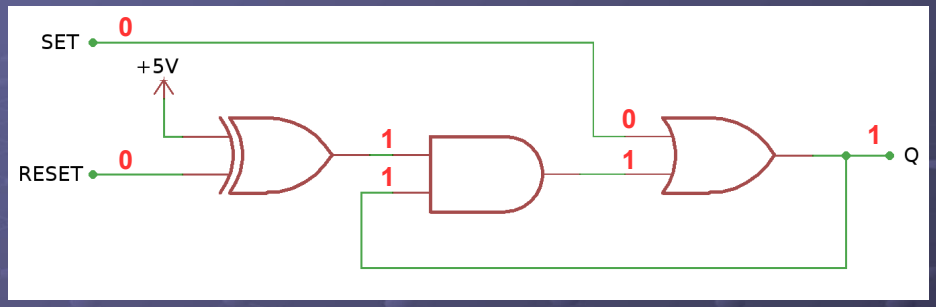
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



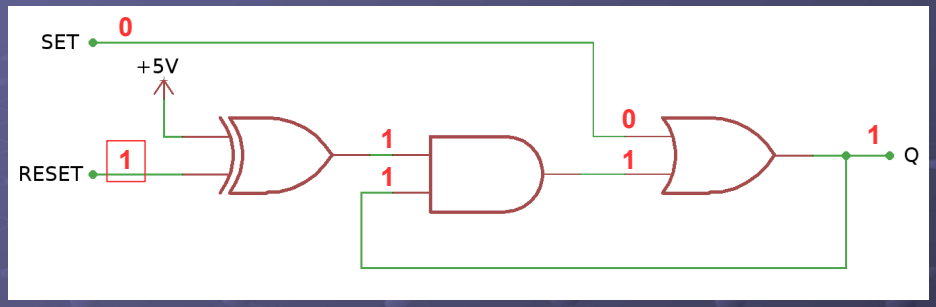
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



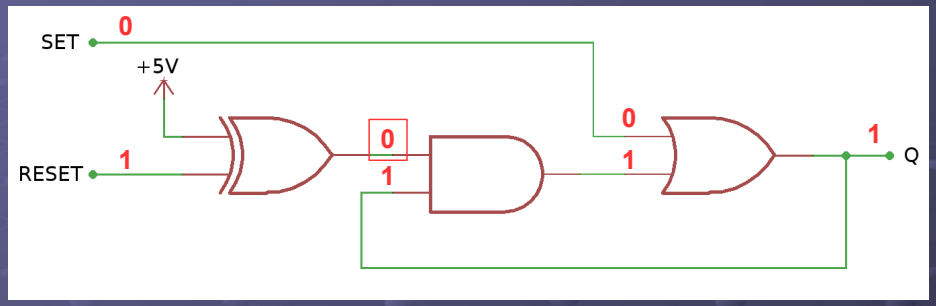
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



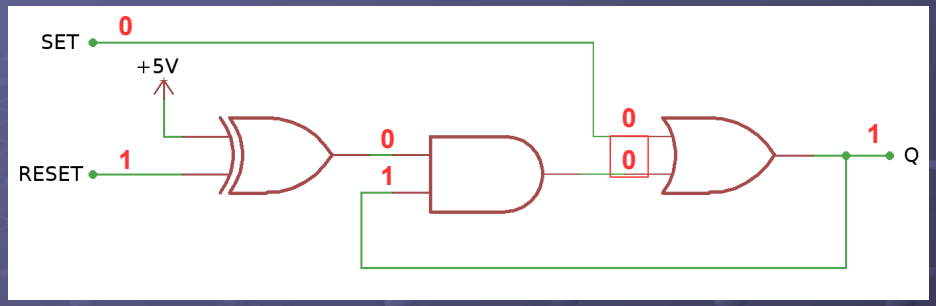
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



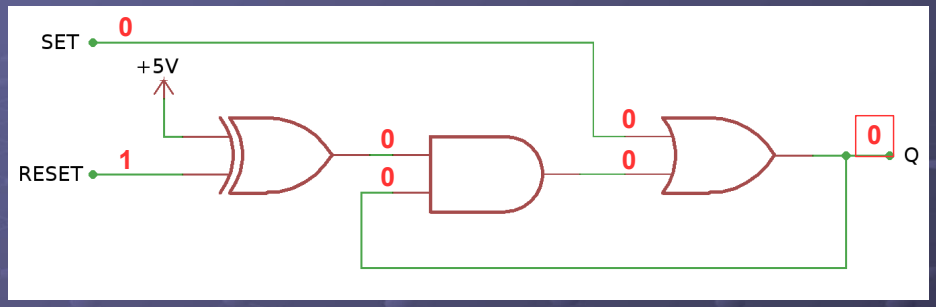
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



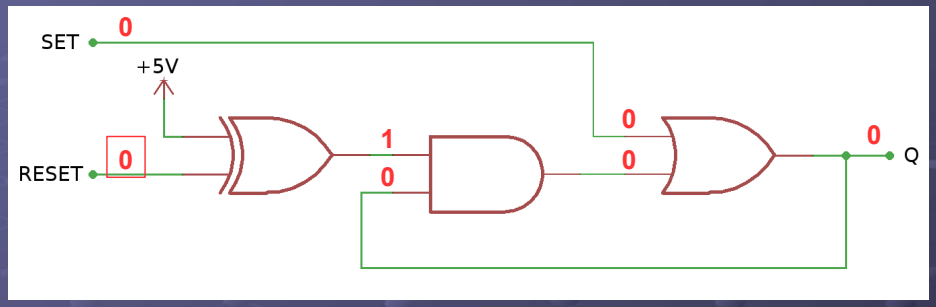
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



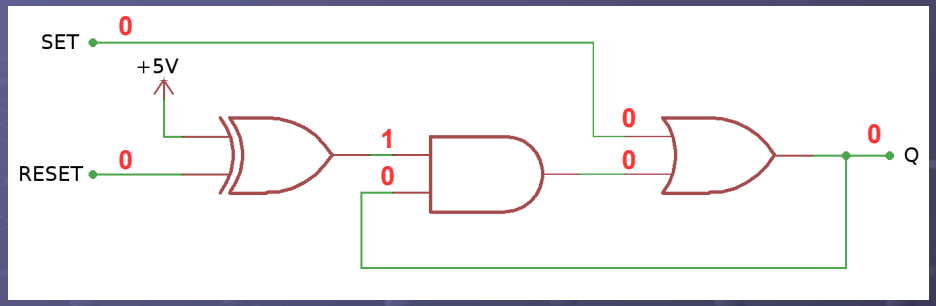
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



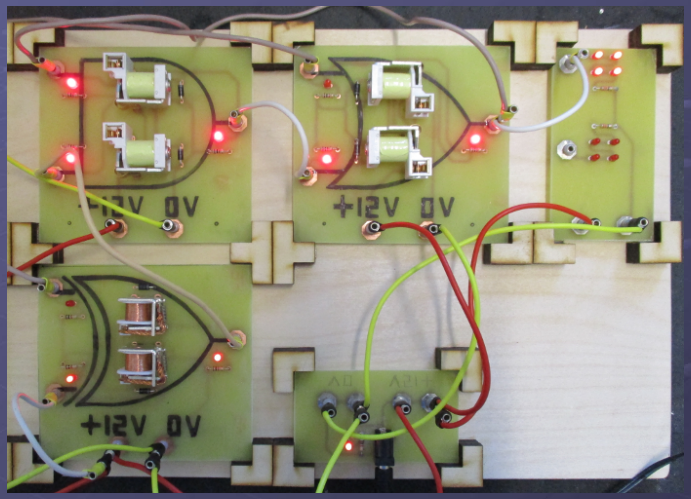
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Flip-Flop



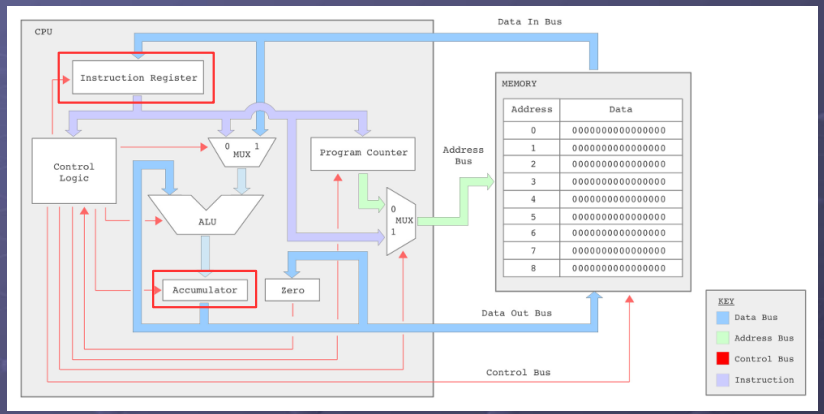
- Set-Reset (SR) Flip-Flop
  - Stores 1 bit of data on Q output, controlled by:
    - SET : active high, set output to a logic 1
    - RESET : active high, set output to a logic 0
  - Can be constructed using other logic circuits e.g. two NAND or two NOR gates (will do this in lab).
    - Q : What does the XOR gate implement?

Demo : relay logic



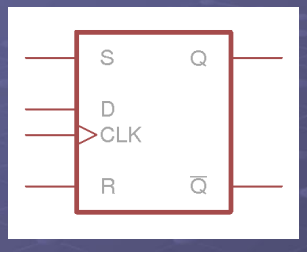
- Set Reset (SR) Flip-Flop

SimpleCPU_v1a



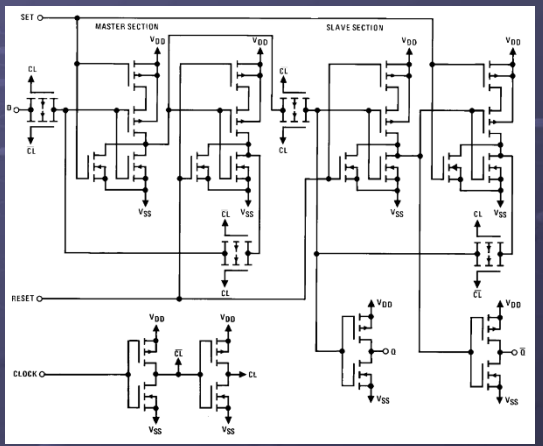
- Block diagram
  - Problem : to store data in a SR flip-flop you need to test the data you are storing i.e. you need to know if the SET or RESET input needs to be pulsed.

Flip-Flop



CL†	D	R	S	Q	$\overline{Q}$
⌐	0	0	0	0	1
⌐	1	0	0	1	0
⌐	x	0	0	Q	$\overline{Q}$
x	x	1	0	0	1
x	x	0	1	1	0
x	x	1	1	1	1

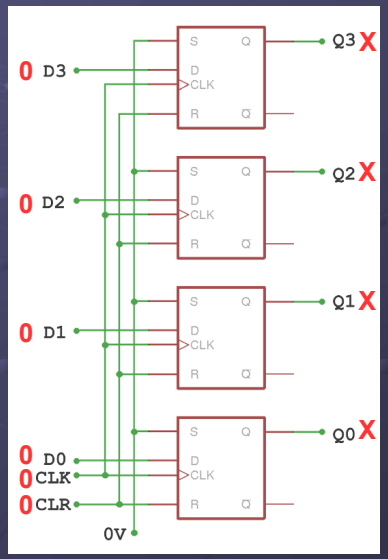
† = Level change x = Don't care case



- Data type Flip-Flop: DFF (4013 IC)
  - Need 32 transistors to store 1 bit (32T cell)
    - NOR gate (4001) only needed 8 transistors

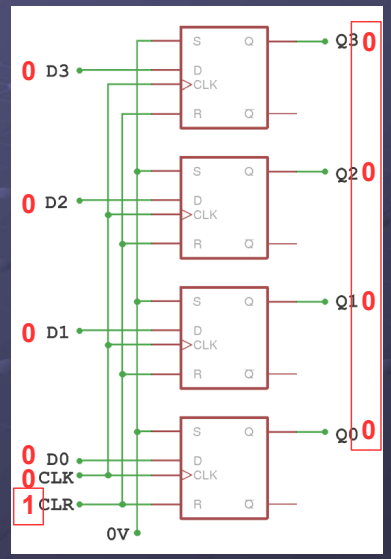
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.



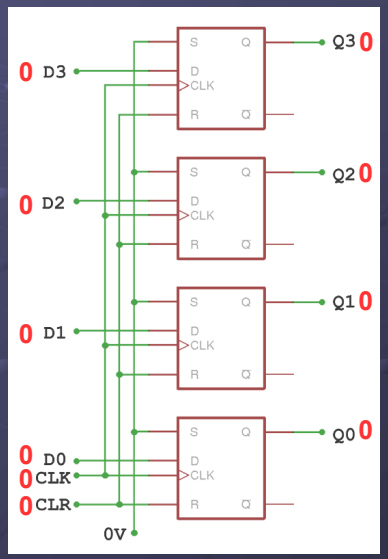
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.



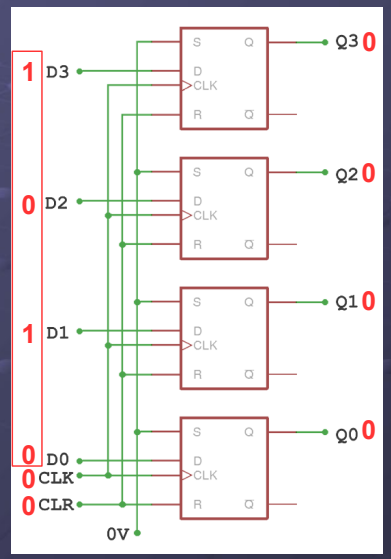
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.



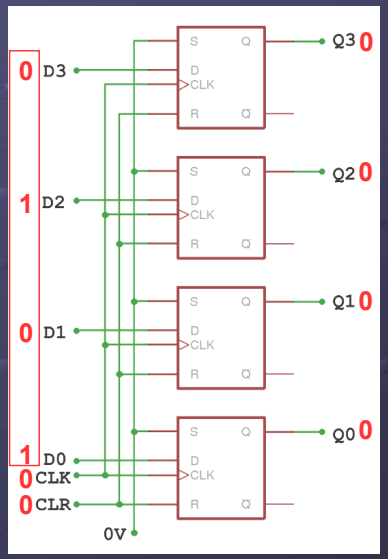
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.



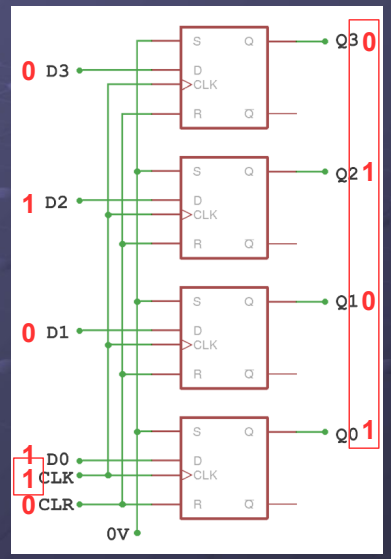
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.



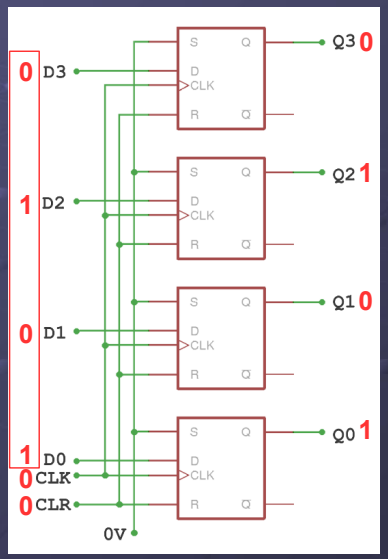
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.



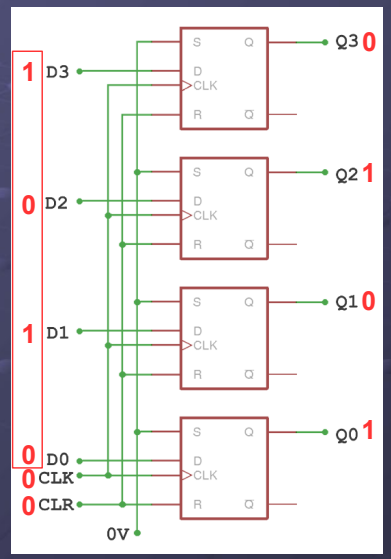
Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.

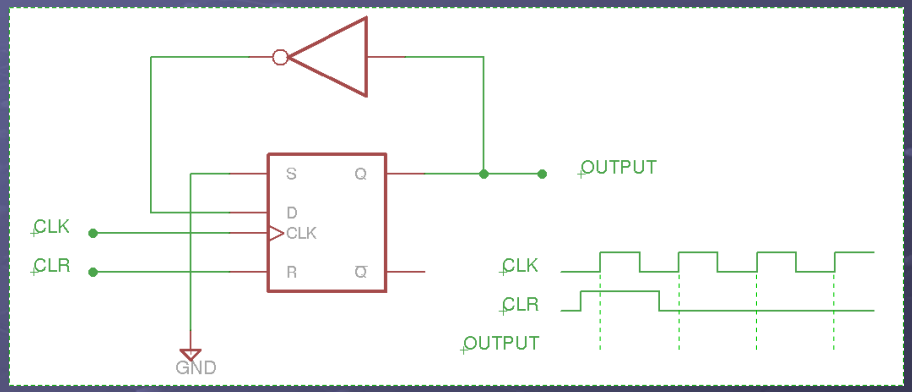


Register

- Registers
  - To increase the number of bits stored connect multiple D-type Flip-Flops (DFF) in parallel.
    - Share common inputs : RESET and CLOCK.
      - All DFF updated and cleared at the same time.
    - SET not used so connected to 0V.
    - Each DFF stores 1 bit, so four Flip-Flops can store a 4 bit value.
    - DFF commonly have another input not shown here: Clock Enable (CE) to disable updates.

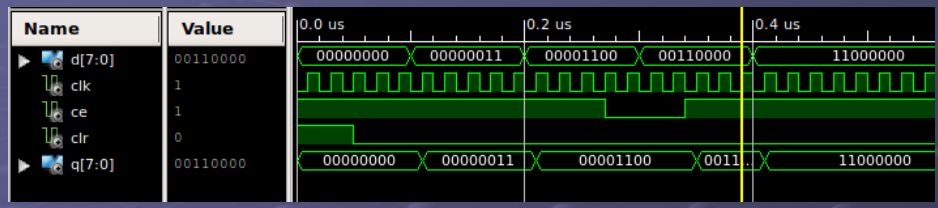


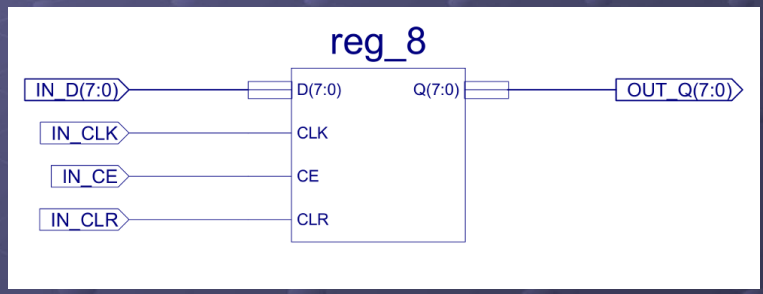
Flip-Flop



- Quick Quizzz
  - What does the output waveform of the flip-flop look like?

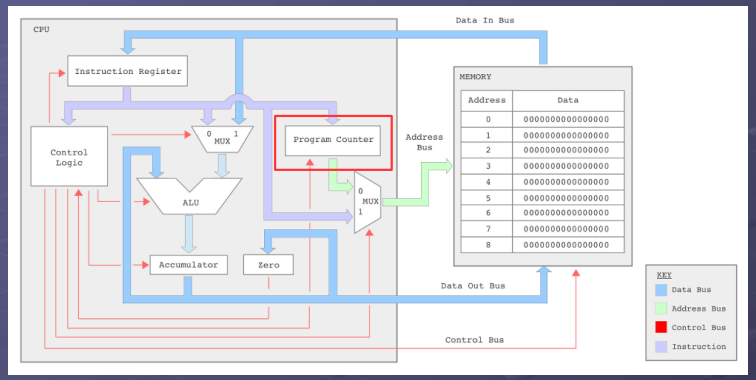
Example : REG_8.zip





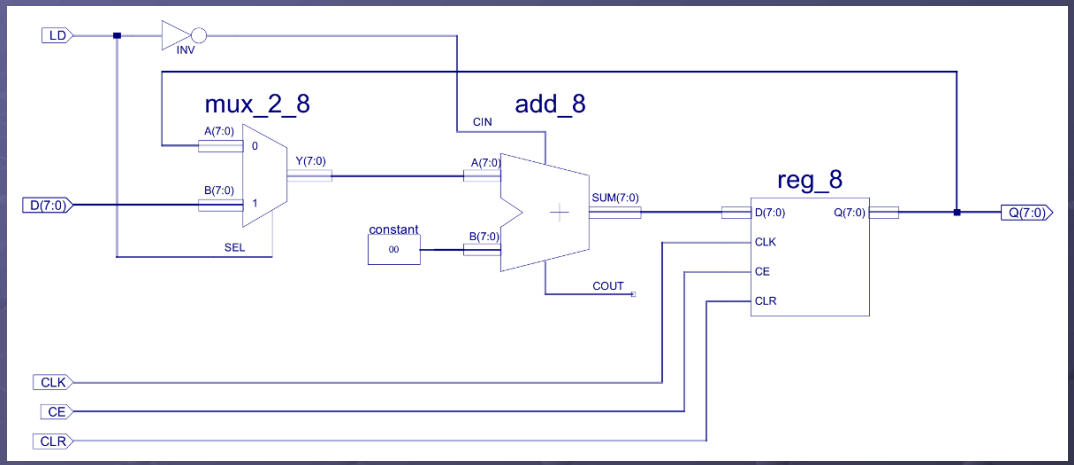
- 8 bit register with clock enable (CE) and clear (CLR)

SimpleCPU_v1a



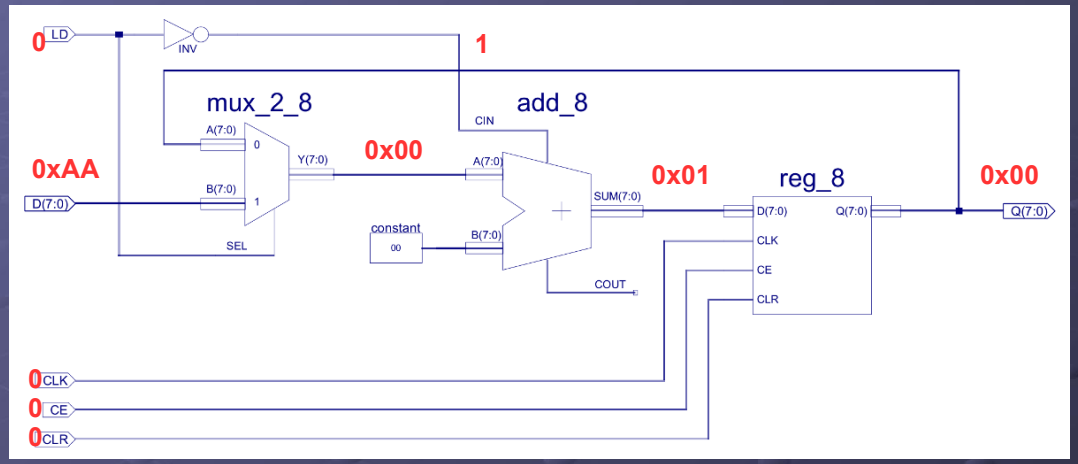
- Block diagram
  - Q: how do we build a counter?
    - Need to store the current count value
    - Increment this value to generate the next count
    - Load a different / starting value

Counter



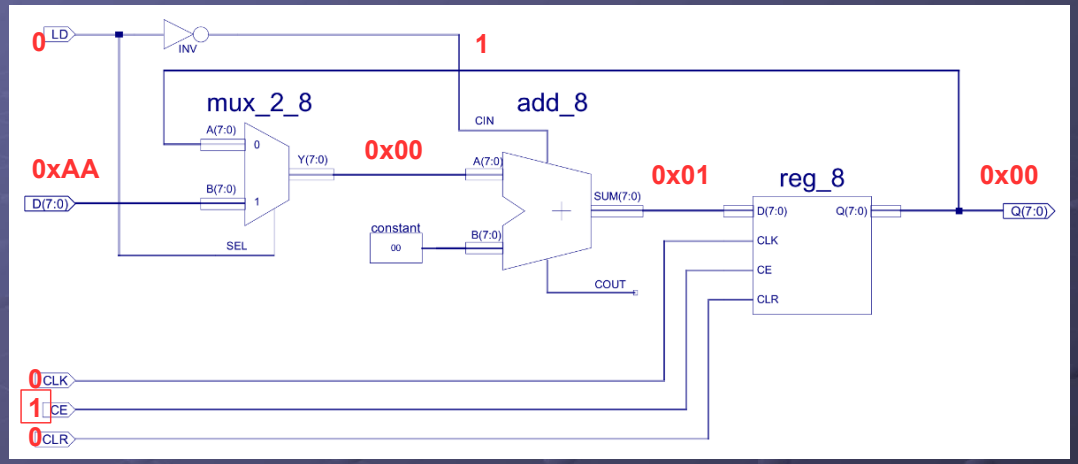
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



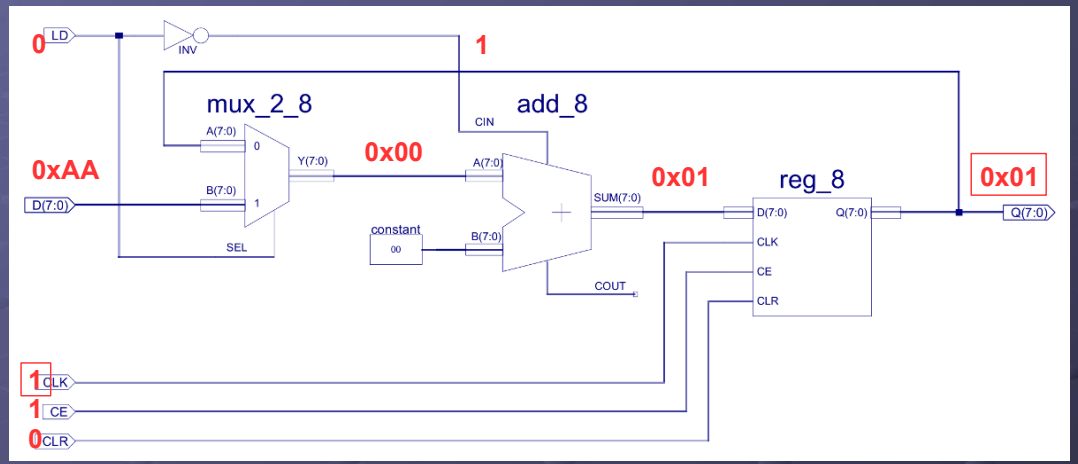
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



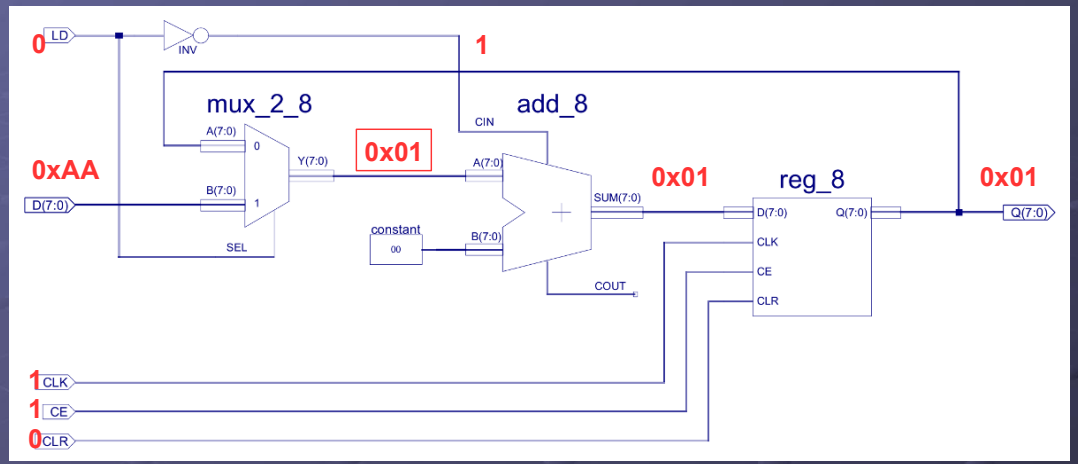
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



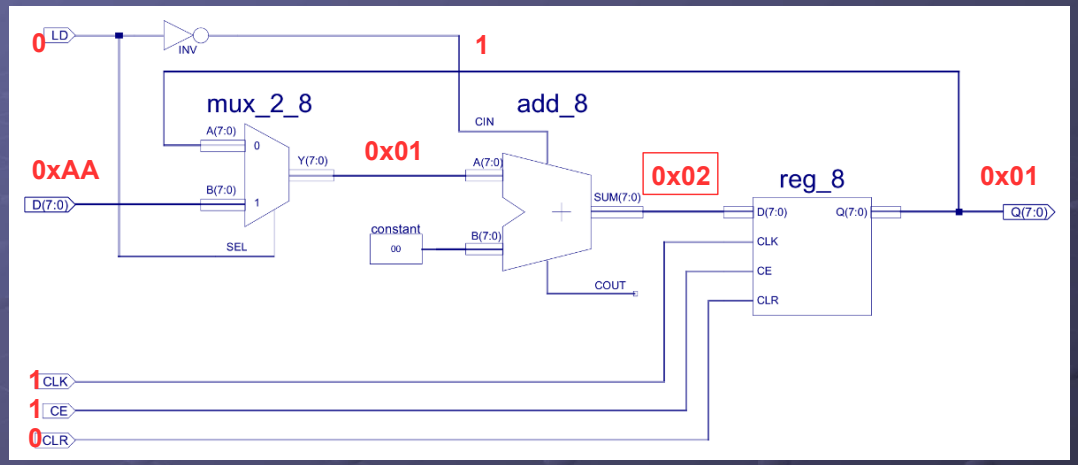
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



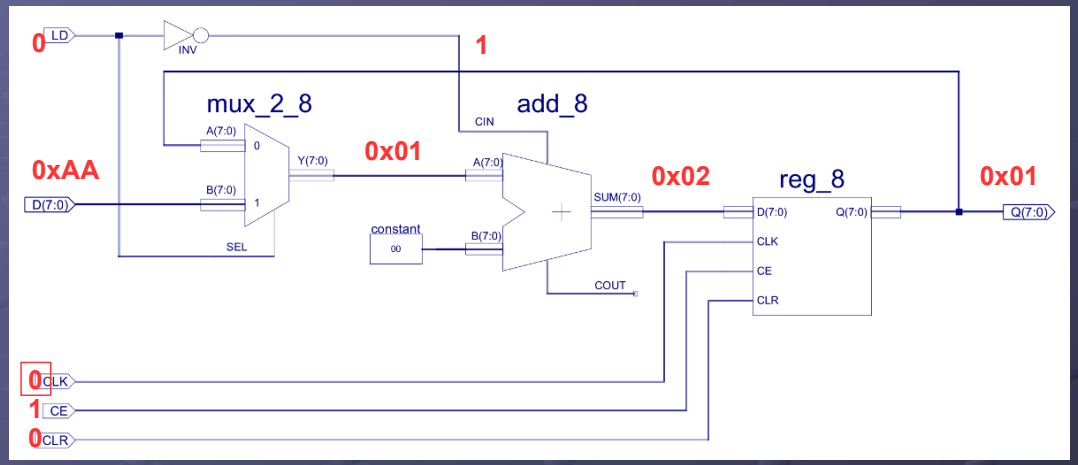
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



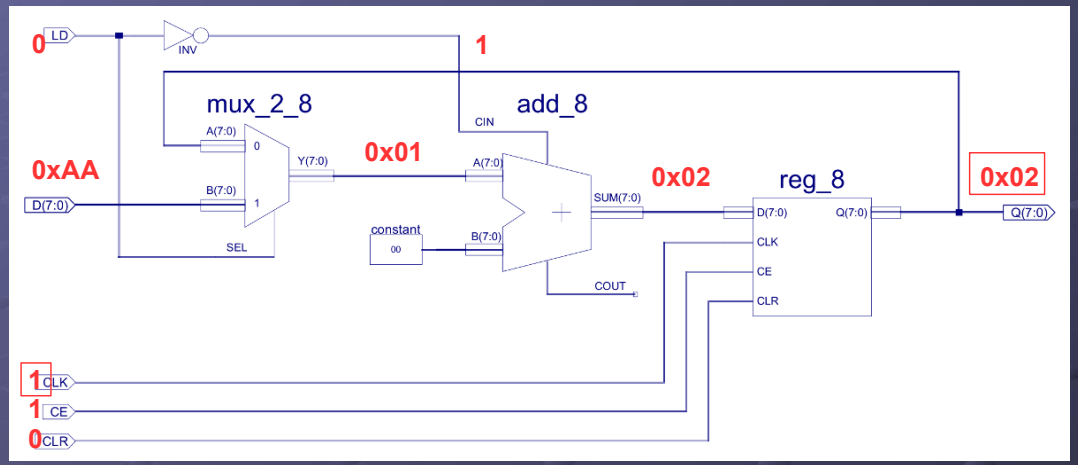
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



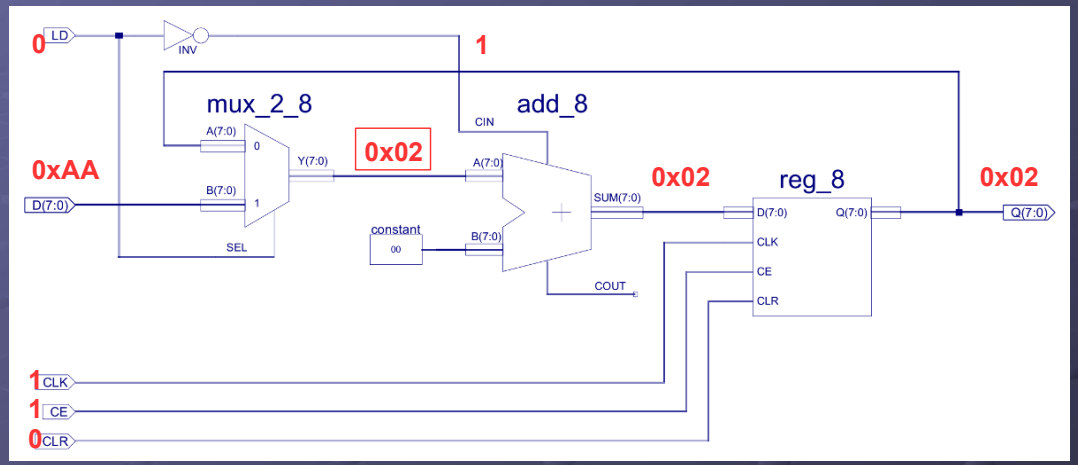
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



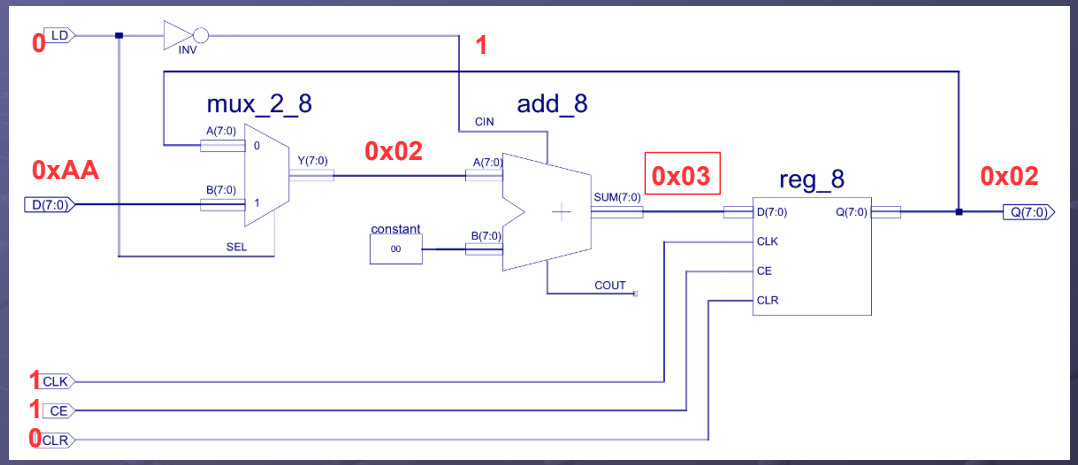
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



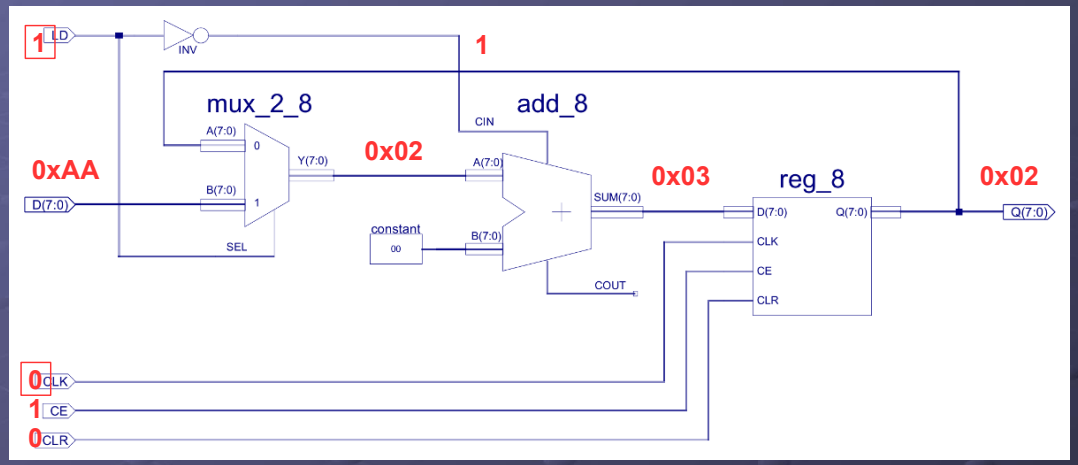
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



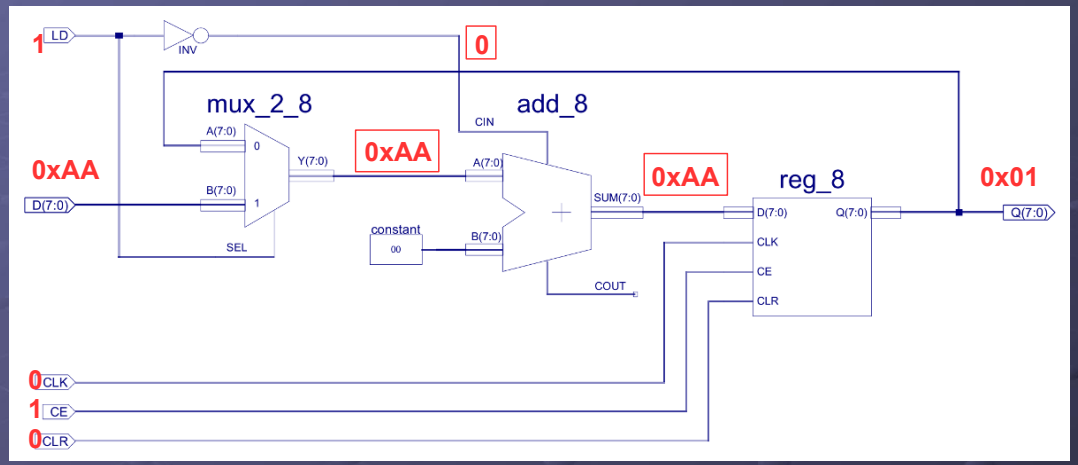
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



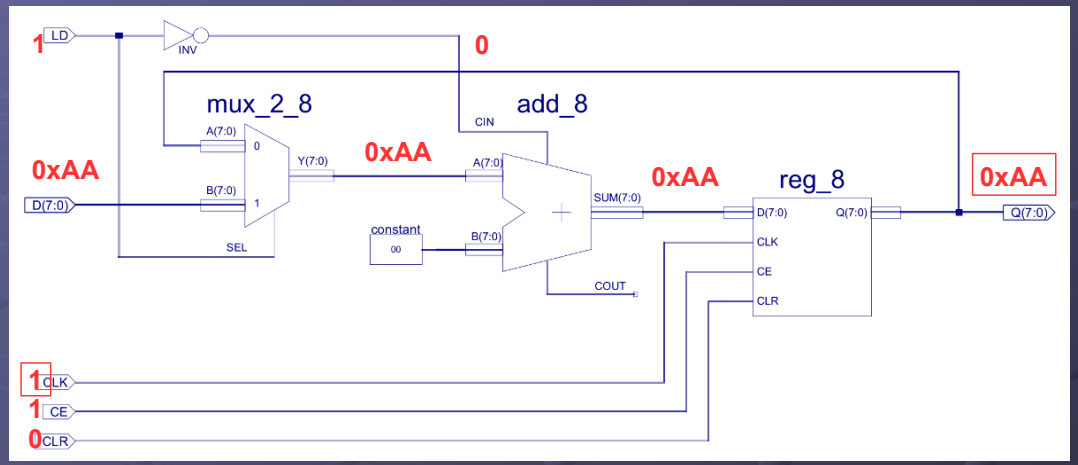
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



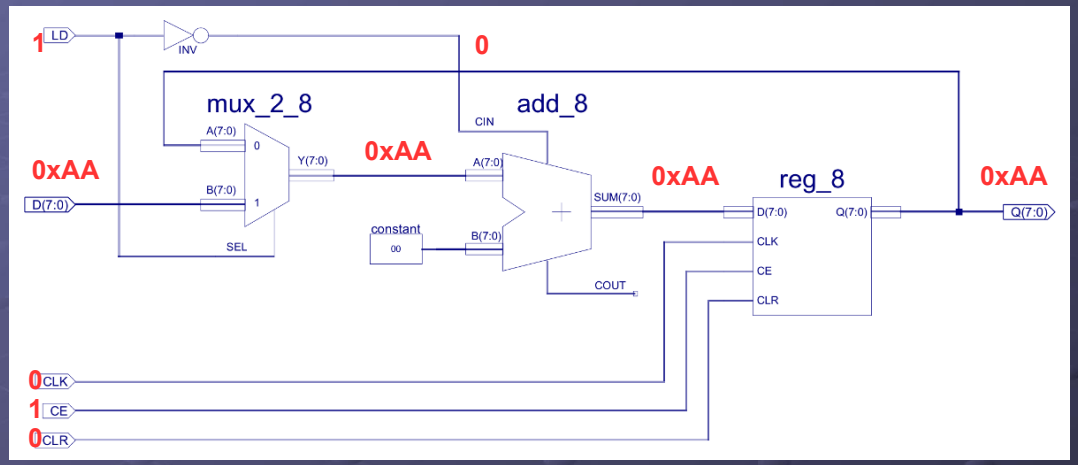
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



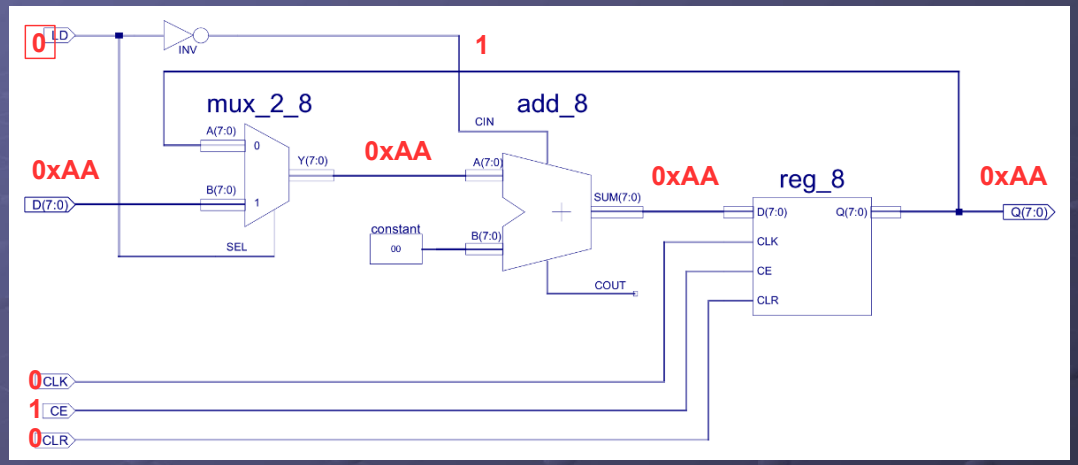
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



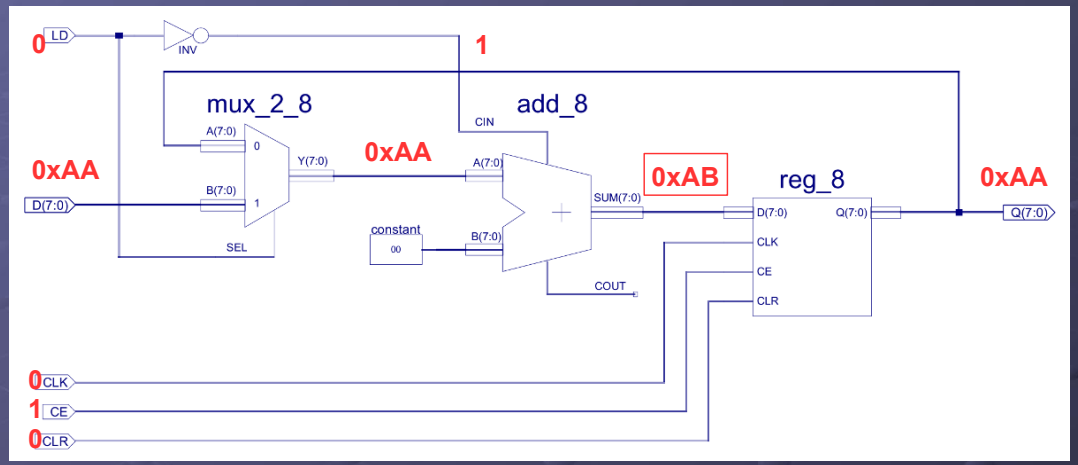
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

Counter



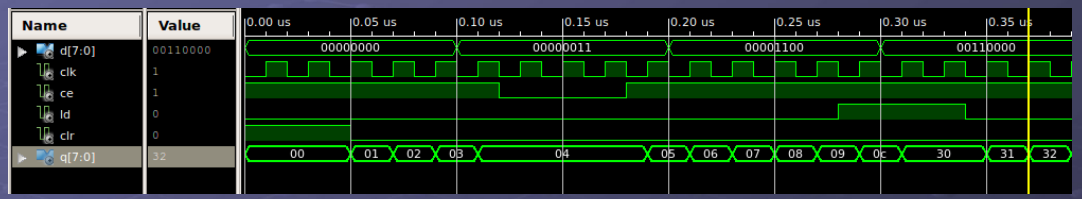
- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

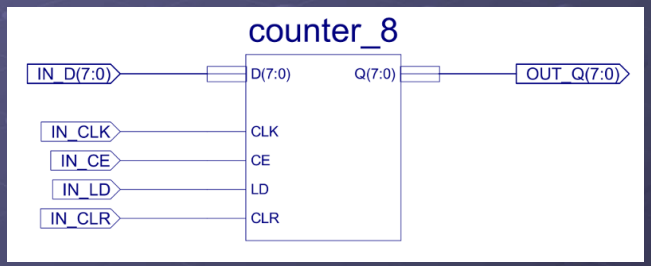
Counter



- Loadable binary counter
  - LD=0 increment count, LD=1 load new count value.

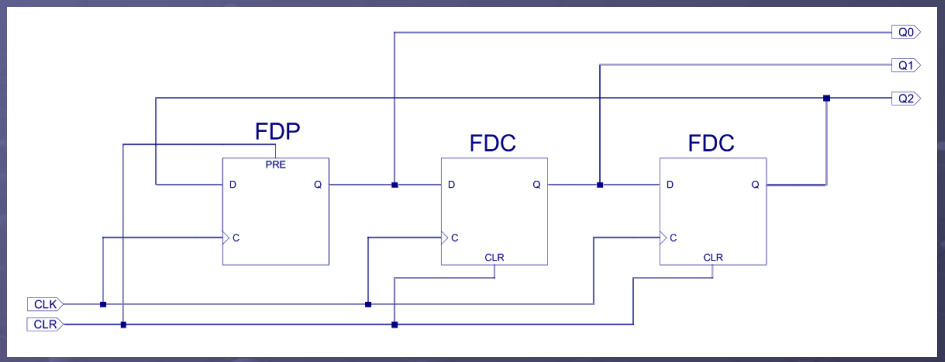
Example : COUNTER_8.zip





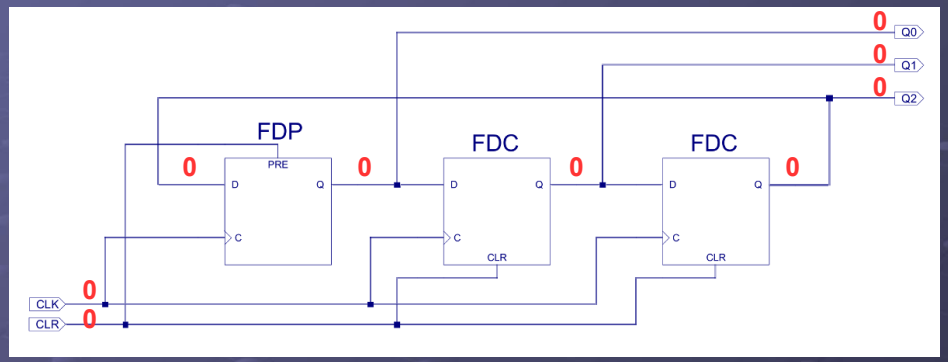
- Loadable binary counter
  - CE must be 1 to enable LD

Counter



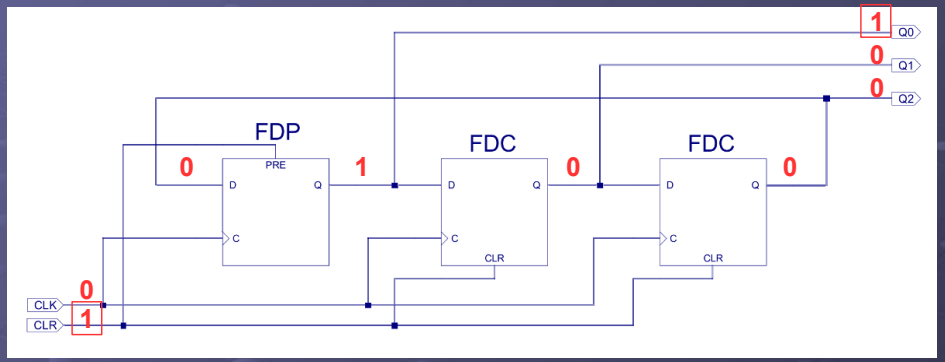
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



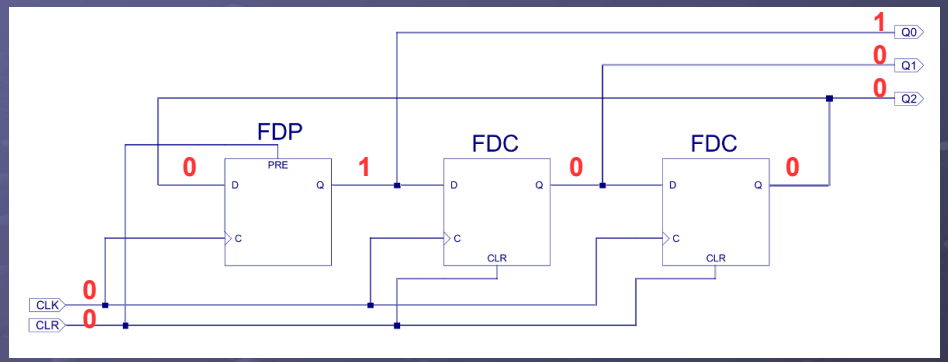
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



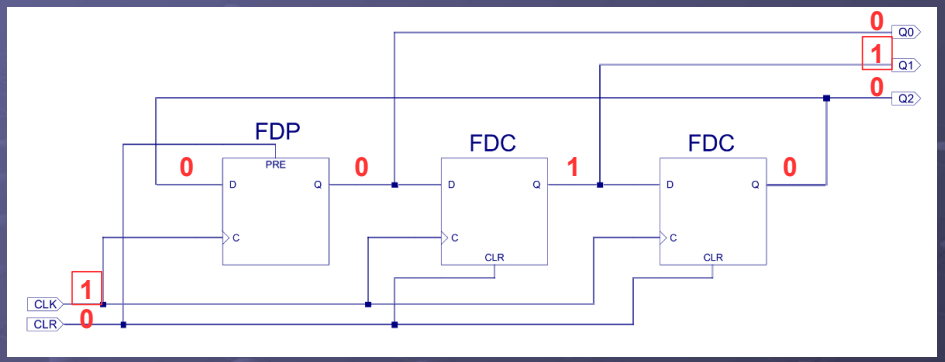
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



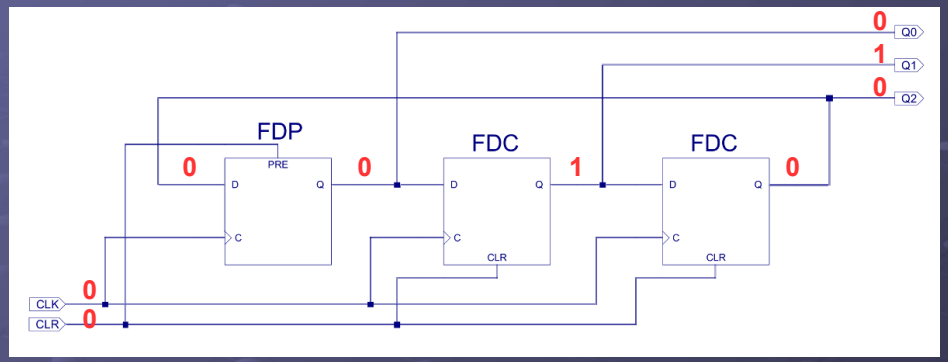
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



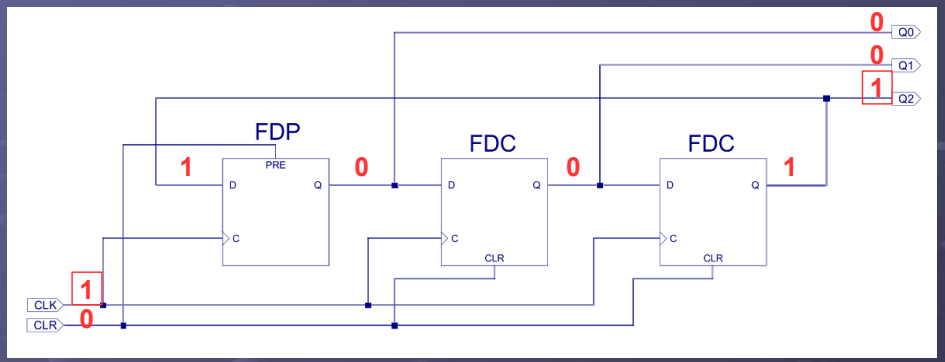
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



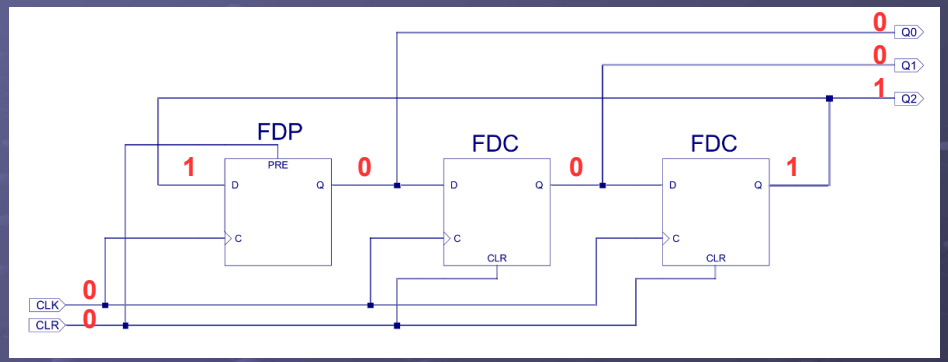
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



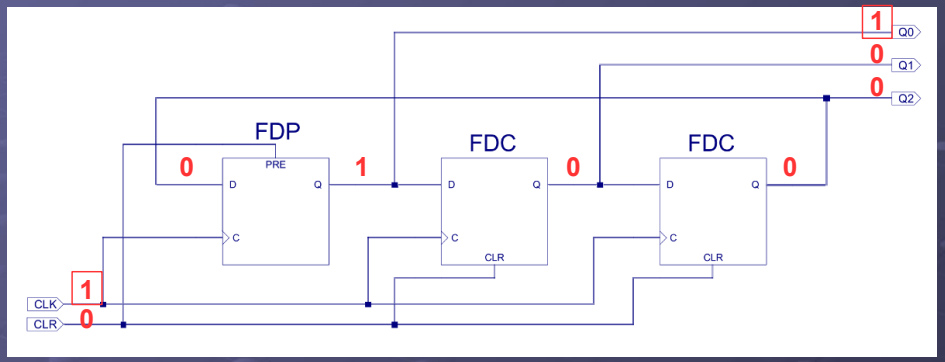
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



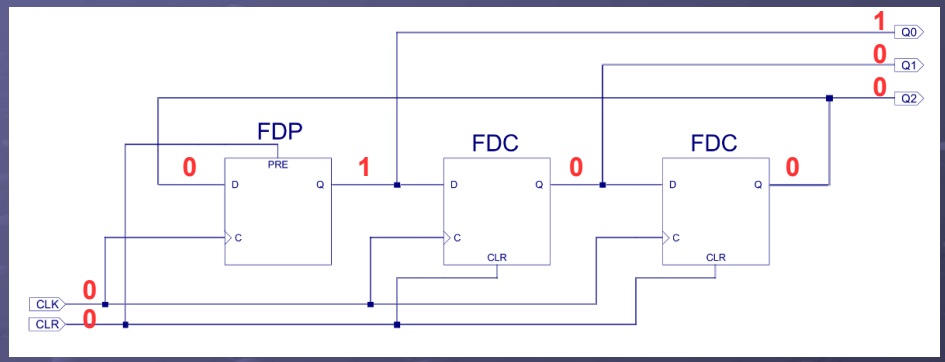
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

Counter



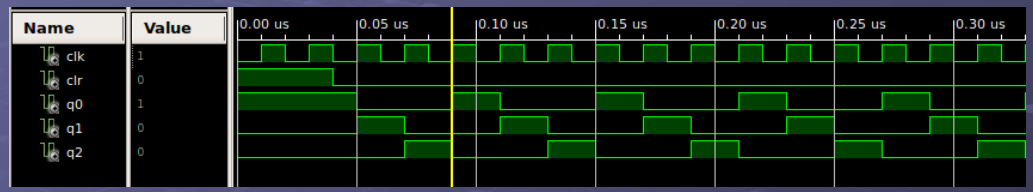
- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

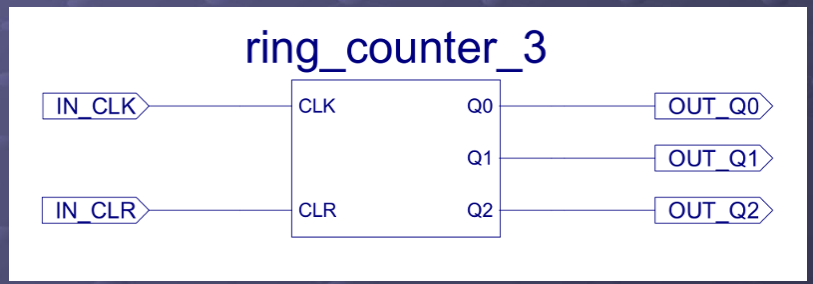
Counter



- Ring counter : a one-hot counter
  - First DFF uses PRE (SET) input to initialise first logic 1. Other DFF use CLR (RESET) to initialise all other DFFs to logic 0.
  - Each time the clock is updated the one logic 1 is rotated to the next position.

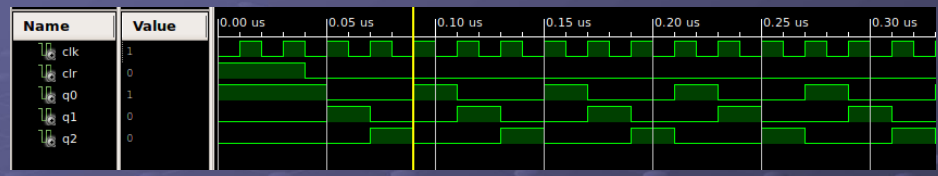
Example : Ring_Counter_3.zip

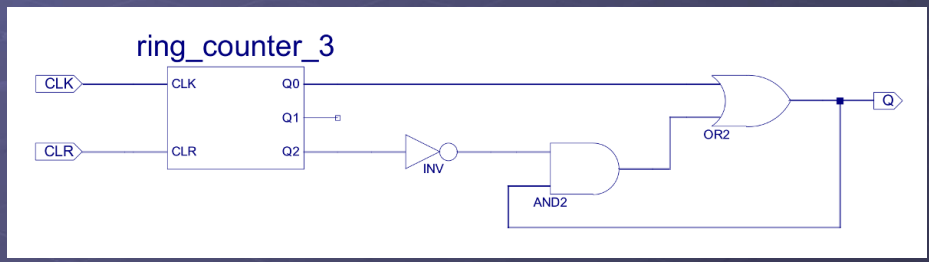




- Ring counter : a one-hot counter

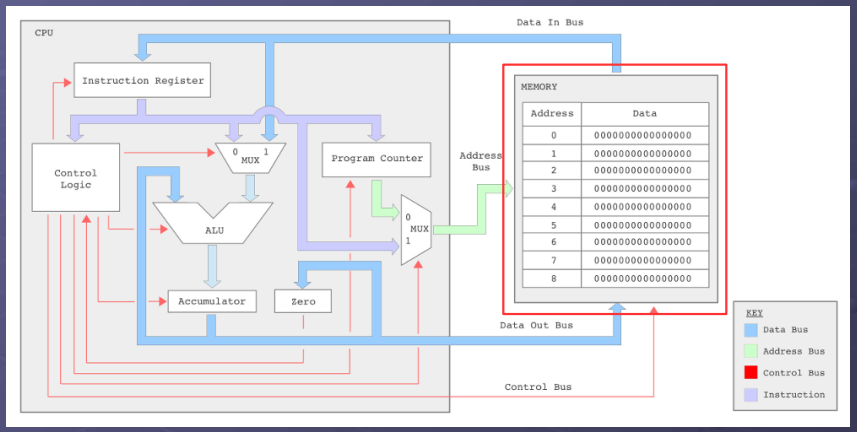
Counter





- Quick Quizzz
  - What does the output waveform Q look like?

SimpleCPU_v1a



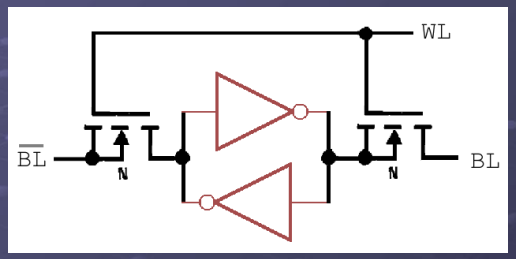
- Block diagram
  - Memory : to execute a program a computer needs to store that program (and data) in memory i.e. a stored program computer.

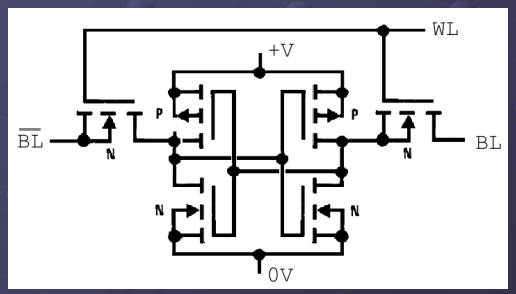
Memory

- Each memory location stores 16bits. If implemented using flip-flops this would require: 32T × 16 = 512T transistors per location.
  - That's a lot of silicon, less memory per unit area.
- To improve packing density we can use different memory cells i.e. sacrifice speed, increasing access times. This led to the development of :
  - Static Random Access Memory (SRAM)
    - Six transistor per bit (6T cell) = 96T
  - Dynamic Random Access Memory (DRAM)
    - Three transistors, 1 capacitor (3T1C cell) = 48T

Memory

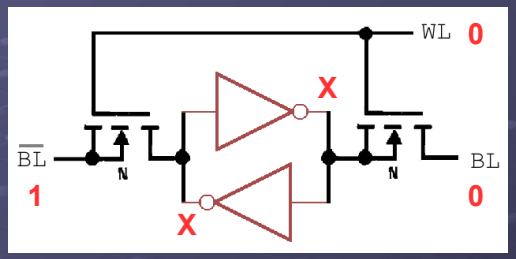
- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.

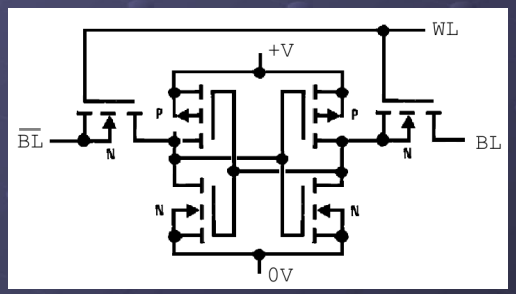




Memory

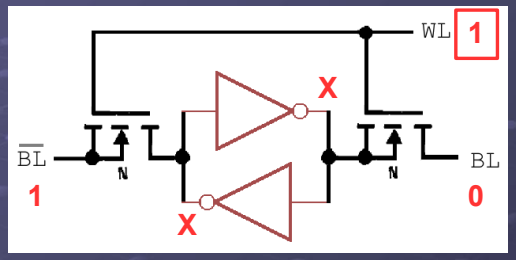
- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.

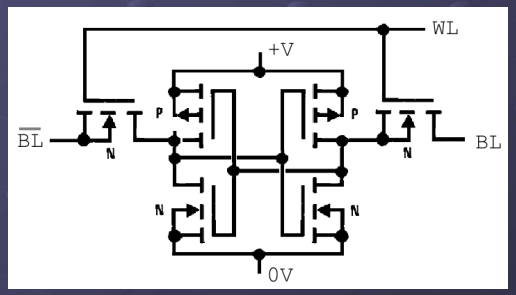




Memory

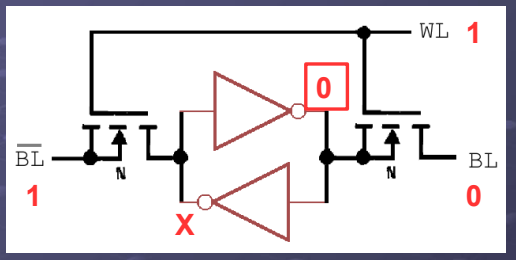
- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.

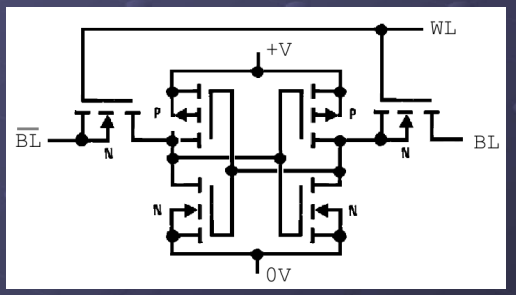




Memory

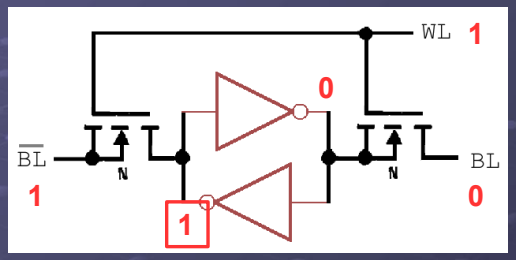
- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.

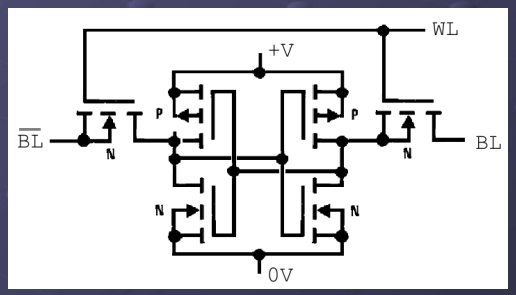




Memory

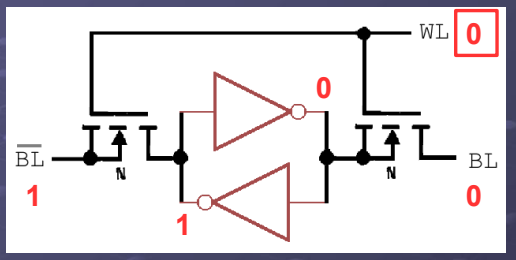
- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.

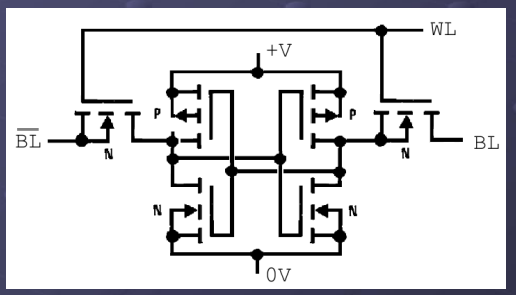




Memory

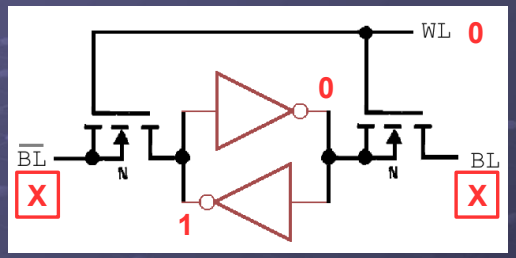
- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.

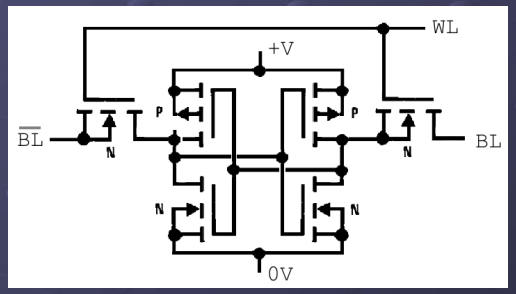




Memory

- Static Random Access Memory cell (SRAM)
- Six transistor cell
  - WL : world line, used to select cell to be read or written to.
  - BL : bit line, used to read data stored in cell, or new data driven onto this line during write.
  - $\overline{BL}$: not BL, inverted version of bit line.





Summary

- Key concepts :
  - Data is stored in the processor using registers
    - Implemented using SR Flip-Flop, D-type Flip-Flop …
    - 32 transistors (3 – 5 logic gates) to store each bit.
  - Registers
    - Parallel array of DFFs, share common inputs : RESET and CLOCK, updated and cleared at the same time.
  - Counters
    - Using different encodings : Binary, Onehot …
    - Constructed from: registers, multiplexers and adders ...
  - Memory
    - SRAM and DRAM memory cells.
    - 3 to 6 transistors to store each bit.