Code and Circuits

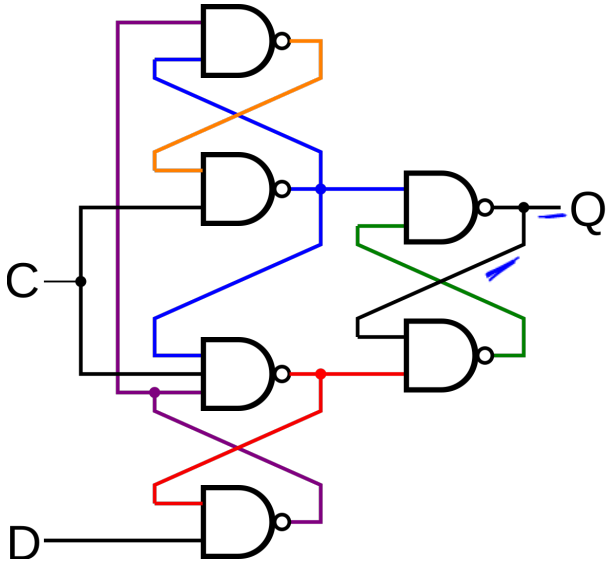
CS 2130: Computer Systems and Organization 1

September 14, 2022

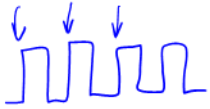
Announcements

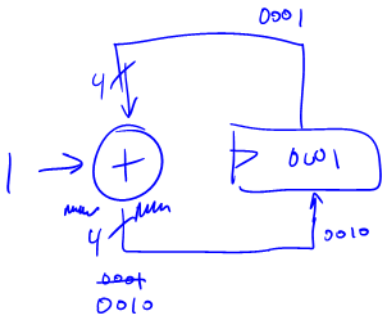
- Homework 1 due tonight!
- Homework 2 available (due on Gradescope, 11pm Monday)

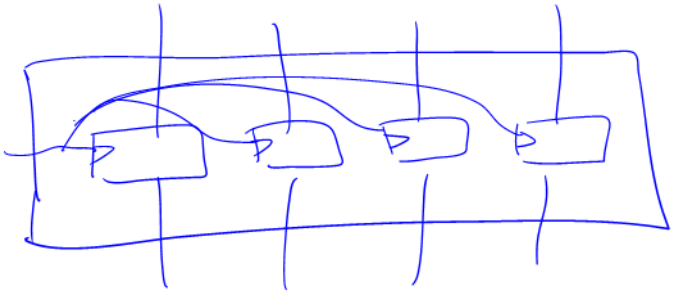
1-bit Register Circuit



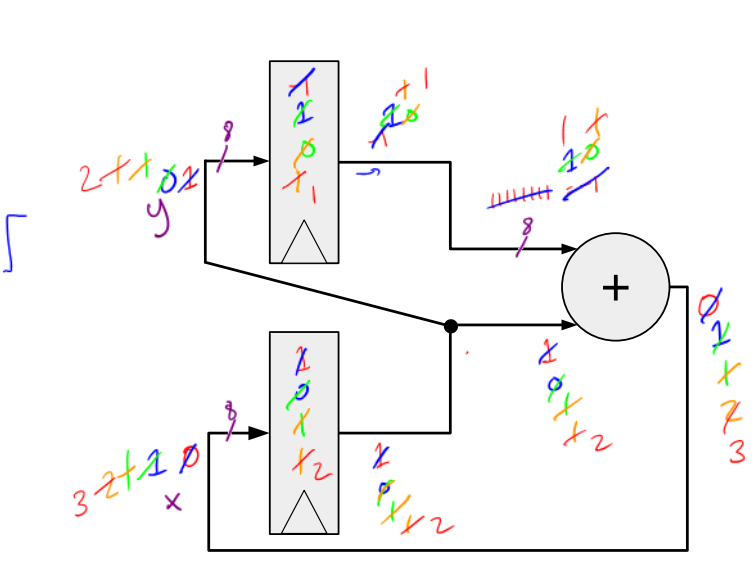
Building a Counter





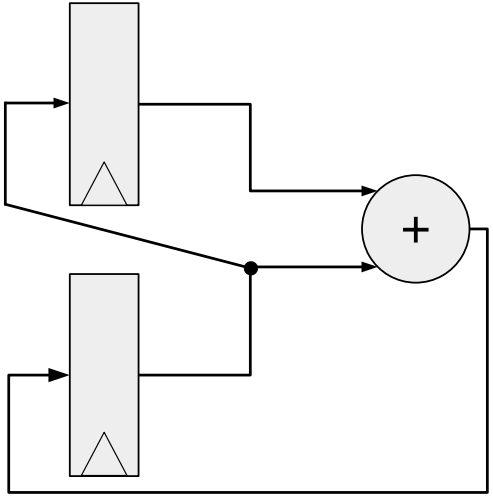


Another Circuit

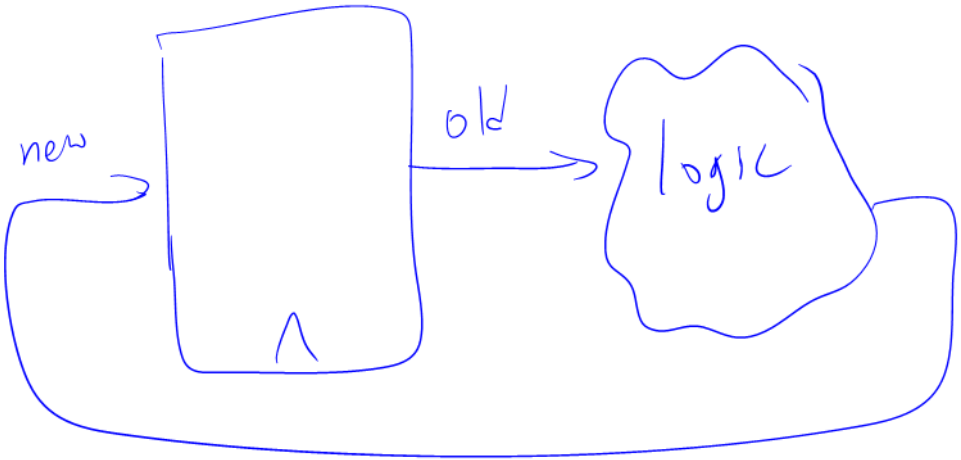


x	y	tick
0	1	0
1	0	1
1	1	2
2	1	3
3	2	4
5	3	5
8	5	6
13	8	7

Another Circuit



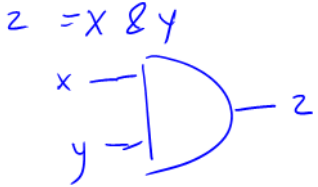
Common Model in Computers

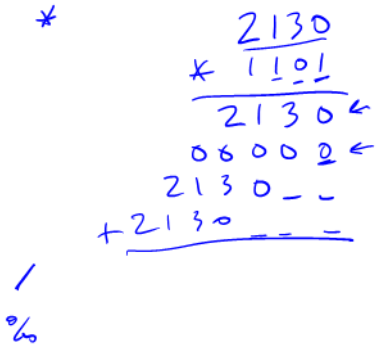


Code to Build Circuits from Gates

Write code to build circuits from gates

- Gates we *already* know: &, |, ^, ~
- Operations we can build from gates: +, -
- Others we can build:





bit shift gate ?

Code to Build Circuits from Gates

Write code to build circuits from gates

- Gates we *already* know: &, |, ^, ~
- Operations we can build from gates: +, -
- Others we can build:
- Ternary operator: ? :

z = a ? x : y ≡ if(a) z=x else z=y

z = (a==b ? 32 : x) * y

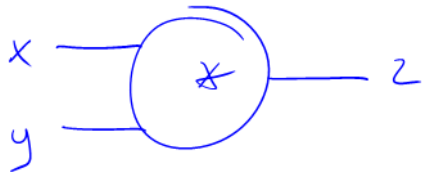
2nd ,or

Equals

Equals: =

z = x * y;

- Attach with a wire (i.e., connect things)
- Ex: `z = x * y`

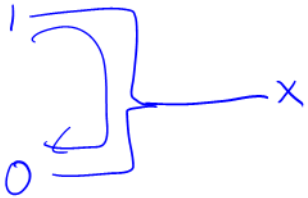


Equals

Equals: `=`

- Attach with a wire (i.e., connect things)
- Ex: `z = x * y`
- What about the following?

```
x = 1
x = 0
```



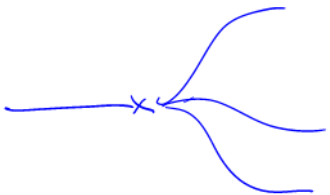
Equals

Equals: `=`

- Attach with a wire (i.e., connect things)
- Ex: `z = x * y`
- What about the following?

  ```
  x = 1
  x = 0
  ```

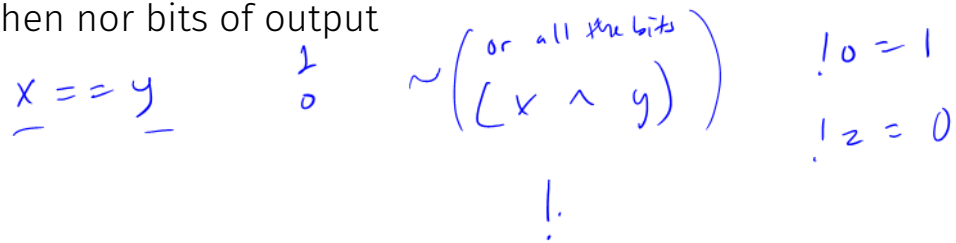
- **Single assignment**: each variable can only be assigned a value once



Comparisons

Each of our comparisons in code are straightforward to build:

- == - xor then nor bits of output



Comparisons

Each of our comparisons in code are straightforward to build:

- `==` - xor then nor bits of output
- `!=` - same as `==` without not of output

Comparisons

Each of our comparisons in code are straightforward to build:

- `==` - xor then nor bits of output
- `!=` - same as `==` without not of output
- `<` - consider `x < 0`

$x < y$ $\quad\quad$ $x - y < 0$

Comparisons

Each of our comparisons in code are straightforward to build:

- `==` - xor then nor bits of output
- `!=` - same as `==` without not of output
- `<` - consider `x < 0`
- `>`, `<=`, `=>` are similar

Indexing

Indexing with square brackets: `[ ]`

- **Register bank** (or **register file**) - an array of registers
  - Can programmatically pick one based on index
  - I.e., can determine which register while running
- Two important operations:
  `x = R[i]` - Read from a register
  `R[j] = y` - Write to a register

Reading

`x = R[i]` - connect output of registers to *x* based on index *i*