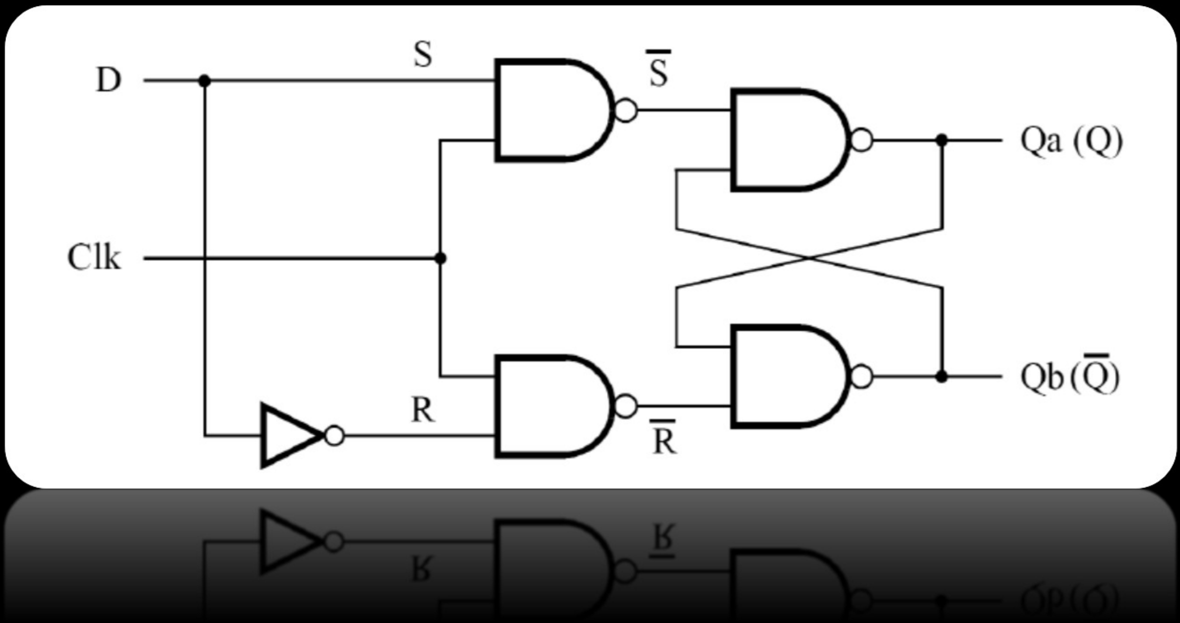
Lab 4 Preparation

What Lab 4 is about

- Part I (1 mark):
  - Implementing latches and flip-flops with IC chips
  - Last thing you'll do on the breadboards!
- Part II (1 mark):
  - Using a register to store & provide ALU values
    - Register stores the output of the ALU
    - Same register also provides an input to the ALU
- Part III (1 mark):
  - Implement a shift register.

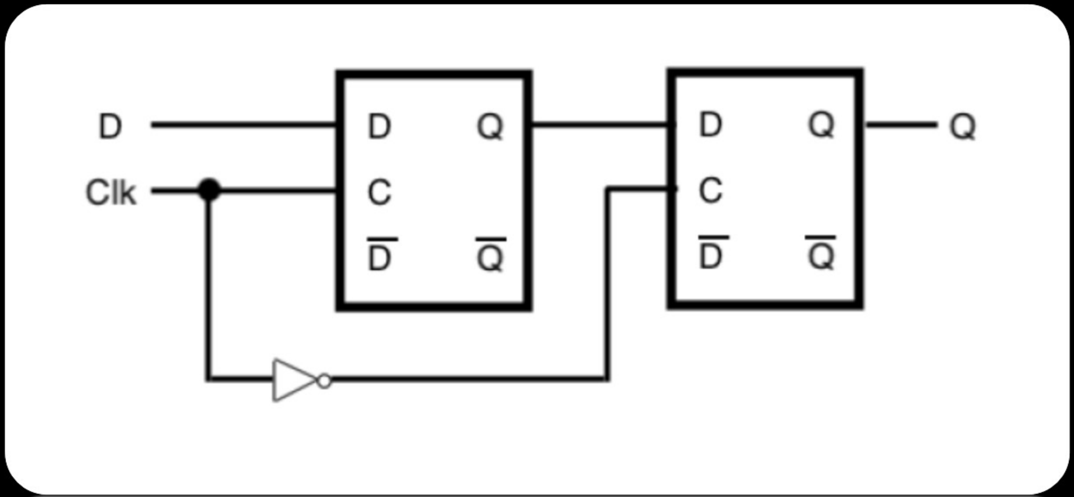
Lab 4 - Part I

- Create a D latch on the breadboard!
  - Using NAND gates!
- Same procedure as for Lab 1:
  - Use Logisim to design a circuit with IC chips
  - Test your circuit using the Poke tool.



Lab 4 – Part I

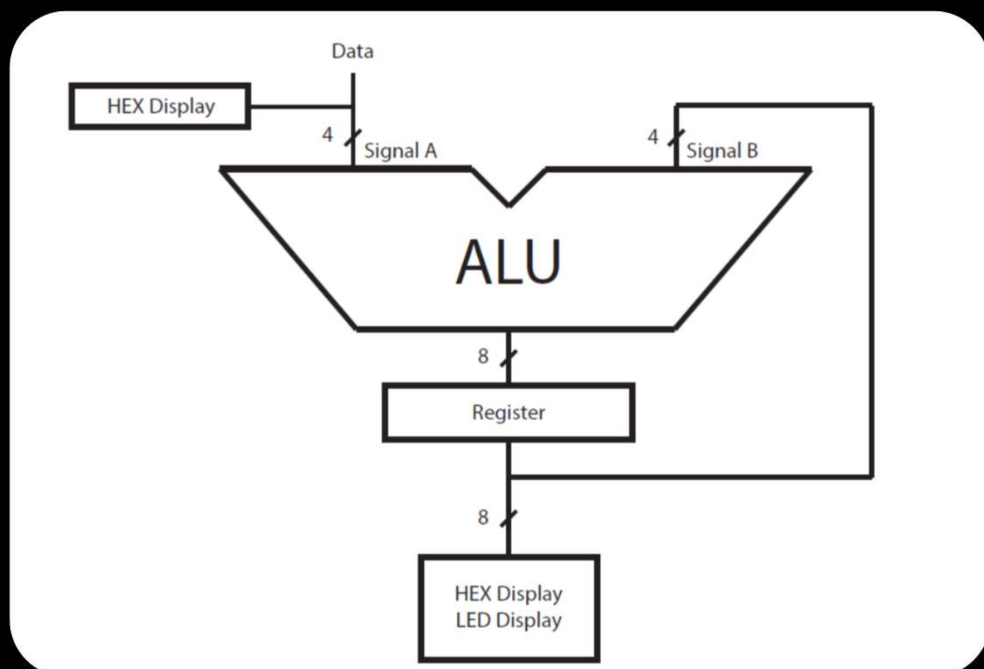
- …then, create a D flip-flop out of D latches ☺



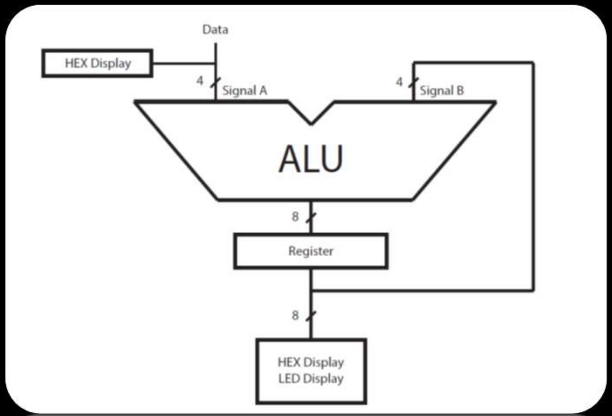
- How are you going to test this?
    - Logisim test vectors have difficulty here ☺

Lab 4 – Part II

- Enhance the ALU from last week:
  - More operations (e.g. multiplication, shifting)
  - Store the result in a register.



Lab 4 – Part II



- Old operations:
  - 0: A+1 (using your adder)
  - 1: A+B (using your adder)
  - 2: A+B (using the Logisim adder)
  - 3: A xor B in lower four bits, A or B in upper four bits
  - 4: Reduction OR operation on A & B
- New operations:
  - 5: Left shift B by A bits
  - 6: Logical right shift B by A bits
  - 7: AxB (multiplication)

All of these are supplied through the collection of components in Logisim.

What does it mean to shift?

- Suppose that B = 00010110 and
  A = 00000011
- "Left shift B by A bits" = shift every bit in B three bits to the left (since A = 3).
  - One bit to the left: B → $\boxed{0010110}$0
  - Two bits to the left: B → $\boxed{010110}$00
  - Three bits to the left: B → $\boxed{10110}$000
- One bit gets shifted off the left side each time, and a zero is shifted into the right.

Logic vs Arithmetic Shift

- Function 6 of the ALU is a logical right shift.
  - This is the same as the left shift, but to the right.
- What does the word "logical" signify?
  - `B` is storing binary values, but not a number.
  - When shifting right, shift in zeroes on the left side.
- What if `B` is storing a binary number?
  - Then you perform an arithmetic right shift.

Logic vs Arithmetic Shift

- Arithmetic right shifts replicate the sign bit instead of using zero to fill in the most-significant bit(s).
  - Used when shifting signed numbers.
  - For unsigned numbers, use logical shift (like in Part 2).

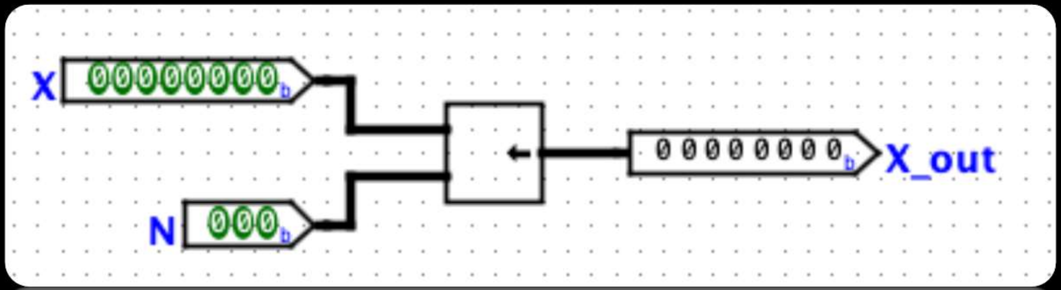
- Example: Shift `10010000` right by 3 bits
  - Arithmetic → **111**`10010`
  - Logical → **000**`10010`

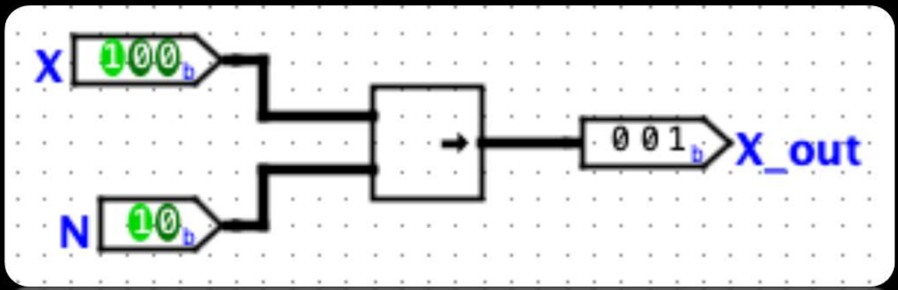
Logic vs Arithmetic Shift

- Why do we shift in the sign bit?
- Shifting B left A bits multiplies B by 2^A.
  - Again, assuming that B = 00010110 (22_{10})
  - If A=1, B→ 00101100 (44_{10})
  - If A=2, B→ 01011000 (88_{10})
- Shifting B right A bits divides B by 2^A, but only if you preserve the sign bit!
  - Suppose that B = 11110110 (-10_{10})
  - Arithmetic shift right: B→ 11111011 (-5_{10})
  - Logical shift right: B→ 01111011 (123_{10})

Logisim Shifter Components

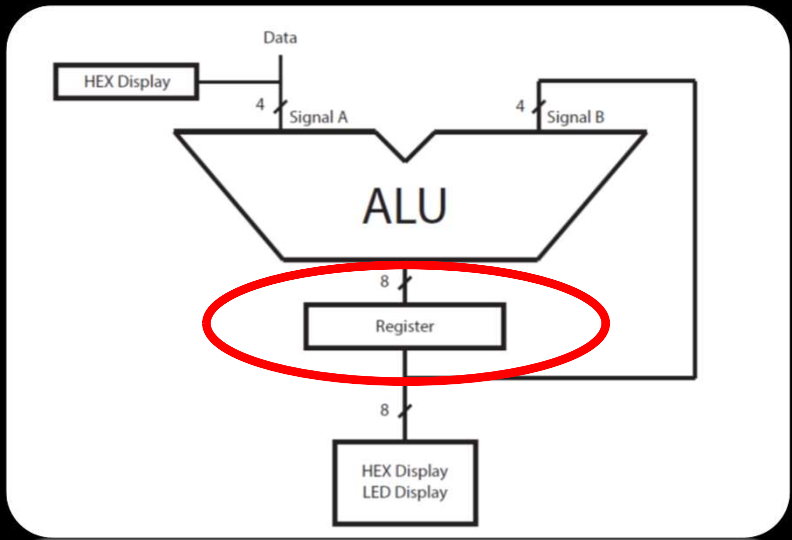
- Getting back to Lab 4, part II…
- Shifter components can be found in Logisim under Arithmetic > Shifter.
  - You can change the shift type in Properties.
  - More details can be found at:
    - http://www.cburch.com/logisim/docs/2.3.0/libs/arith/shifter.html
- Illustrated below: Logical Shift (left and right)





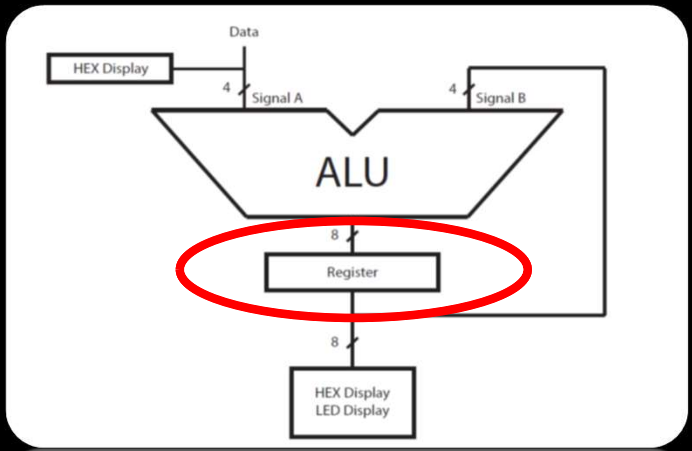
Lab 4 – Part II

- The other major addition to the ALU is the register that stores the output value.
  - 8 bits (flip-flops) long.
  - Compoment found in Memory > Register.
  - You can change the number of flip-flops in Properties > Data bits.



Why is the register here?

- The ALU is a transparent device, meaning that input values are passed straight through to the output (with some small delay).
  - Since the output of the ALU feeds back to input B, without the register the ALU output would change constantly.
  - The register has a clock signal that only updates its contents when the clock changes from `0` to `1`.



Final notes about Part II

- Make sure that for addition operations, the output preserves the carry bit.
  - For example, if `A=1101` and `B=1011`, the ALU output should be `00011000`.
  - If you didn't do this for Lab 3, make sure to do it for Lab 4 ☺
- The input B to the ALU is the least significant bits of the ALU output.
- Remember to display the inputs and outputs on seven-segment displays (and outputs on LEDR)

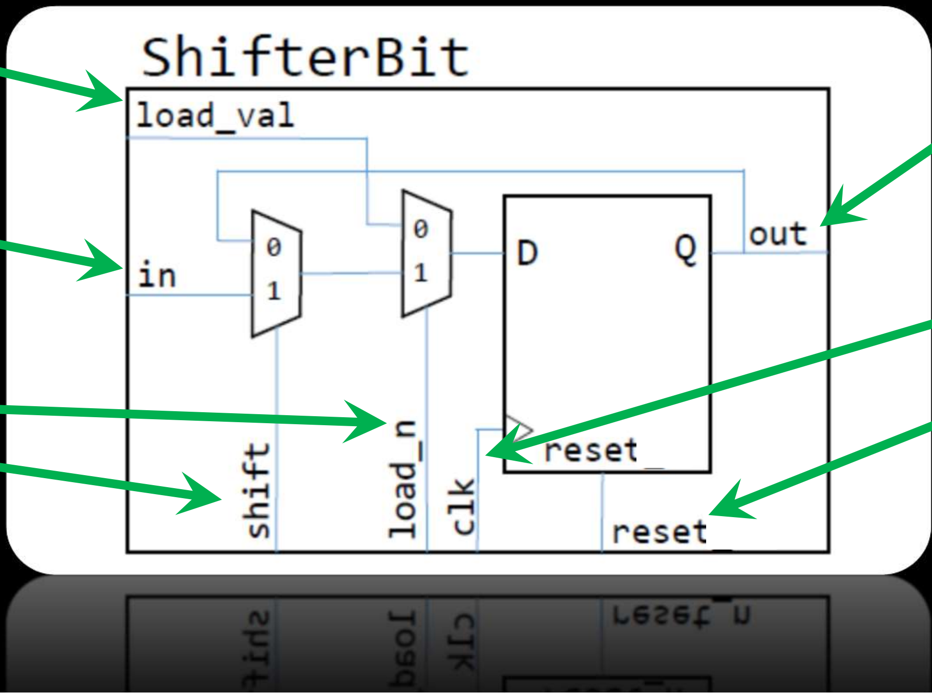
Lab 4 – Part III

- Make an 8-bit shift register out of 8 individual shifter units (each unit stores a single bit)
  - Similar to how the ripple carry adder was created.
- Below: the diagram for a single shifter unit.

load_val: external value to load into register.

in: input value coming from next shifter unit

shift_load_n: are we doing a shift or load operation?



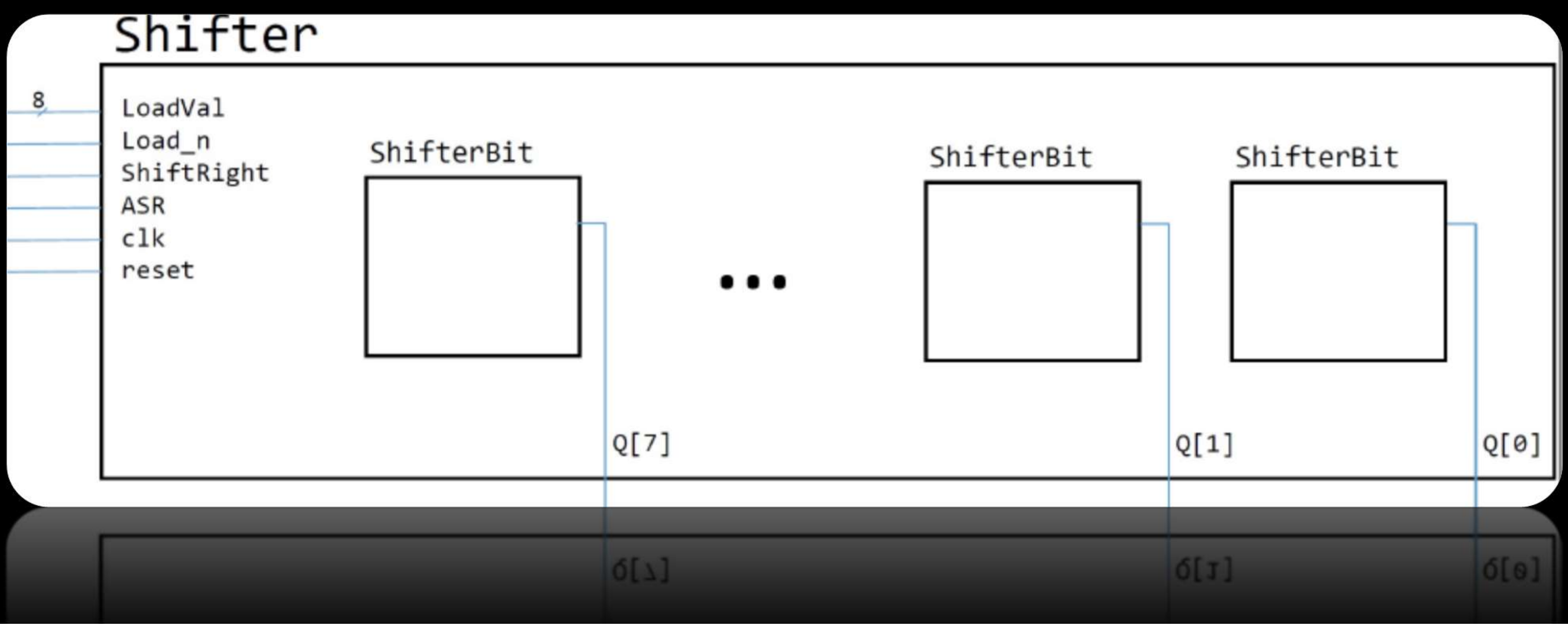
out: outputs the contents of this shifter unit.

clk: external clock signal for register.

reset: clear unit contents to zero.

Lab 4 – Part III

- Once you've made a module for the shifter unit, connect 8 shifter units together to make the shift register.
- <u>Note:</u>
  - This shift register only shifts to the right.
  - The shift register can load a new value from the 8 input bits in `LoadVal` (only happens when `Load_n` is ***low***)



Lab 4 – Part III

- Shifter signals:
  - Load_n signals the ShifterBit units to load a new value from the LoadVal input (that you provide)
  - clk (the clock signal) signals when the ShifterBit contents should change (both shifting and loading operations).
  - reset is asynchronous (independent of the clk signal).
  - ShiftRight tells you to shift, ASR tells you what kind of shift to do (0=logical, 1=arithmetic).