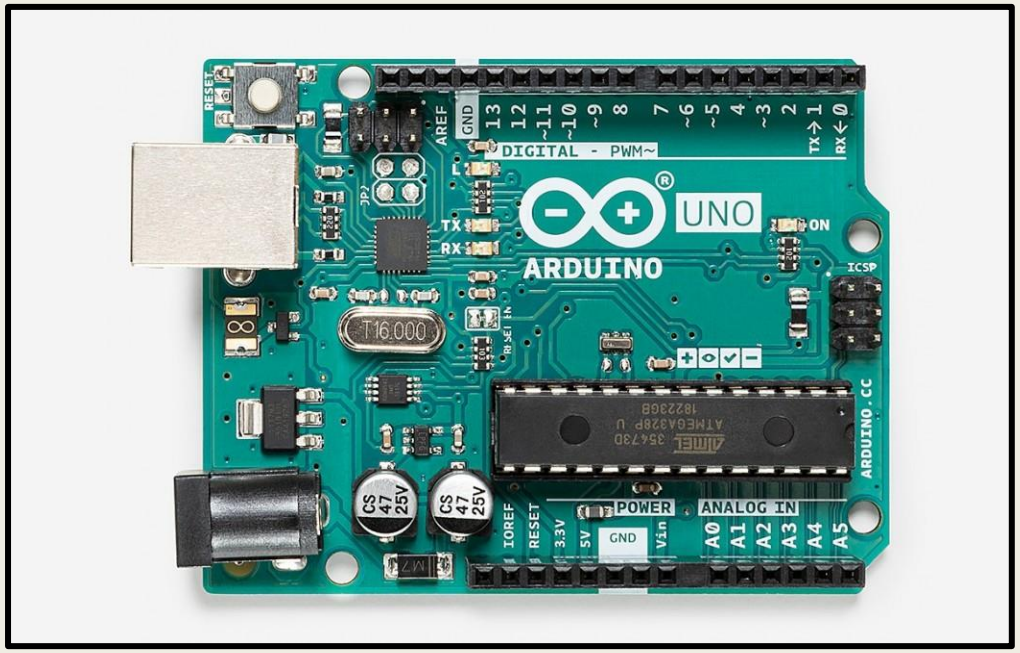
Monthly Mini Hacks

From Arduino to FPGA: Synthesizing hardware from code!

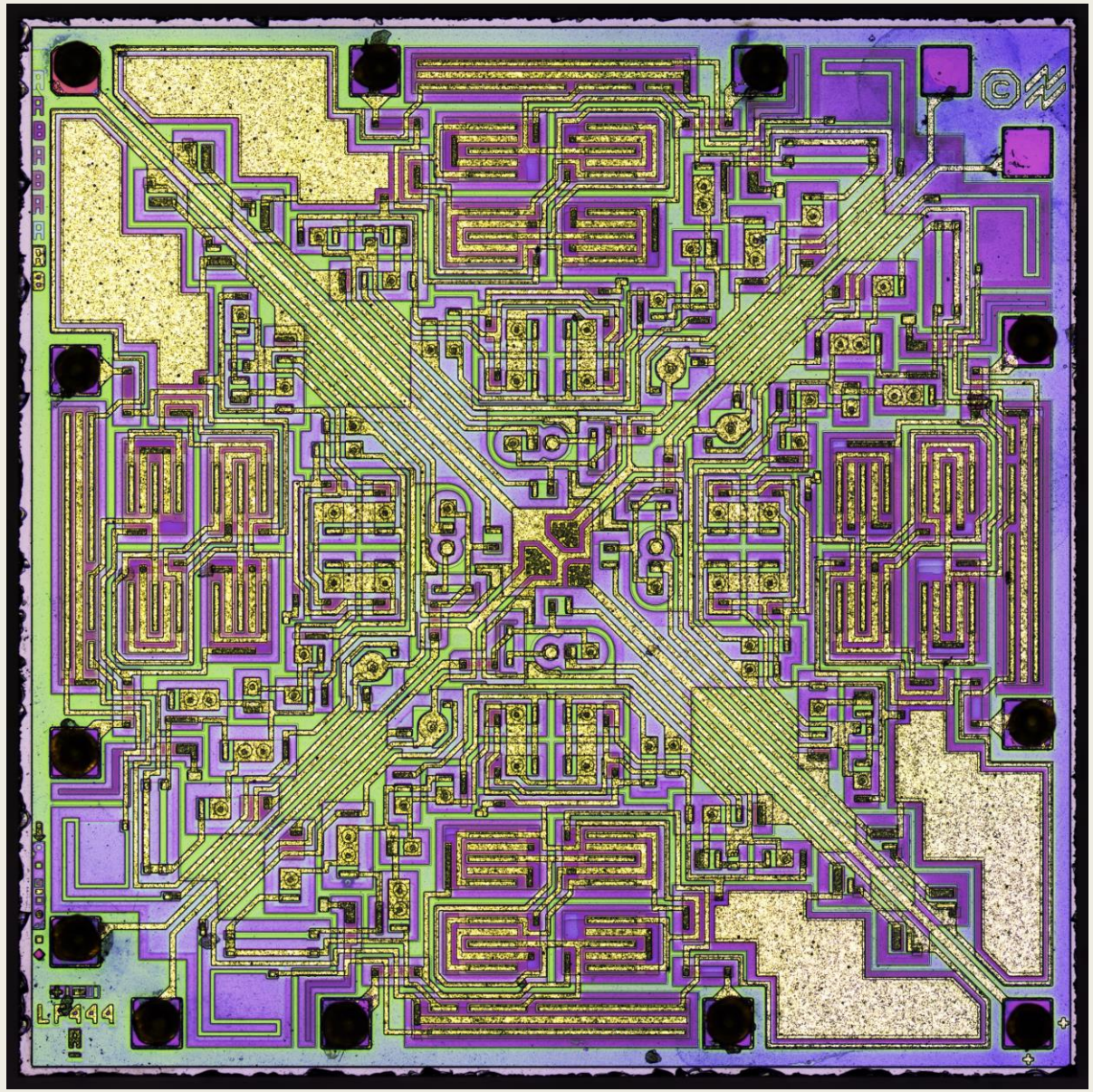
July 31st, 2020

Justin Silver



Plan

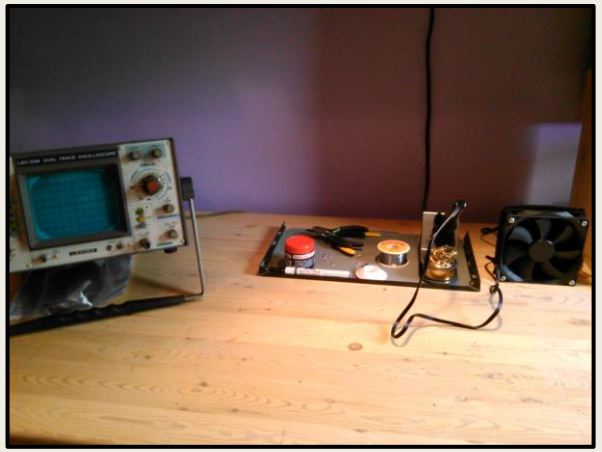
1. Presenter introduction
2. Topic presentation
3. Arduino introduction
4. Coding – Arduino
5. Digital logic principles
6. FPGA and VHDL introduction
7. Coding – FPGA
8. Improvements
9. Useful links

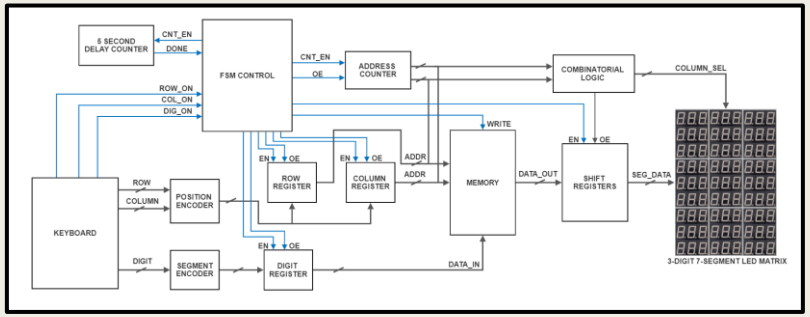


Die photo of LF444 – quad JFET op-amp *(credit: zeptobars.com)*

A little about me

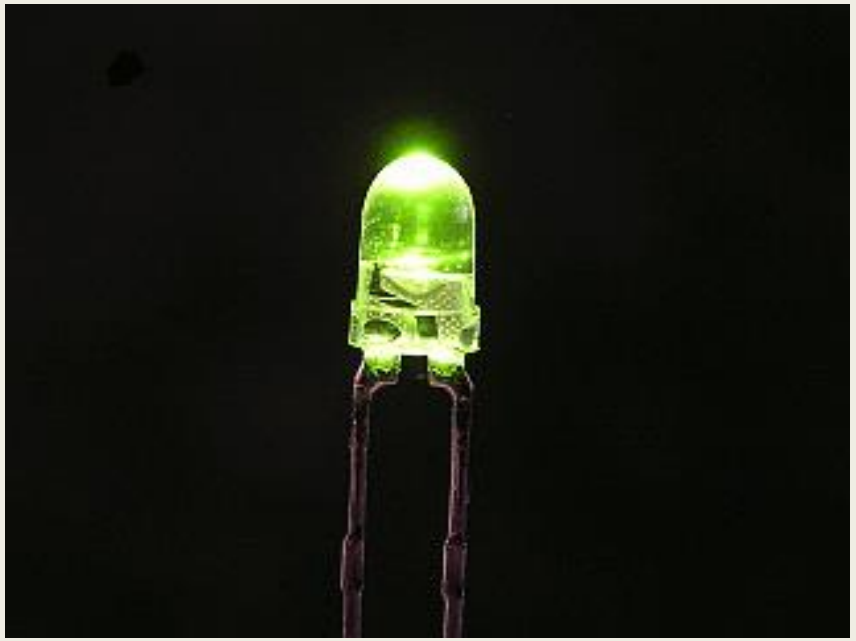
- American and British
- Florida State University (Chemistry and French)
- Reorientation to Electrical Engineering
- Personal electronic projects
- www.justin-silver.com
- Passion for speaking French
- University of Strasbourg, France (Electronics)





Topic presentation

- Let’s blink an LED!
- Hardware equivalent of “Hello, World!”
- Arduino implementation (C/C++)
- FPGA implementation (VHDL)



But like… why??

- Spark your interest in electronics and programming
- Coding software vs. hardware
- Abstraction (facility) vs. complexity (flexibility)
- Learn about digital logic

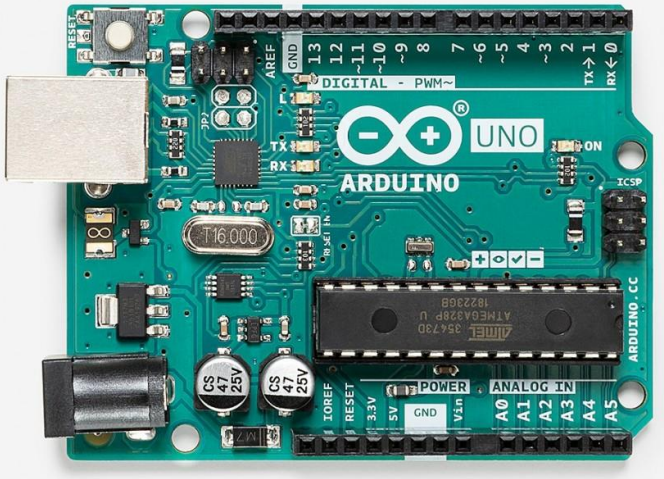
Arduino time…

Arduino introduction

From the official Arduino website…

What is Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.



- Originally started as a research project by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, and David Mellis at the Interaction Design Institute of Ivrea in the early 2000s
- The first Arduino board was released in 2005 for students with no previous knowledge of electronics and programming to prototype and create devices
- C/C++ programming language
- Hobbyist/maker movement

Example projects

Arduino Controlled Pinball Machine

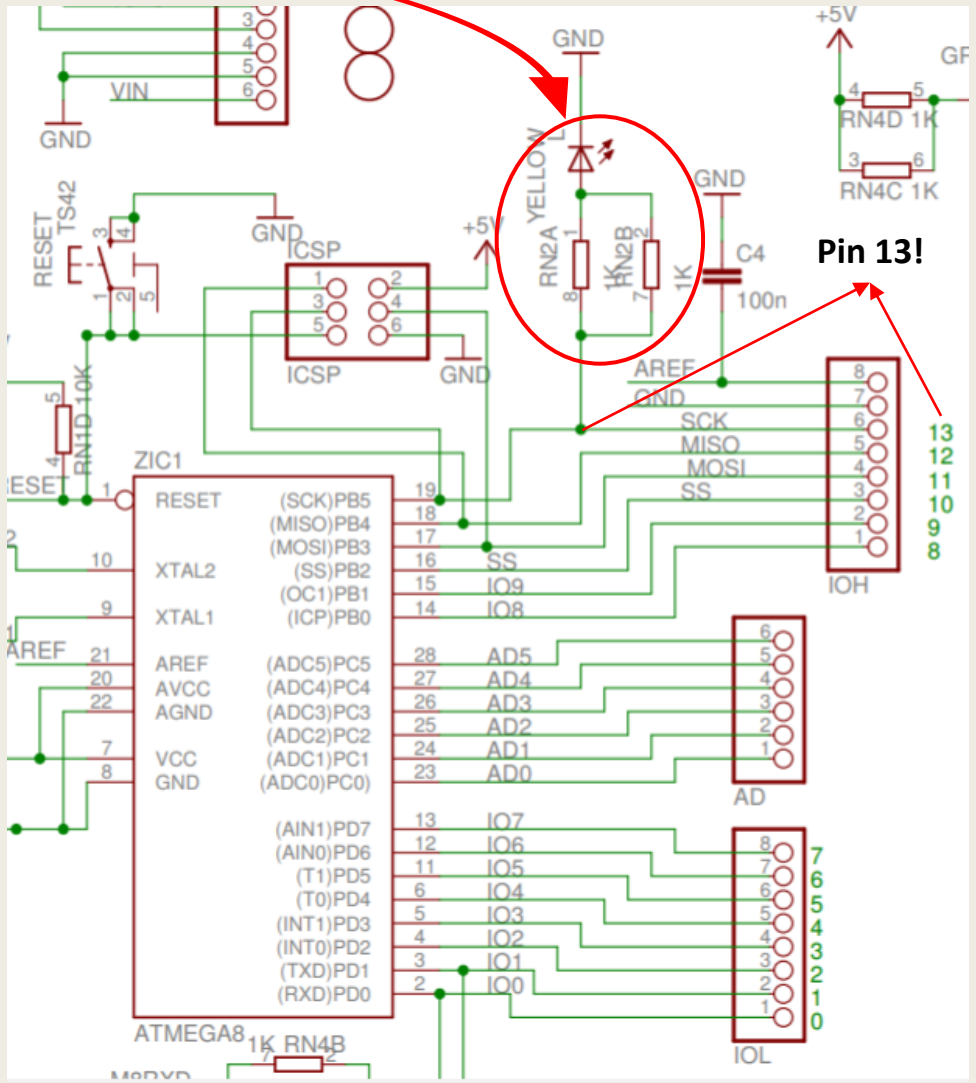
Gizmo, App and Voice Controlled Robotic Vehicle

UAV Arduino

Autonomous Home Assistant Robot

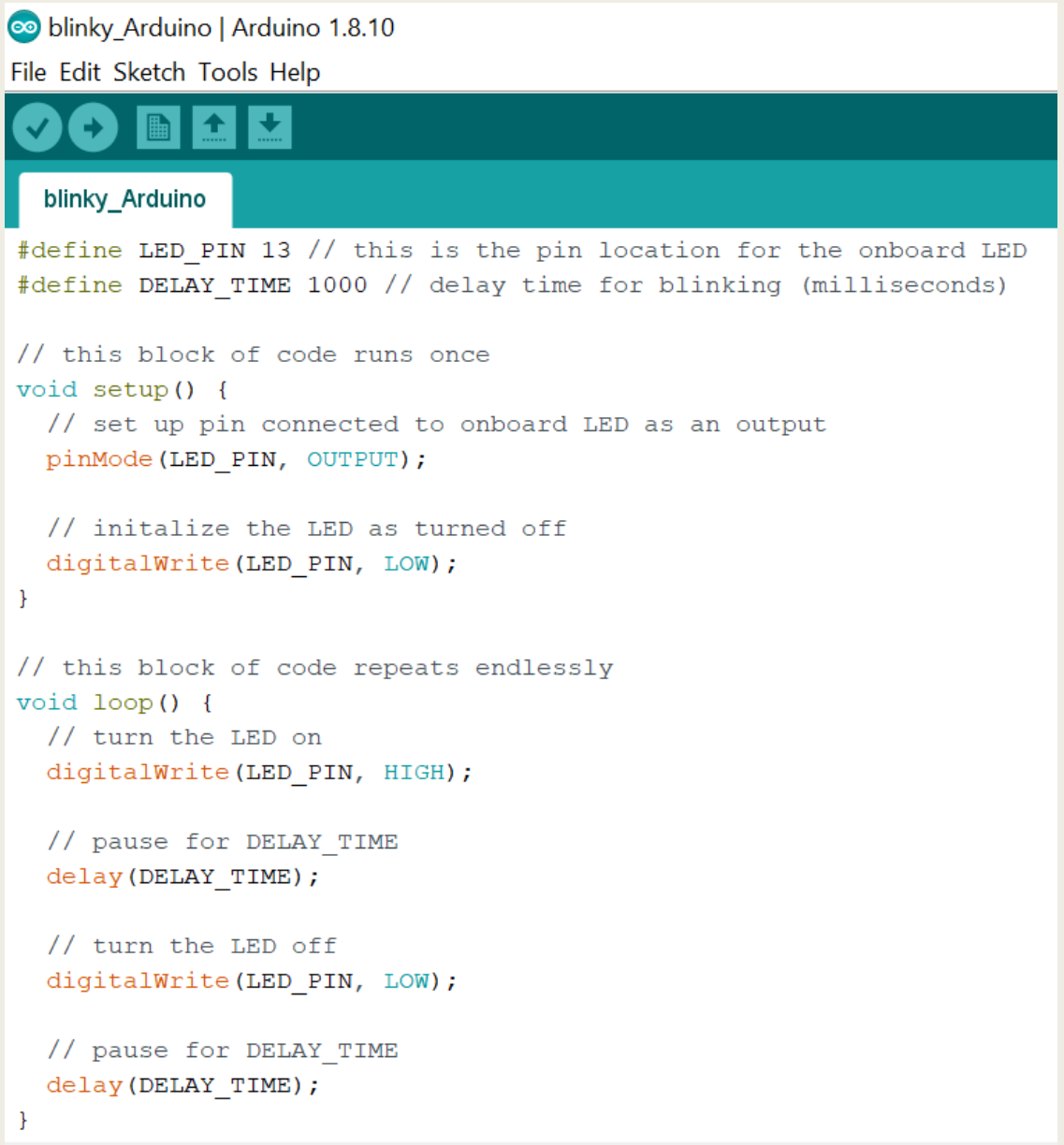
Coding - Arduino

- ELEGOO is an open-source hardware and software company – they build Arduino kits and UNO clones
- Although an external LED could be connected, let's blink the onboard one that's included with the board



Schematic layout for the Arduino UNO board

Coding - Arduino




```
#define LED_PIN 13 // this is the pin location for the onboard LED
#define DELAY_TIME 1000 // delay time for blinking (milliseconds)

// this block of code runs once
void setup() {
  // set up pin connected to onboard LED as an output
  pinMode(LED_PIN, OUTPUT);

  // initalize the LED as turned off
  digitalWrite(LED_PIN, LOW);
}

// this block of code repeats endlessly
void loop() {
  // turn the LED on
  digitalWrite(LED_PIN, HIGH);

  // pause for DELAY_TIME
  delay(DELAY_TIME);

  // turn the LED off
  digitalWrite(LED_PIN, LOW);

  // pause for DELAY_TIME
  delay(DELAY_TIME);
}
```

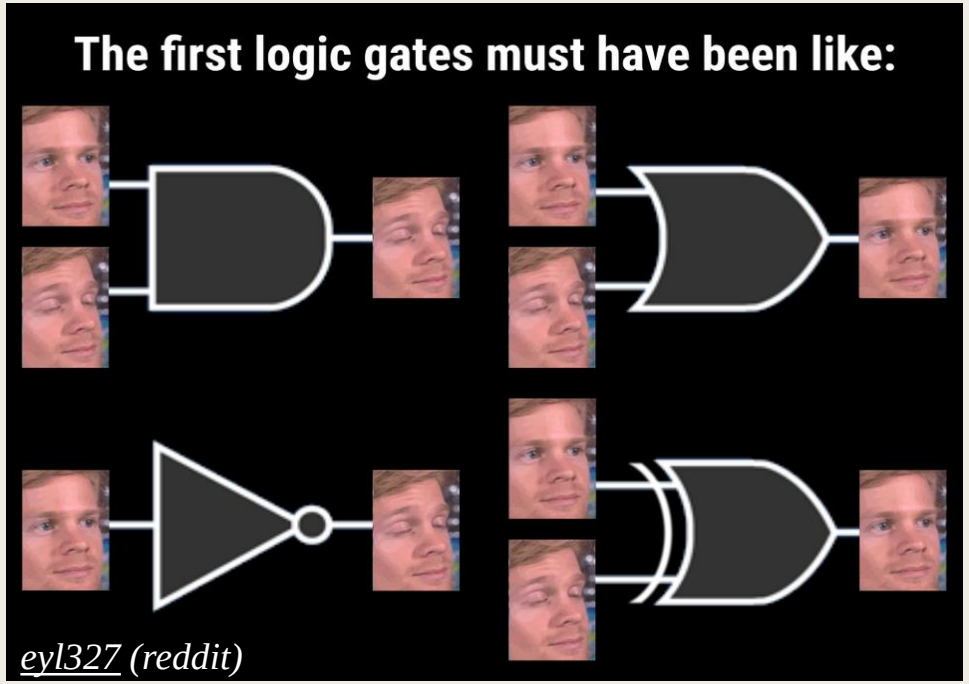
- Simple blinking program that turns on and off the LED (1 Hz)
- This is achieved by using high level Arduino functions (pinMode, digitalWrite, etc.)
- In-line comments preceded by “//”

FPGA time…

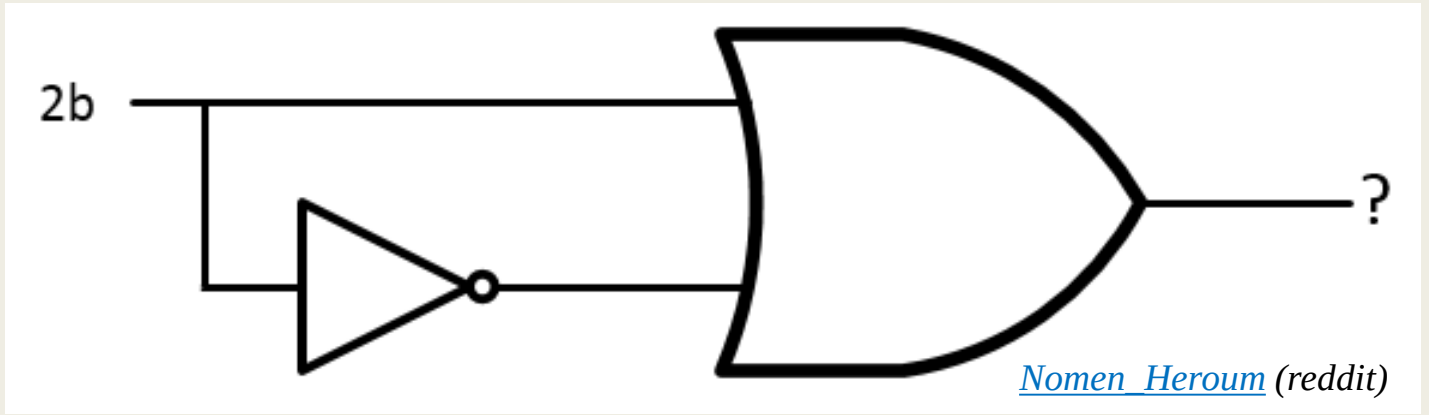
But before that….

Memes!

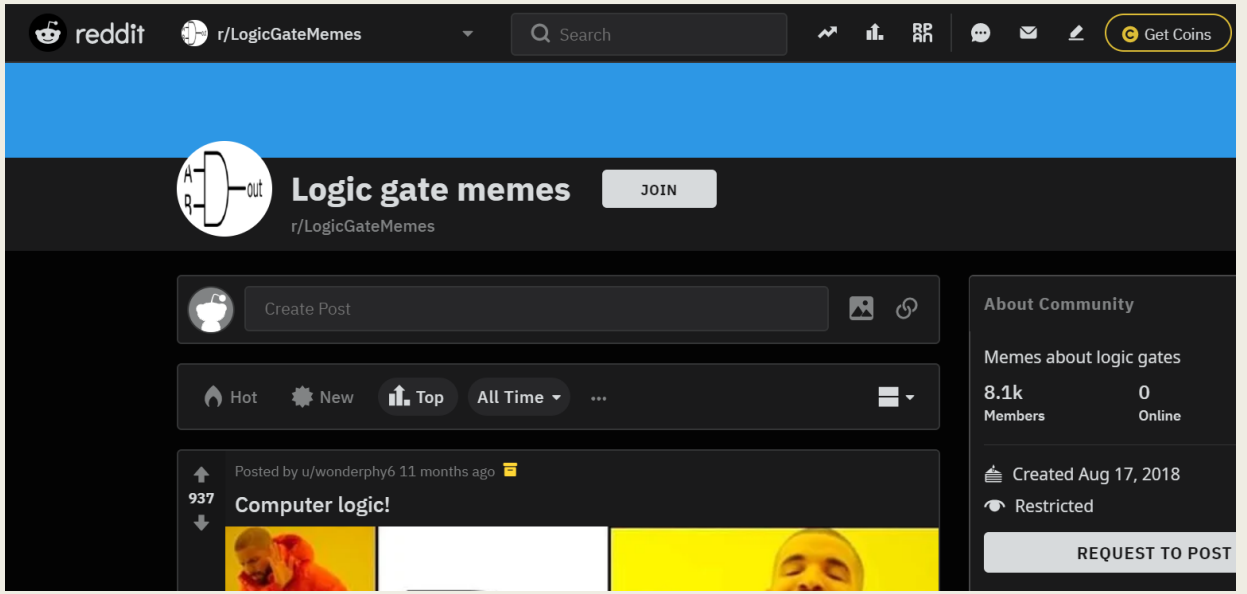
Digital logic principles



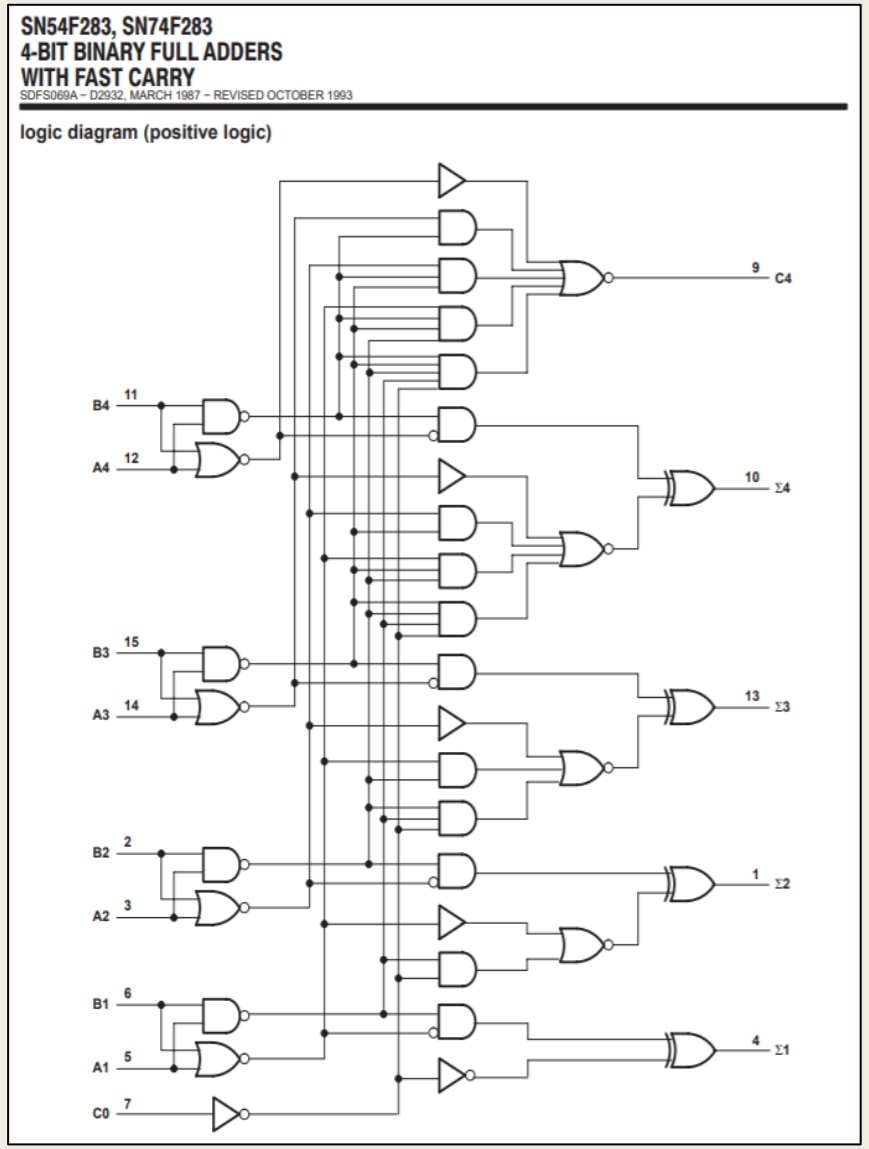
eyl327 (reddit)



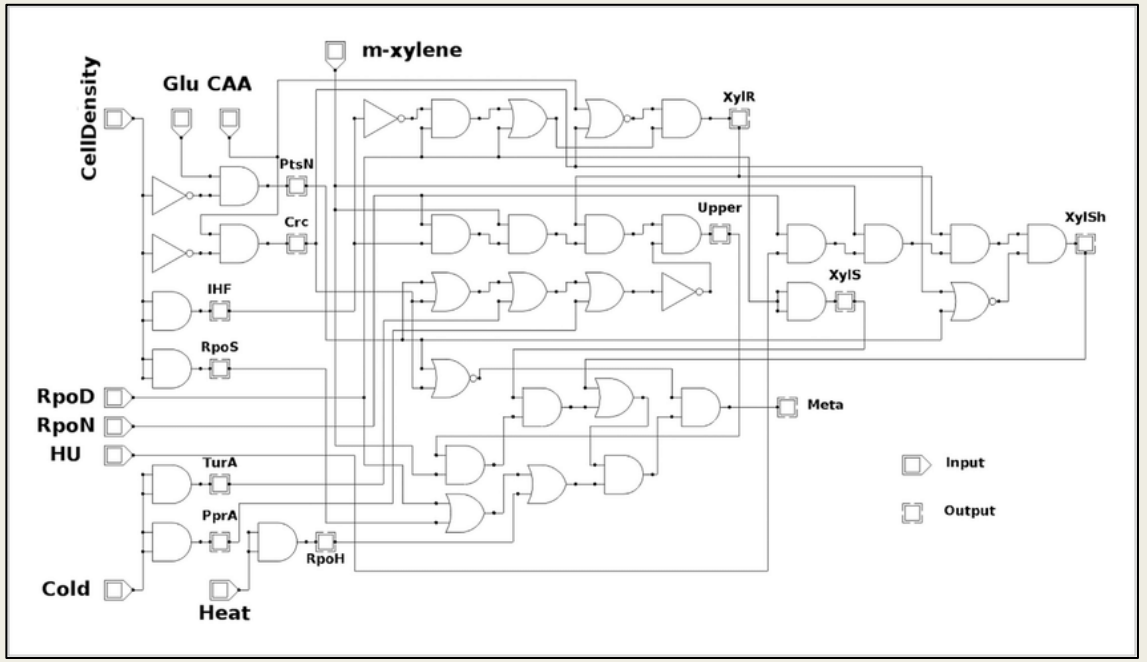
Nomen_Heroum (reddit)



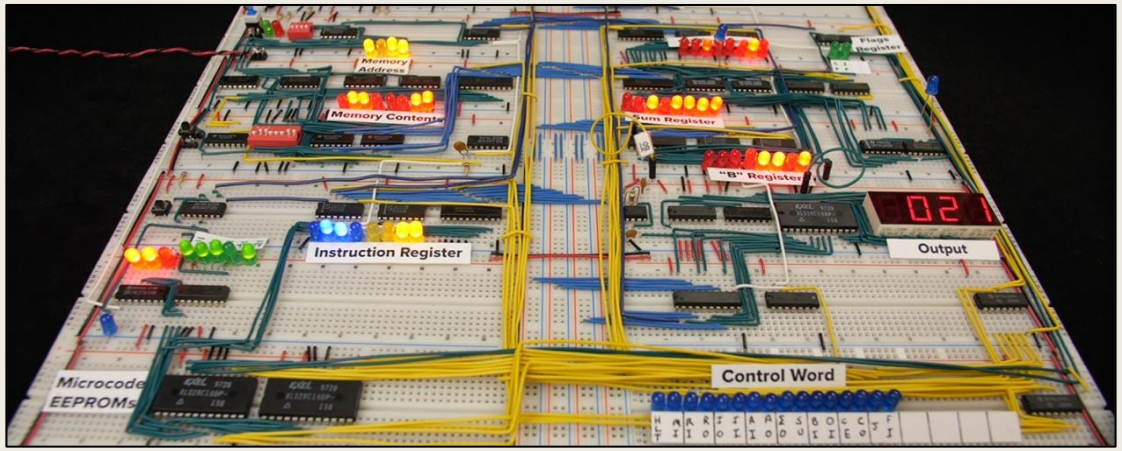
Cool applications!



Texas Instruments 4-bit full adder (SN54F283)

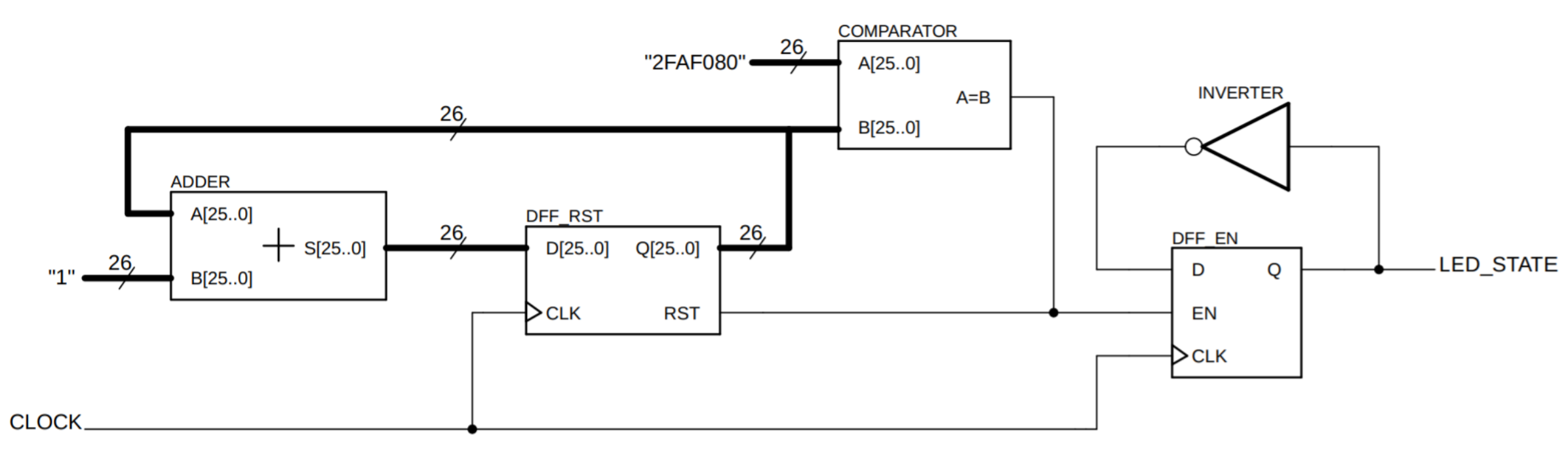


V. de Lorenzo, M. Schmidt, Biological standards for the Knowledge-Based BioEconomy: What is at stake, New Biotechnol. (2017), http://dx.doi.org/10.1016/j.nbt.2017.05.001



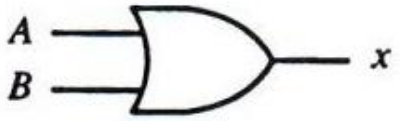
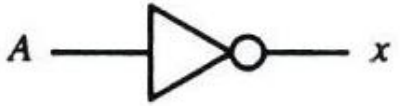
8-bit breadboard computer made by Ben Eater (www.eater.net)

Digital logic – blinking an LED

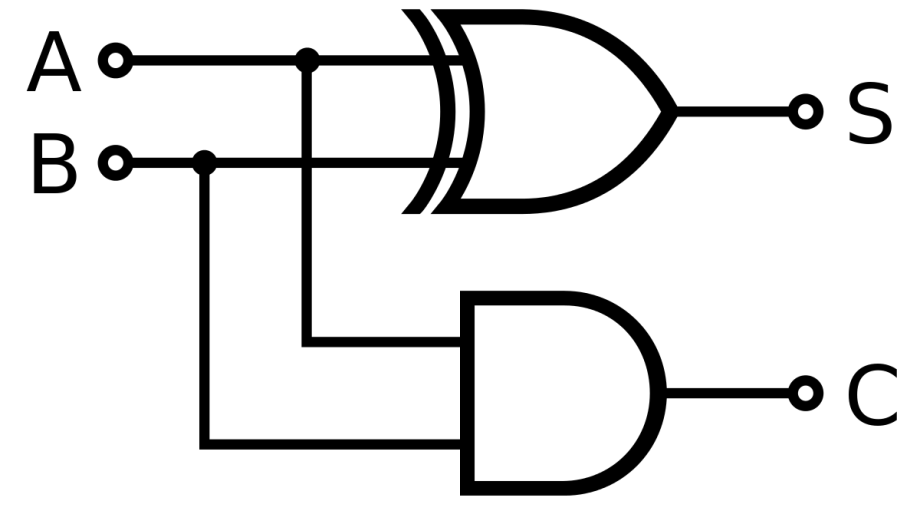


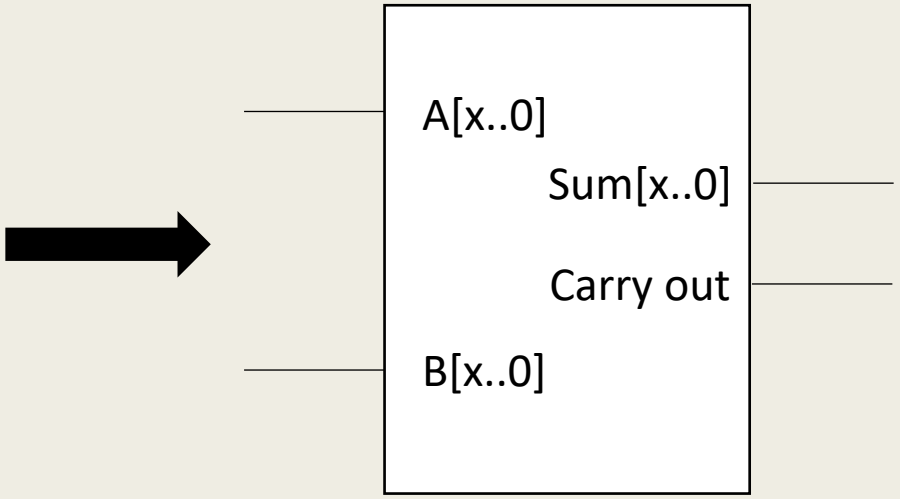
- This is the circuit we will build (inside the FPGA) to blink an LED
- Let's break it down into simple components and see how each one works

Basic logic gates

Name	*Graphic symbol*	*Algebraic function*	*Truth table*
AND		$x = A \cdot B$ or $x = AB$	A B | x 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$x = A + B$	A B | x 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
Inverter		$x = A'$	A | x 0 | 1 1 | 0

Adders



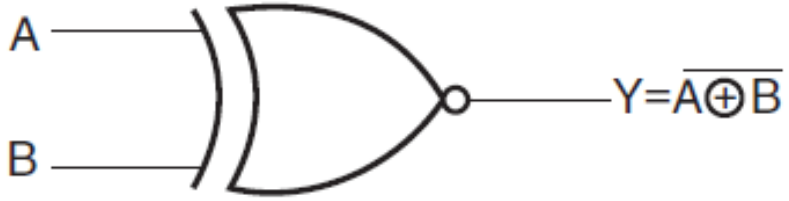


Truth table

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

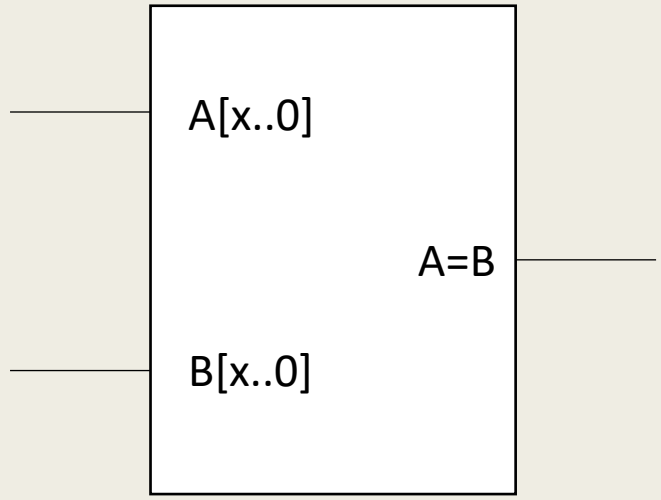
Comparators

A simple comparator is obtained from a single logic gate (XNOR)

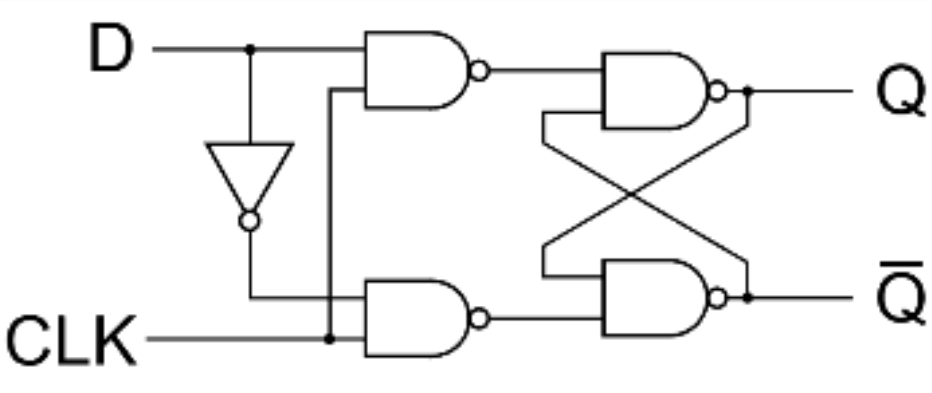


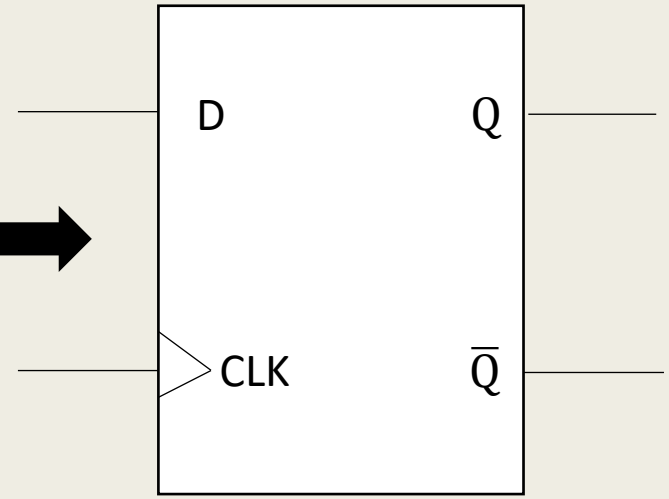
$$Y = (\overline{A \oplus B}) = (A.B + \overline{A}.\overline{B})$$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1



D Flip-Flop (register)

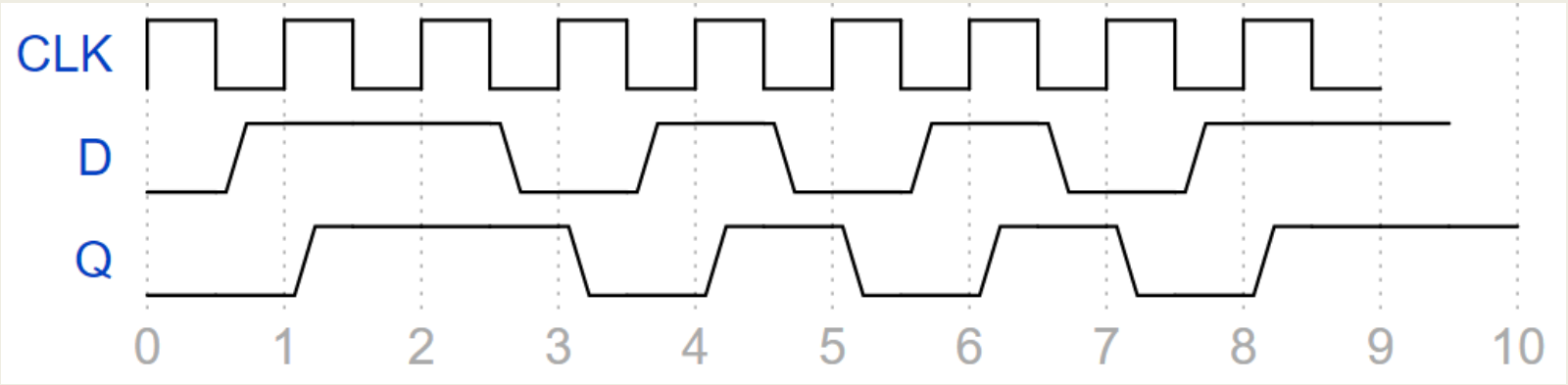




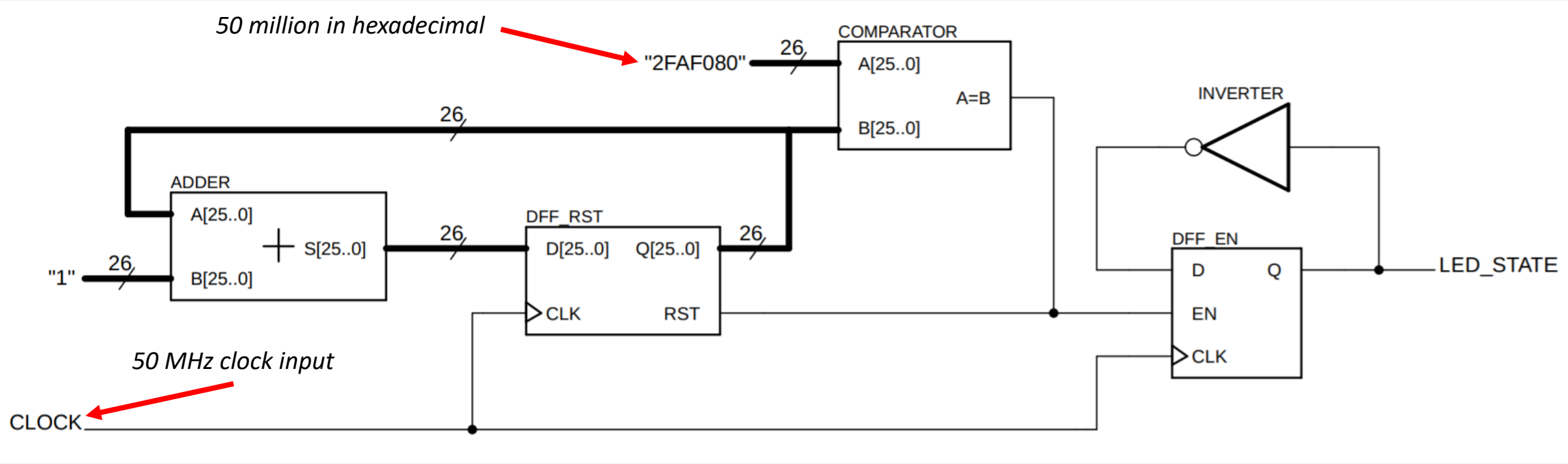
Truth table

CLK	D	Q	$\bar{Q}$
0	X	Q	$\bar{Q}$
1	X	Q	$\bar{Q}$
↑	0	0	1
↑	1	1	0

Chronogram (timing diagram)



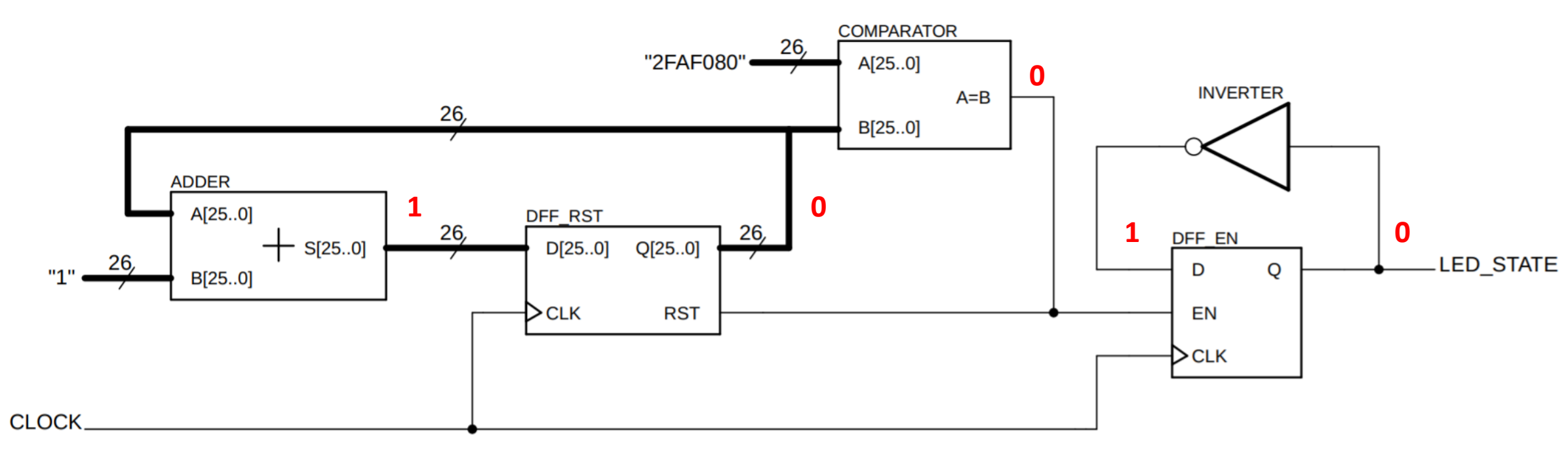
Taking another look…



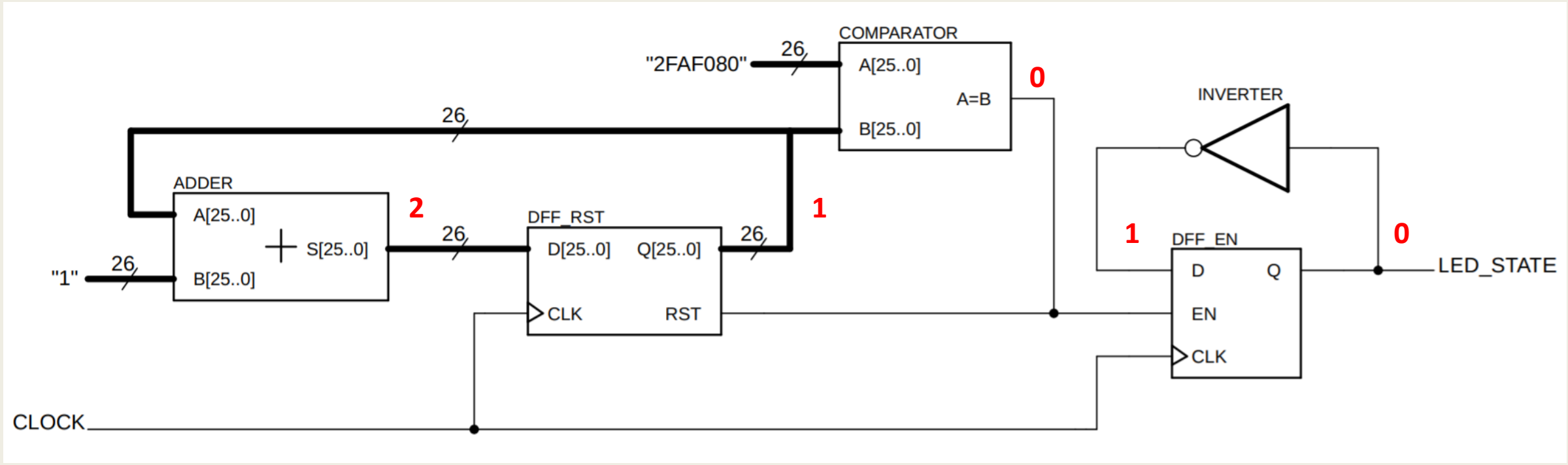
- This circuit takes in a 50 MHz clock, counts to 50 million, switches the LED state, and then resets
- Cycle repeats endlessly

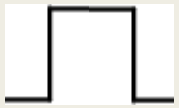
Taking another look…

Initial state – LED is OFF



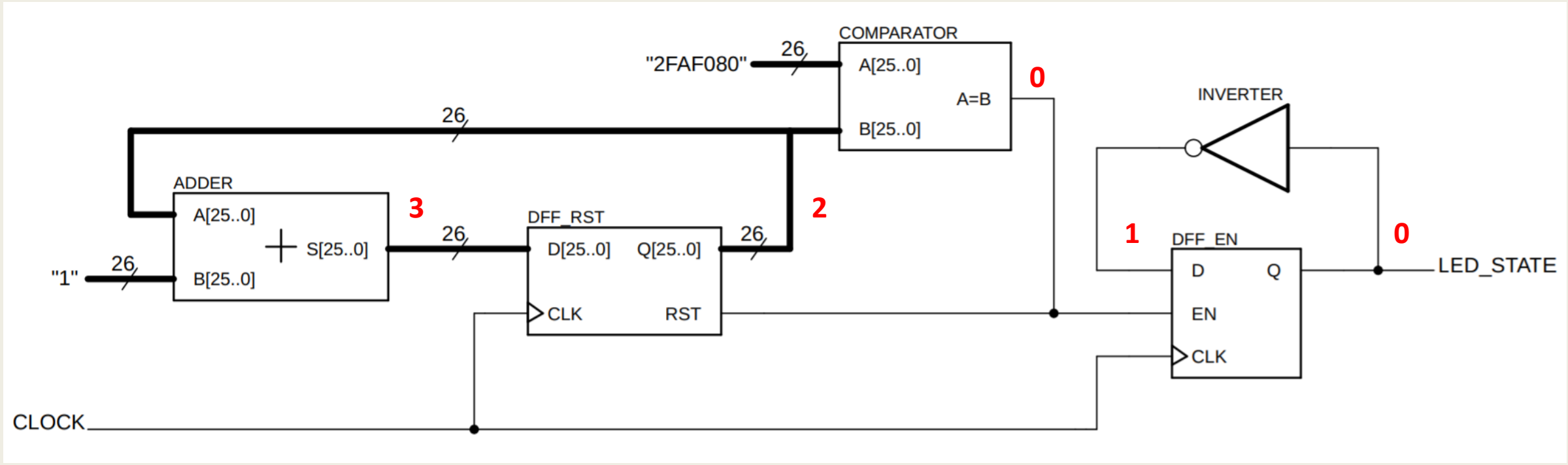
Taking another look…

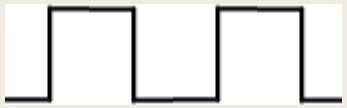




1st clock cycle… (rising edge)

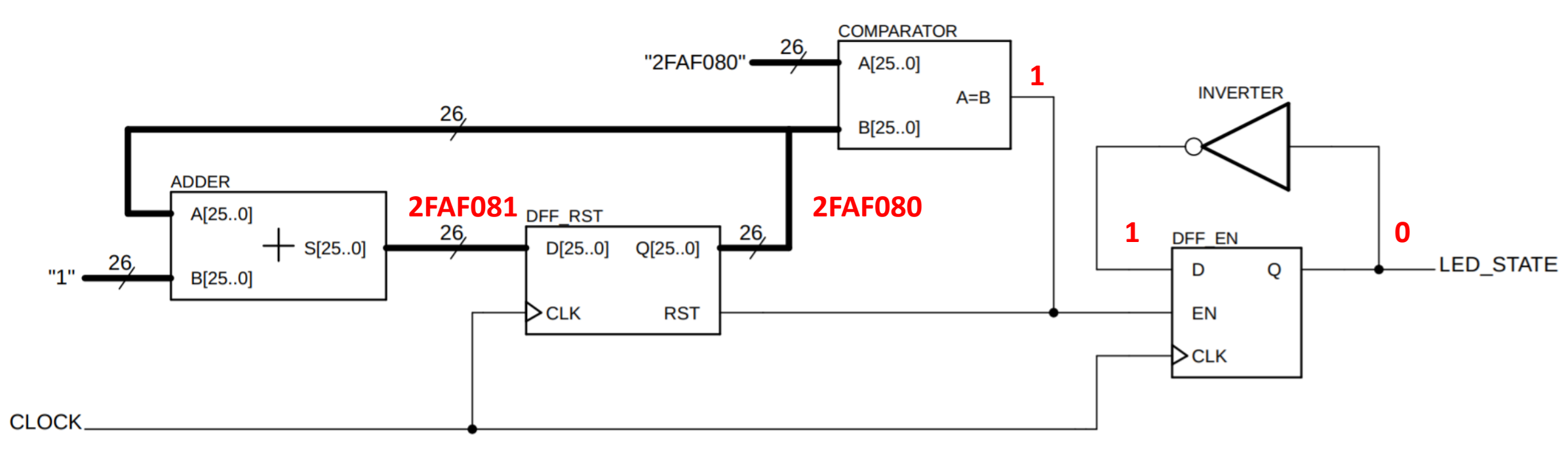
Taking another look…

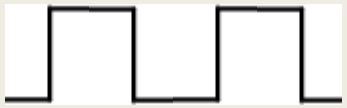




2nd clock cycle… (rising edge)

Taking another look…

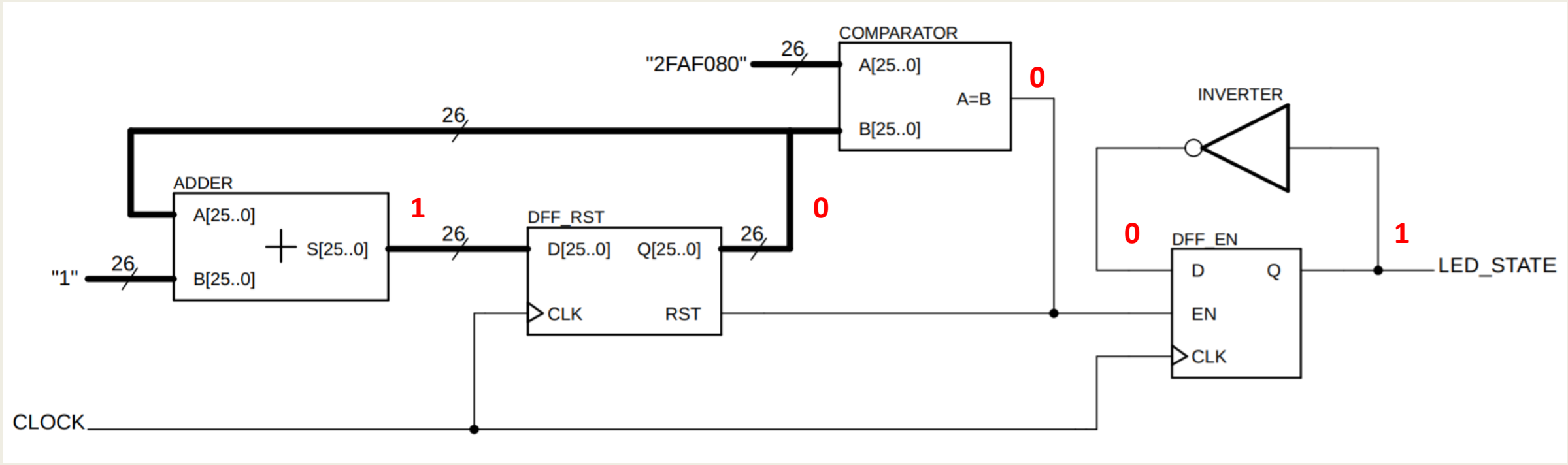




……

The 50,000,000th clock cycle…

Taking another look…



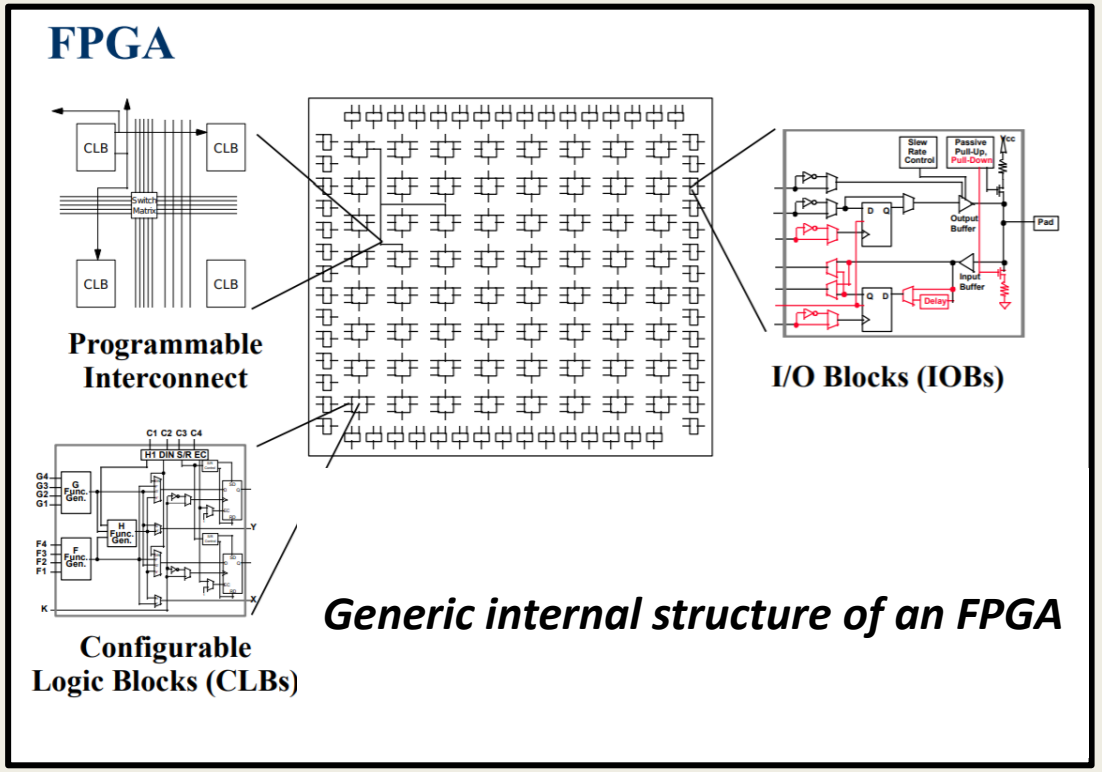
.....

Back where we started, but now the LED is on!

Introduction – FPGA and VHDL

FPGA

- Field Programmable Gate Array
- Predecessors: PLD, CPLD
- Synthesis of any digital logic hardware, completely programmable
- **Parallel** architecture
- Coded using VHDL, Verilog, schematic entry
- FPGA ≠ microcontroller*



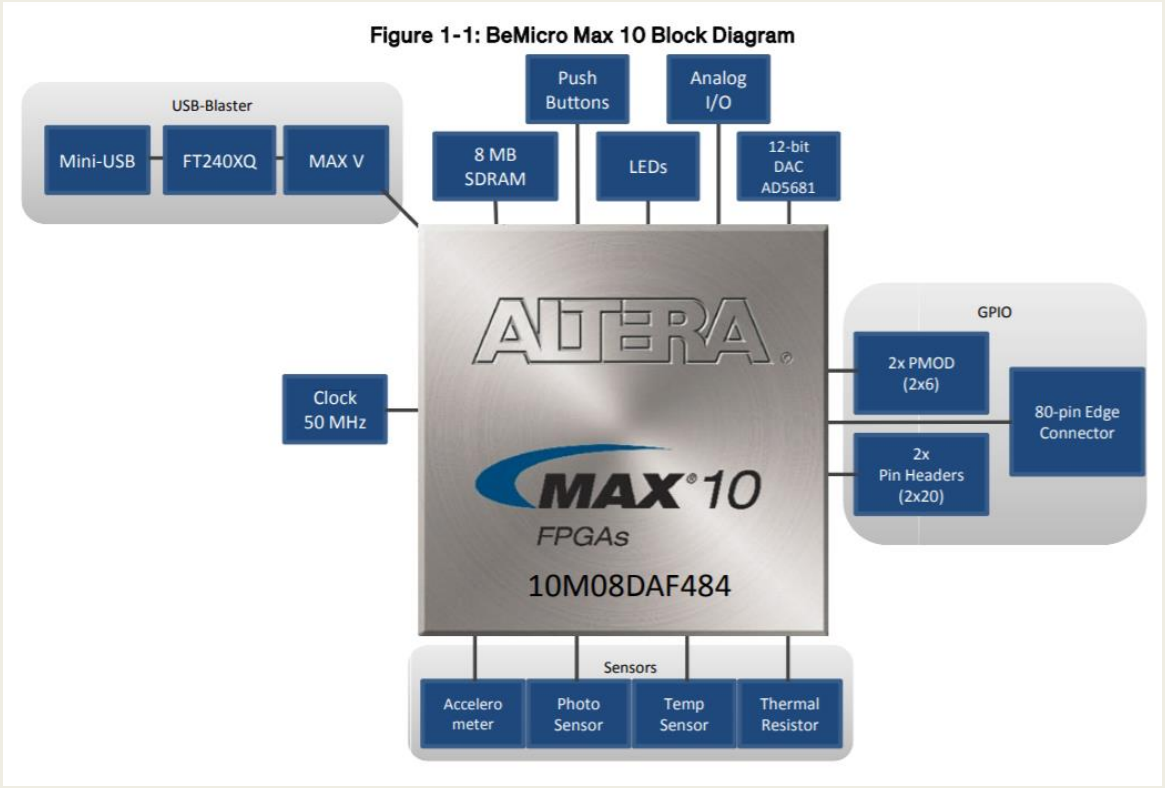
VHDL

- VHSIC-HDL, Very High Speed Integrated Circuit Hardware Description Language
- Behavioral vs. structural approach
- **Parallel** code execution

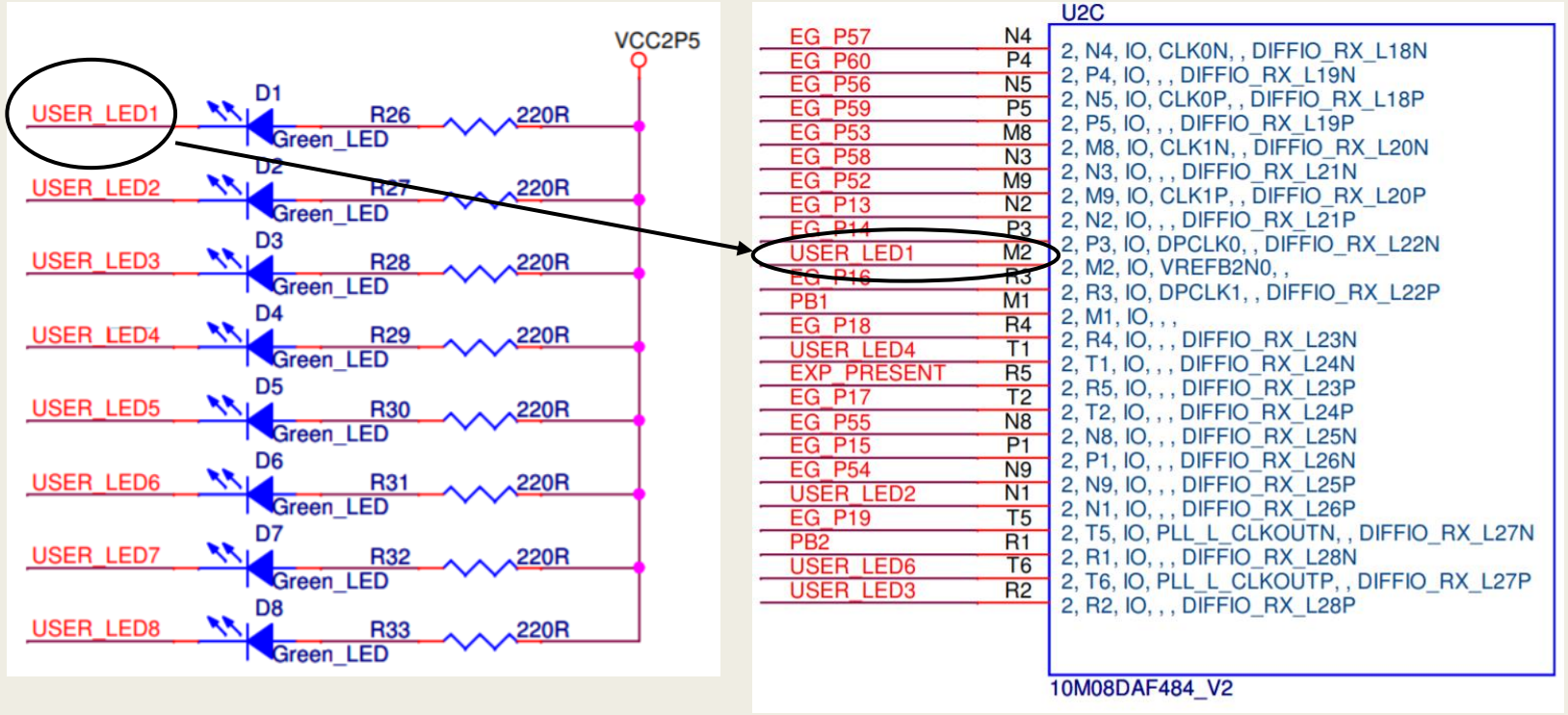
*it could be programmed to be though… https://www.fpga4student.com/2016/12/a-complete-8-bit-microcontroller-in-vhdl.html

FPGA hardware

- I will be using a BeMicro MAX 10 FPGA development kit
- These “Dev Kits” have everything you need to start programming the FPGA
- They also include some onboard peripherals (LEDs for example)



Pin assignments - FPGA



Signal Name	MAX 10 Pin	Description
SYS_CLK	N14	50 MHz "System" Clock Input

- Just like with the Arduino, we have to find out which pin of the FPGA is connected to the onboard LED (we will use USER_LED1)
- Notice that if the USER_LED1 output pin is HIGH, the LED is OFF!
- We will also note the 50 MHz clock pin (for our counting circuit)

VHDL code - adder

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
-- the std_logic_1164 package is the IEEE
-- standard for describing digital logic
-- values (std_logic and std_logic_vector types)

-- the std_logic_unsigned package provides
-- support for performing arithmetic functions
-- with std_logic vectors (we will use vector addition)

-- here is where we define the inputs/outputs of the adder
entity adder is
    port(
        a   : in std_logic_vector(25 downto 0);
        sum : out std_logic_vector(25 downto 0)
    );
end entity adder;

-- here is where we define the function
-- performed by the adder
architecture logic of adder is
begin
    -- whenver input "a" changes, the sum will
    -- automatically change
    sum <= a + 1;
end architecture logic;
```

VHDL code - comparator

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
-- the std_logic_1164 package is the IEEE
-- standard for describing digital logic
-- values (std_logic and std_logic_vector types)

-- here is where we define the inputs/outputs of the comparator
entity comparator is
    port(
        a : in std_logic_vector(25 downto 0);
        q : out std_logic
    );
end entity comparator;

-- here is where we define the function performed by the comparator
architecture logic of comparator is
    constant MAX_CLK_CYCLES : std_logic_vector(25 downto 0) := "10111110101111000010000000";
    -- the long binary number is 50 million in decimal
begin
    -- this with/select statement is similiar to the
    -- C programming language's switch/case function
    with a select
        -- whenever the input vector (a) reaches 50 million, the output of
        -- the comparator turns ON
        -- otherwise, the output is OFF
        q <= '1' when MAX_CLK_CYCLES,
             '0' when others;
end architecture logic;
```

VHDL code - registers

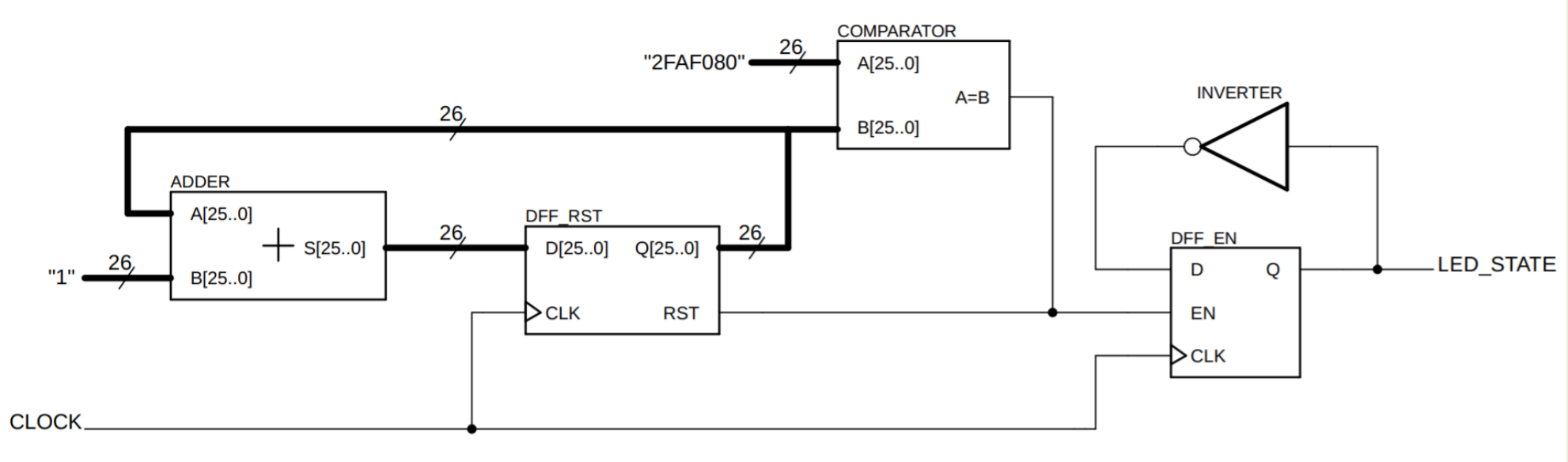
```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
-- the std_logic_1164 package is the IEEE
-- standard for describing digital logic
-- values (std_logic and std_logic_vector types)

-- here is where we define the inputs/outputs of
-- the DFF w/reset
entity register_rst is
    port(
        d       : in std_logic_vector(25 downto 0);
        rst,clk : in std_logic;
        q       : out std_logic_vector(25 downto 0) := (others => '0')
        -- "others => '0'" is a succinct way
        -- to set the bits of a vector to 0
    );
end entity register_rst;

-- here is where we define the function
-- performed by the DFF w/reset
architecture logic of register_rst is
begin
    -- this block of code infers a register
    process(clk)
    begin
        -- this is a synchronous reset DFF
        -- it only checks the state of the reset pin
        -- when the clock input transistions
        -- from low to high
        if rising_edge(clk) then
            -- if the reset signal is off
            -- copy the input to the output
            if (rst = '0') then
                q <= d;
            -- otherwise, set all of the output bits to 0
            else
                q <= (others => '0');
            end if;
        end if;
    end process;
end architecture logic;
```

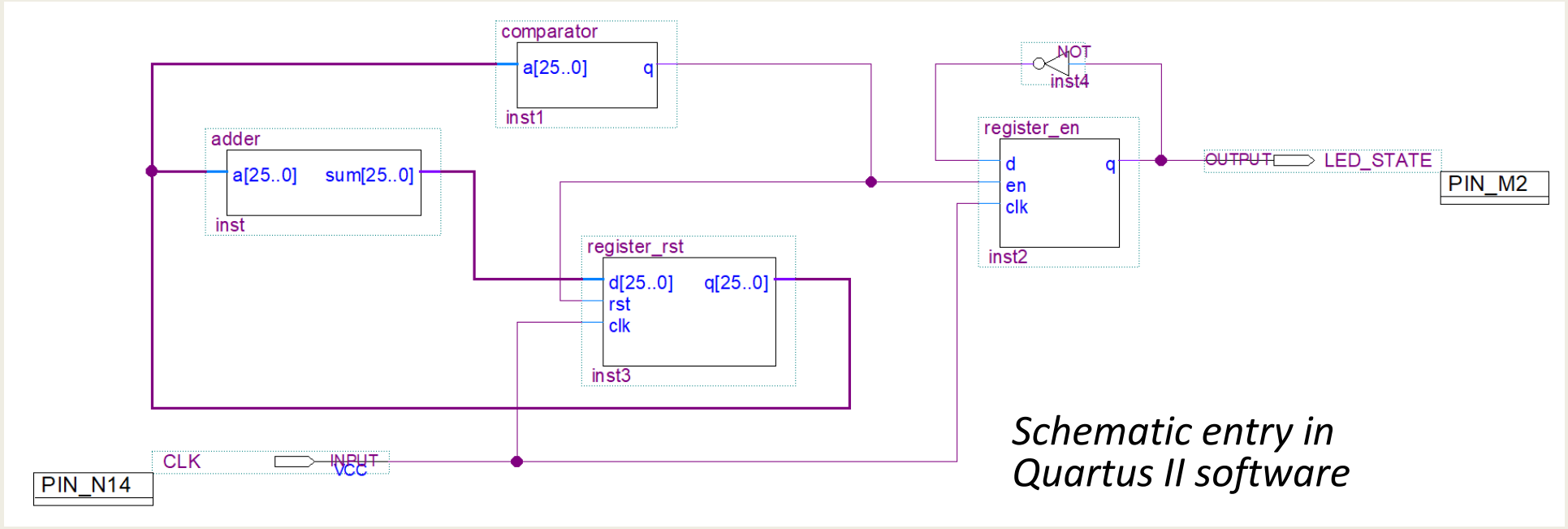
DFF w/reset

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
-- the std_logic_1164 package is the IEEE
-- standard for describing digital logic
-- values (std_logic and std_logic_vector types)

-- here is where we define the inputs/outputs of
-- the DFF w/enable
entity register_en is
    port(
        -- because d, en, and clk are of the same
        -- digital logic type, we can define them
        -- all at once by using commas
        d,en,clk : in std_logic;
        q : out std_logic := '0'
        -- the LED_STATE signal will be initialized
        -- to LOW
    );
end entity register_en;

-- here is where we define the function
-- performed by the DFF w/enable
architecture logic of register_en is
begin
    -- this block of code infers a register
    process(clk)
    begin
        -- this is a synchronous enable DFF
        -- it only checks the state of the enable pin
        -- when the clock input transistions
        -- from low to high
        if rising_edge(clk) then
            -- if the enable signal is on
            -- copy the input to the output
            if (en = '1') then
                q <= d;
            end if;
        end if;
    end process;
end architecture logic;
```

DFF w/enable

Putting it all together…





Schematic entry in Quartus II software

Improvements?

- Adding button inputs – each button gives a different delay length
- Connecting a thermistor – as the temperature increases, the blinking frequency increases
- Connecting a DC buzzer to the ON/OFF signal – now you have a makeshift metronome!
- Our structural FPGA code is not flexible: changing the delay time is not easy
- Speed, performance, compile time optimization within Quartus II

Useful links

****Source code for this presentation is available on my GitHub****

- Buying an Elegoo Arduino Starter Kit
- Check out more awesome Arduino projects
- Instructions for installing the Arduino software
- Arduino tutorials
- Buying a BeMicro MAX 10 FPGA dev kit
- Installing Quartus II (pay attention to the version)
- Basic Quartus II tutorial (Eric M. Schwartz - University of Florida)
- Intel YouTube page
- VHDL basics
- Tinkercad - Simulating online digital circuits (breadboard)
- Ben Eater (YouTube)
- EEVblog (YouTube)
- GreatScott (YouTube)

Thanks for reading!

…and a huge thanks to Monthly Mini Hacks, the Surrey Reproducibility Society, and Danielle Kurtin