Build your own FM transmitter using an FPGA

...and rickroll your neighbors![1]

Nina Engelhardt

December 7, 2014

[1] Note: This is illegal in most places. But fun!

Thanks for coming!

This is the second slide

A bit more information about this

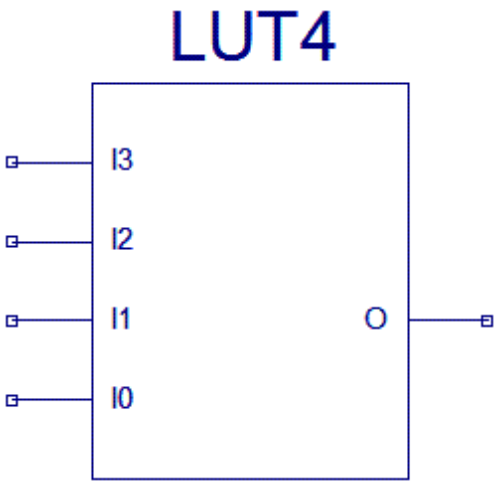
- Right now I'll talk your ear off a bit
  - boolean logic
  - what's in an FPGA?
  - combinatorial vs synchronous circuits
- If you've heard that stuff before, or like to know what's coming, you can already start reading/working on the tutorial files
  - tutorial.pdf = introduction to migen, how to build combinatorial and synchronous logic in migen
  - fmtransmitter.pdf + fm_transmitter_your_name_here.py = you know, that thing you came here for
- ssh into the lab computer to get them
  - username + password on the wall

Boolean Logic

- Boolean Algebra ($\mathbb{Z}/2\mathbb{Z}$)
  - Values $= 0, 1$
  - Operators $=$ and, or, not, xor (`&`, `|`, `!`, `^`) ($\wedge, \vee, \neg, \oplus$)
- not getting into all the properties
- Boolean Functions
  - $f(i_1, ..., i_n) = (o_1, ..., o_m)$
  - many possible ways to express one function as formulas
    - $f(i_1, i_2, i_3) = (i_1 \wedge i_2) \vee i_3 = (i_1 \vee i_3) \wedge (i_2 \vee i_3)$
  - but one unique representation: truth table

a	b	(a $\vee$ b)
1	1	1
1	0	1
0	1	1
0	0	0

Inside an FPGA

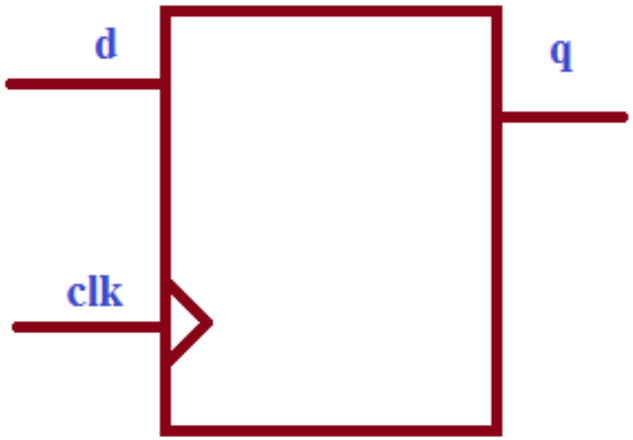


A	B	C	D	Out
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

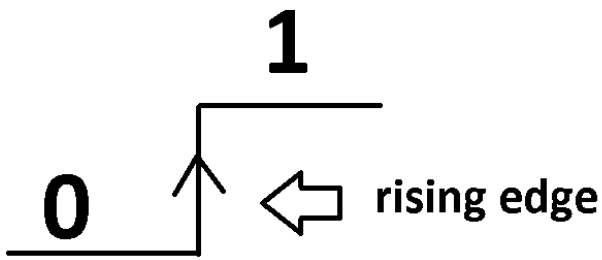
- Truth table!
- any function with more inputs (or outputs) is broken up into multiple truth tables

Inside an FPGA

The LUT's trusty sidekick: the flip-flop

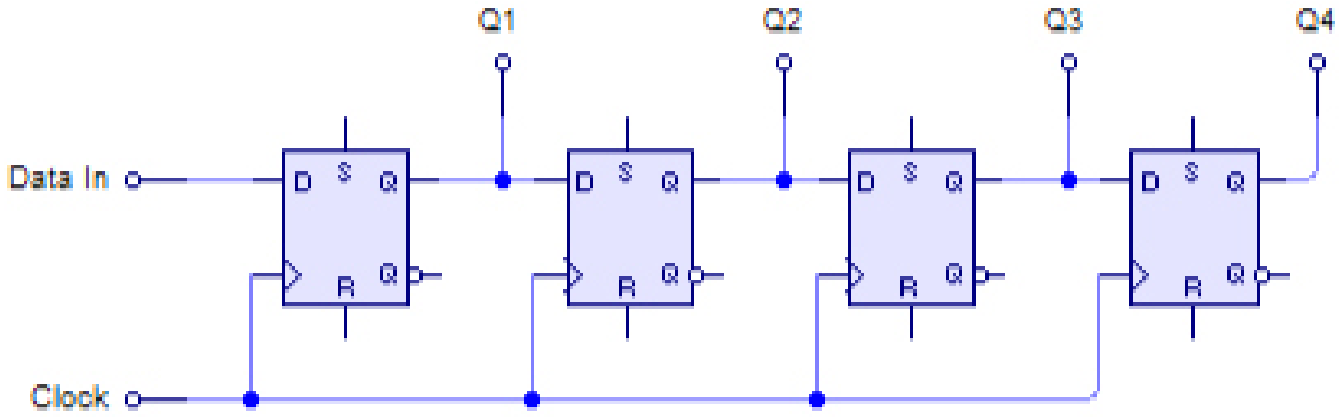


- Saves input value d when you tell it "now!"
- Outputs saved value on q
- Saying "now!" $=$ rising edge on port clk



Inside an FPGA

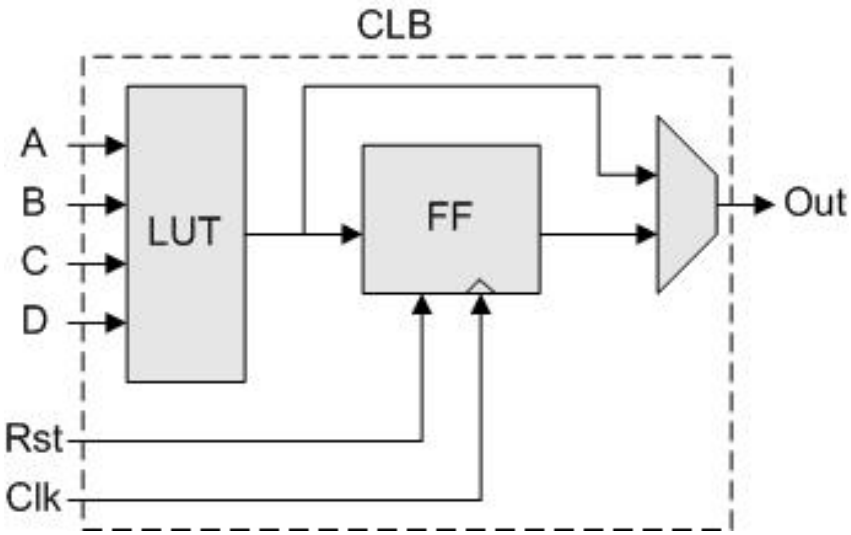
The LUT's trusty sidekick: the flip-flop



- Very important that all flip-flops get "now!" signal at the same moment (synchronous)
- Otherwise Q1 might have already changed when 2nd FF gets signal
- Clock signals are special, have their own routing network

Inside an FPGA

A maze of square little logic blocks, all alike

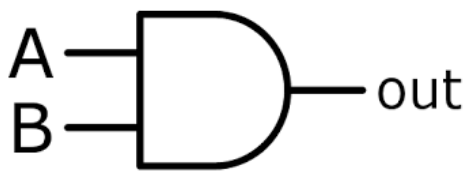


- Lots of these all over the FPGA
- Configurable routing network to connect them however you want
- Programming an FPGA = filling truth tables and routing signals (not by hand, thankfully)

Hardware Design

it's not programming

- Hardware design is describing how you want to connect wires, logic, and FFs together
- Not at all like writing code: spatial instead of temporal
- But looks very similar: `out = A & B`
- Keep in mind what you get is this:



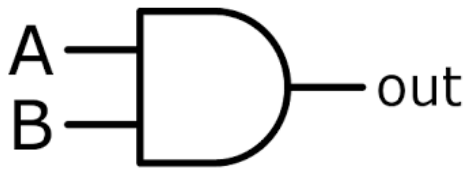
Hardware Design

but it's a lot like programming

- need to define variables first:

```
out = Signal()
A = Signal()
B = Signal()
```

- `out.eq(A & B)`



- Just floating in space like that it's no good, add it to a module

```
module.comb += out.eq(A & B)
module.sync += out.eq(A & B)
```

- adding to `sync` will make `out` into a register (FF)
- then if you read `out`, you get the value from previous cycle!

Hardware Design

but it's not programming

```
out.eq(out + 1)
```

- if you add it to `combinatorial` statements:

  ```
  out = (((((out + 1) + 1) + 1) + 1) + 1) + ...
  ```

- does not compute (combinatorial loop)

- if you add it to `synchronous` statements:

  ```
  out = previous_out + 1
  ```

- oh look, a counter!

That's it from me

fly, my lovelies, fly!

- Now you are equipped to start the tutorial
- We'll go through it together when you're done
- And then I'll unleash you onto the FM transmitter