Fermilab | U.S. DEPARTMENT OF ENERGY | Office of Science

Digital Design for Astrophysics Detectors

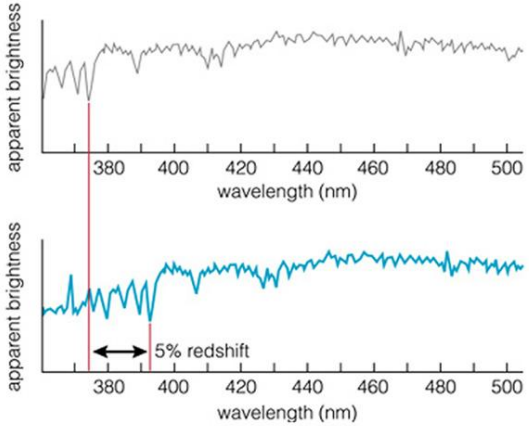
Collin Bradford

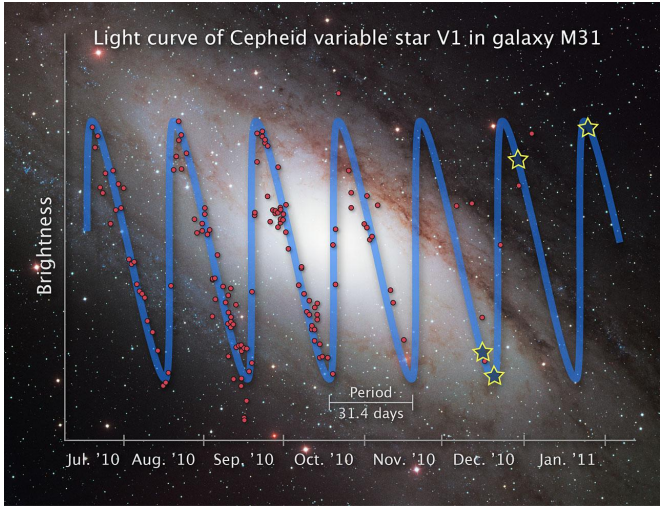
Quarknet Teachers' Workshop

28 July 2016

How to look at the sky

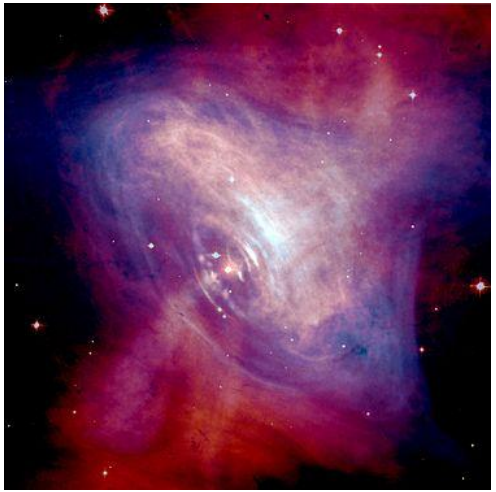
When astronomers look at the sky, they have to choose:

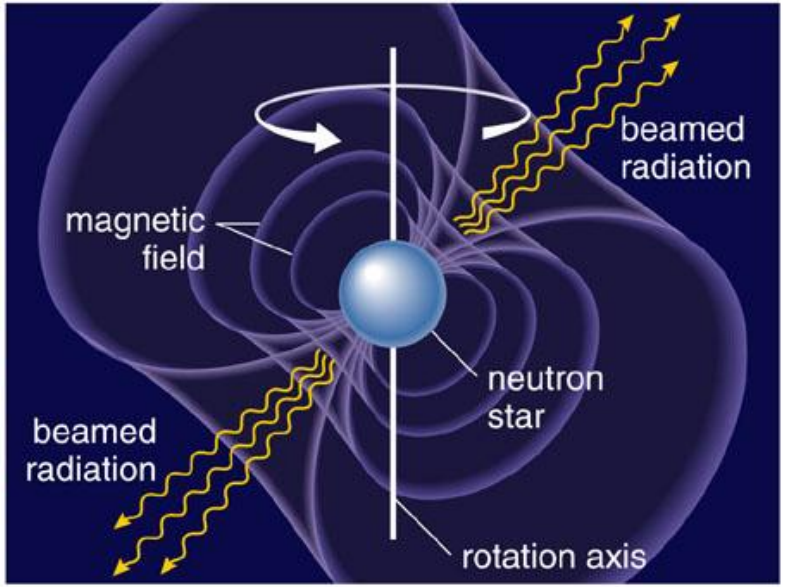


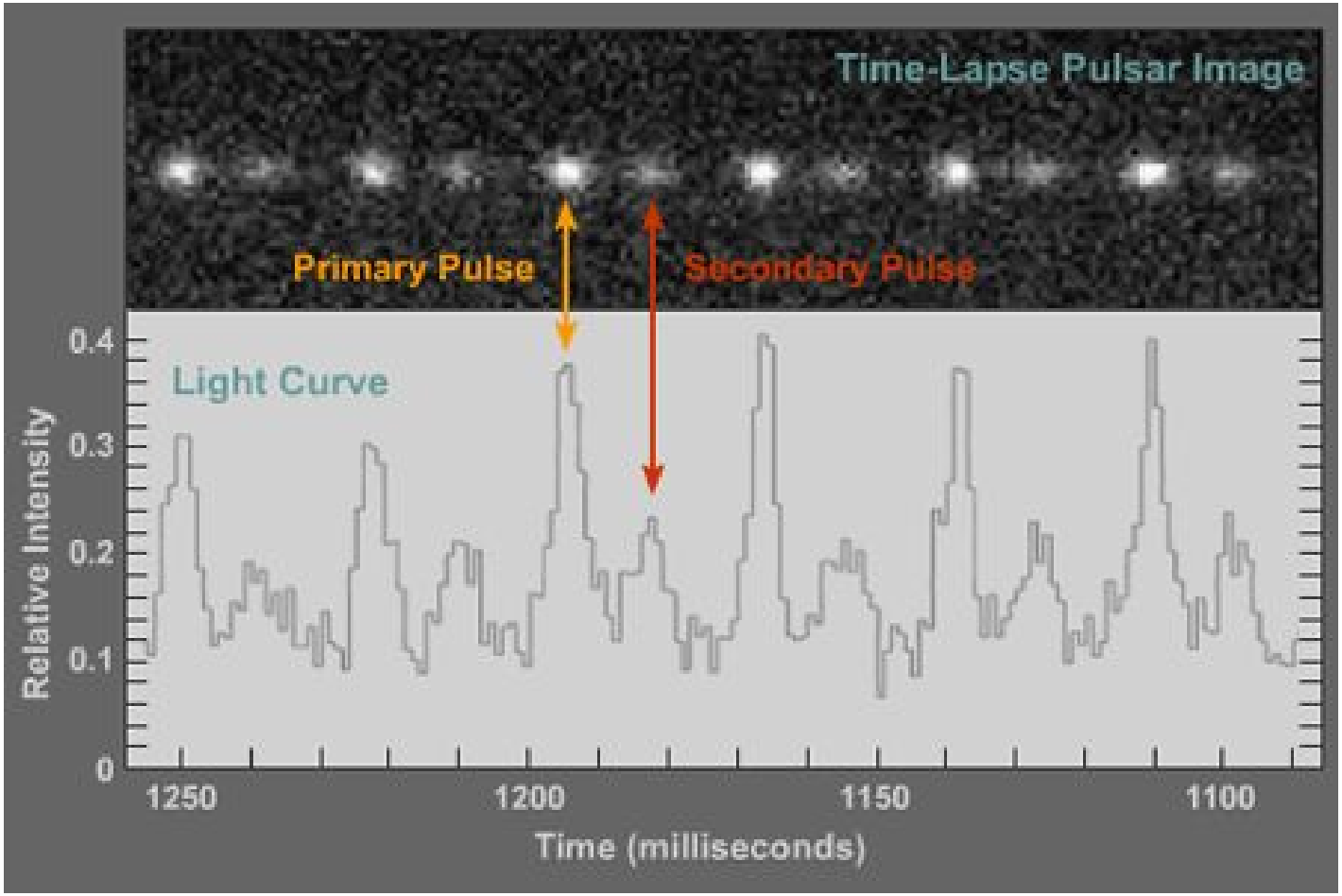


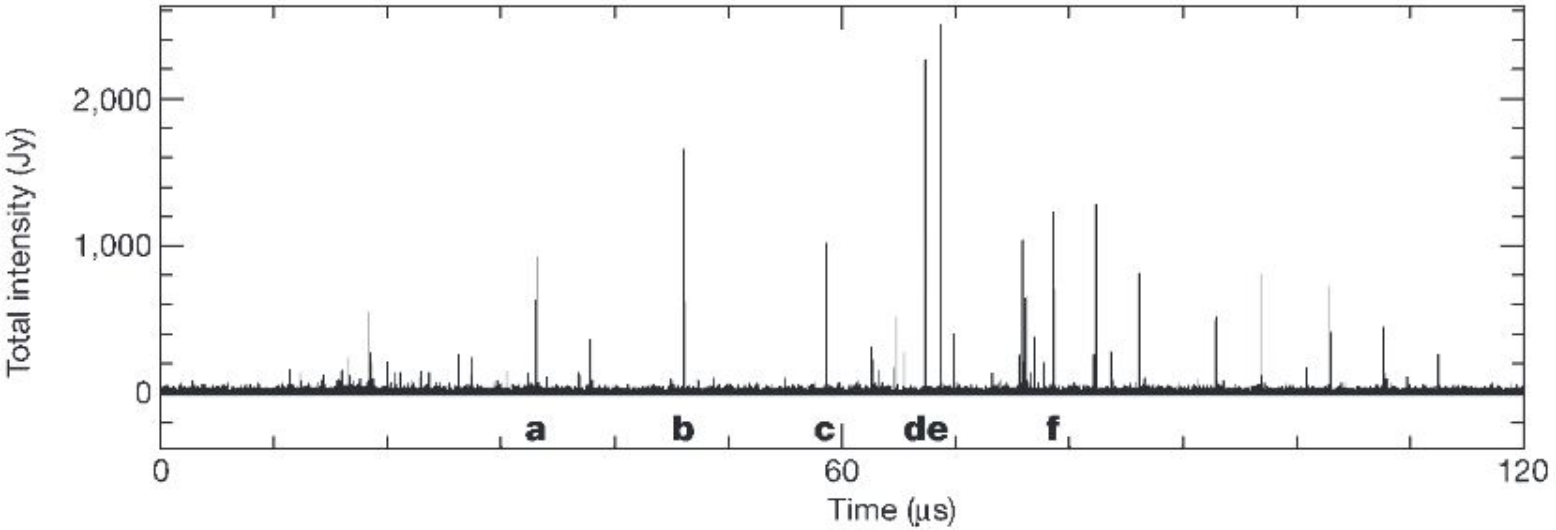
What we're studying here

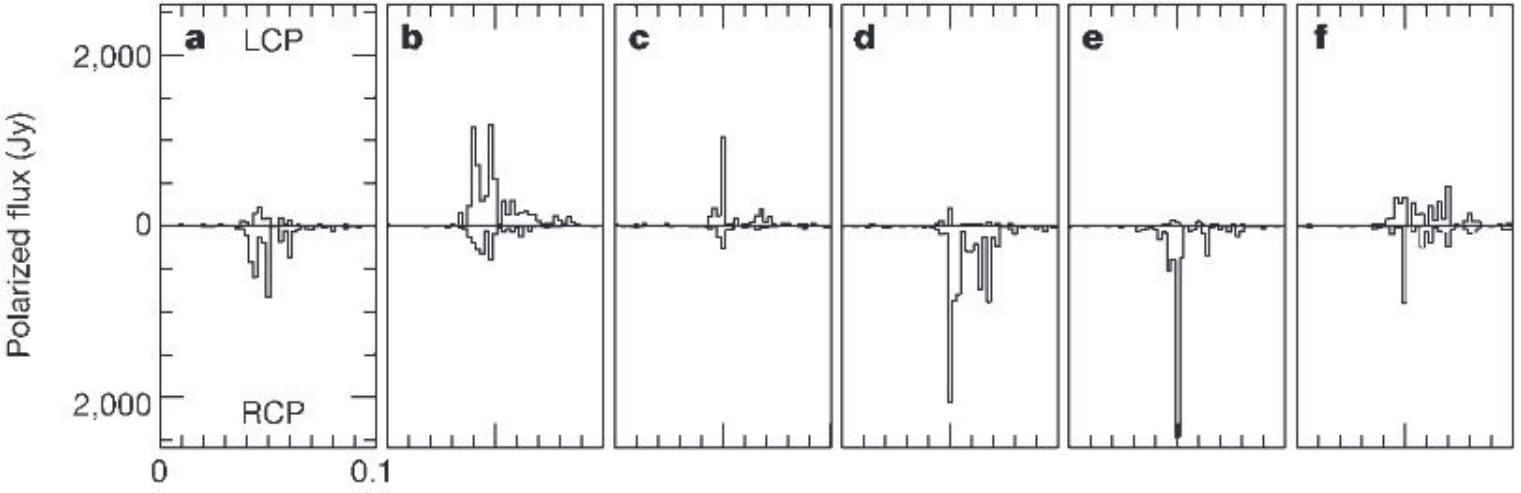
- The Crab Pulsar
  - One of the only neutron pulsars to be identified optically.
  - About 20 Km in diameter.
  - Normal pulses are every 33 milliseconds -- about 30 times a second.
  - Much faster than normal timing observations would capture!






Time-Lapse Pulsar Image
Primary Pulse
Secondary Pulse
Light Curve
Relative Intensity
0.4
0.3
0.2
0.1
0
1250
1200
1150
1100
Time (milliseconds)





Hankins et al.

Nature
13-March
2003

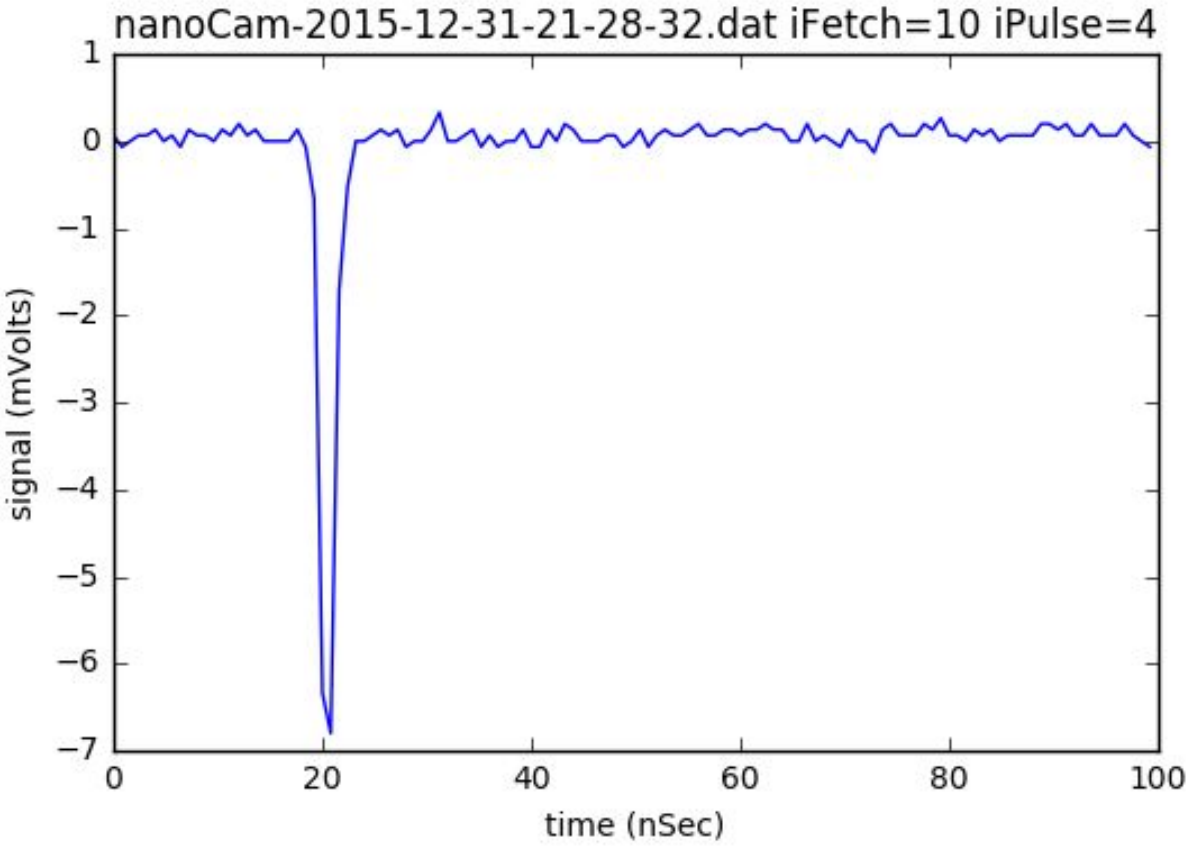
The Goal

- The end goal is to see if the same thing happens with optical photons
- To do this, we are going to use a photon counter attached to a high speed ADC to read out the data (more on that later.)
- Readout should be around 1.5 GSPS.
- This assembly will be tested with a telescope at the Yerkes Observatory.
- If all goes well, we plan to use it on the 200” Hale telescope at the Palomar Observatory.

The Problem

- Wait, 1.5 gigasamples per second?
- Let's see, 1.5 billion 8 bit samples in a second… That's **1.5 gigabytes of data in one second.**
- In under 12 minutes we collect over a terabyte of data.
- A fairly good consumer hard drive runs at just 0.09 gigabytes a second.
- Gigabit ethernet only goes to ⅛ of a gigabyte per second.
- This is way out of the range of anything a computer can process on its own.

One Photon


nanoCam-2015-12-31-21-28-32.dat iFetch=10 iPulse=4
signal (mVolts)
time (nSec)
1
0
−1
−2
−3
−4
−5
−6
−7
0
20
40
60
80
100

The Solution

- From all this data, we only need the parts with the actual pulses.
- Unfortunately, since a computer can't process it fast enough, we need an alternative.
- One solution is to design a custom integrated circuit, or ASIC to do this for us. This is great, but getting it made is expensive and if we make a mistake, there is no fixing it.
- This is comparable to using a GPU instead of a CPU.
- FPGAs are like blank chips that can be programmed with a circuit. They are less expensive than ASICs and can be reprogrammed easily.

A little more detail

Analog signal from telescope

Portion I am working on

Analog to Digital Converter

64 bit bus

An ADC takes the analog signal from the phototube and converts it into digital numbers that can be processed.

```
01101111 01101110
01100101 01110011
00100000 01100001
01101110 01100100
00100000 01111010
01100101 01110010
```

Continued

The FPGA does signal processing for peaks in the signal that show a major pulse. It only sends the peaks to the computer

Ethernet

The computer logs the data and renders it for visualization.

SATA

Chris's universal converter

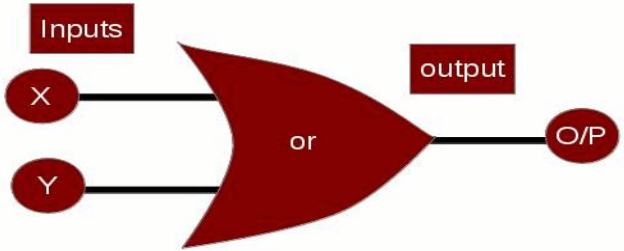
What is a Digital Logic Circuit?

- A digital logic circuit is a circuit that uses electrical signals that are either high or low. The signals are combined with logic gates to produce different outputs based on a given input.

A Quick Example

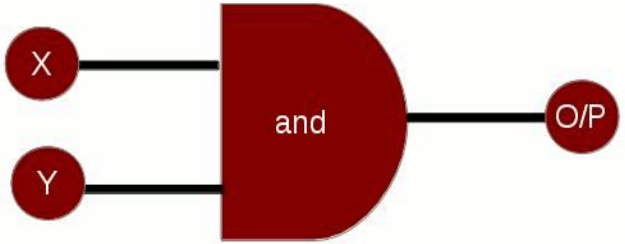
or Gate Animation

X	Y	O/P
0	0	0
0	1	1
1	0	1
1	1	1

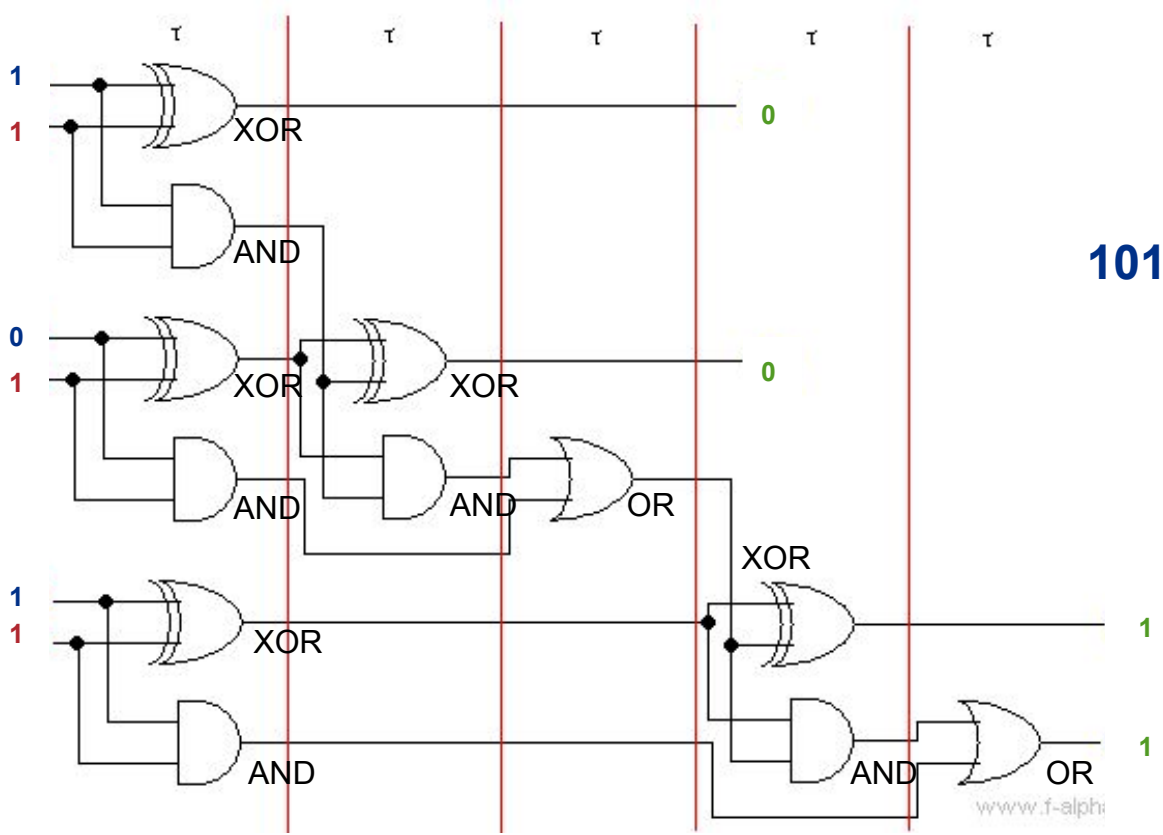


and Gate Animation

X	Y	O/P
0	0	0
0	1	0
1	0	0
1	1	1

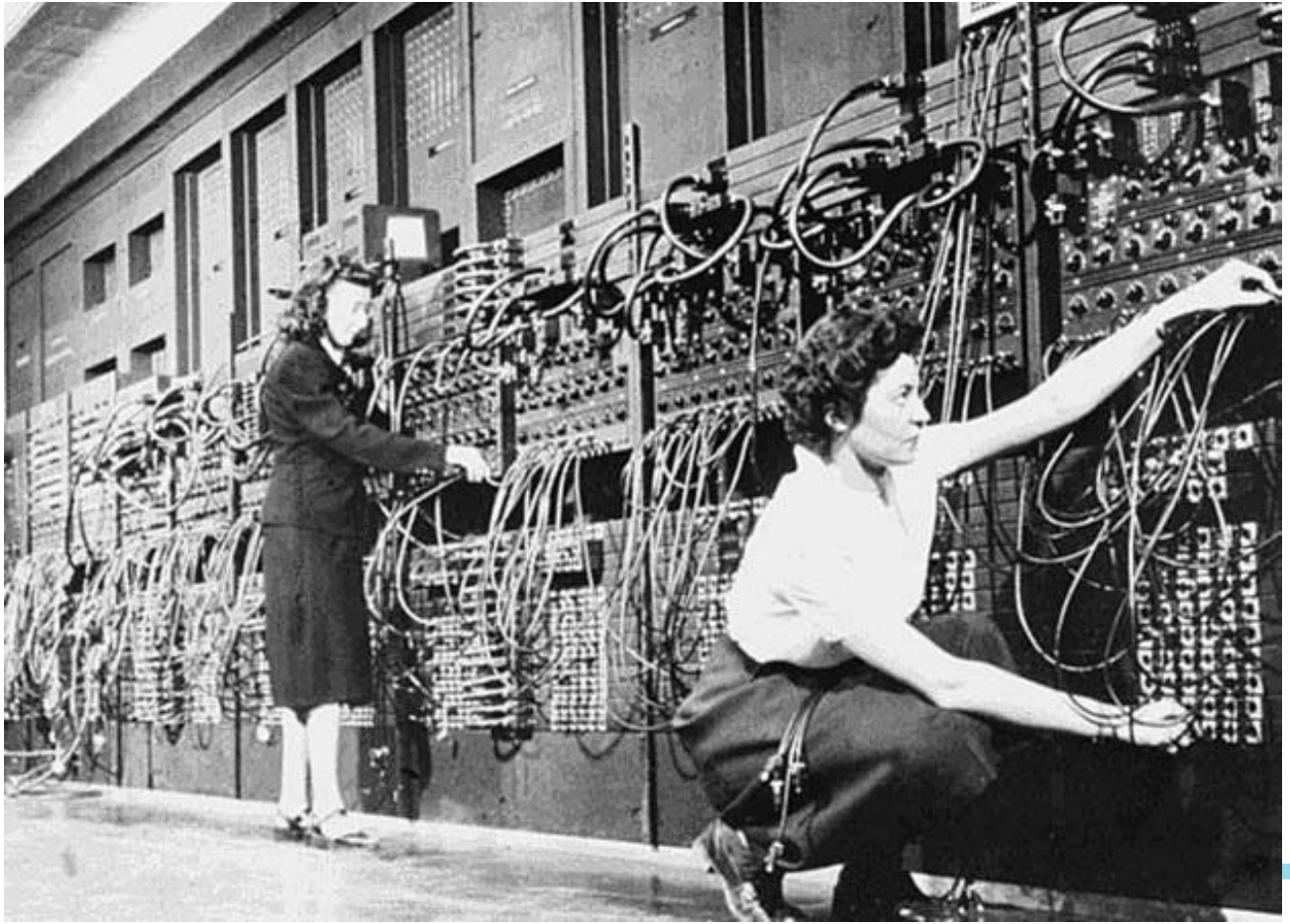


Ripple Carry Adder



101 + 111 = 1100

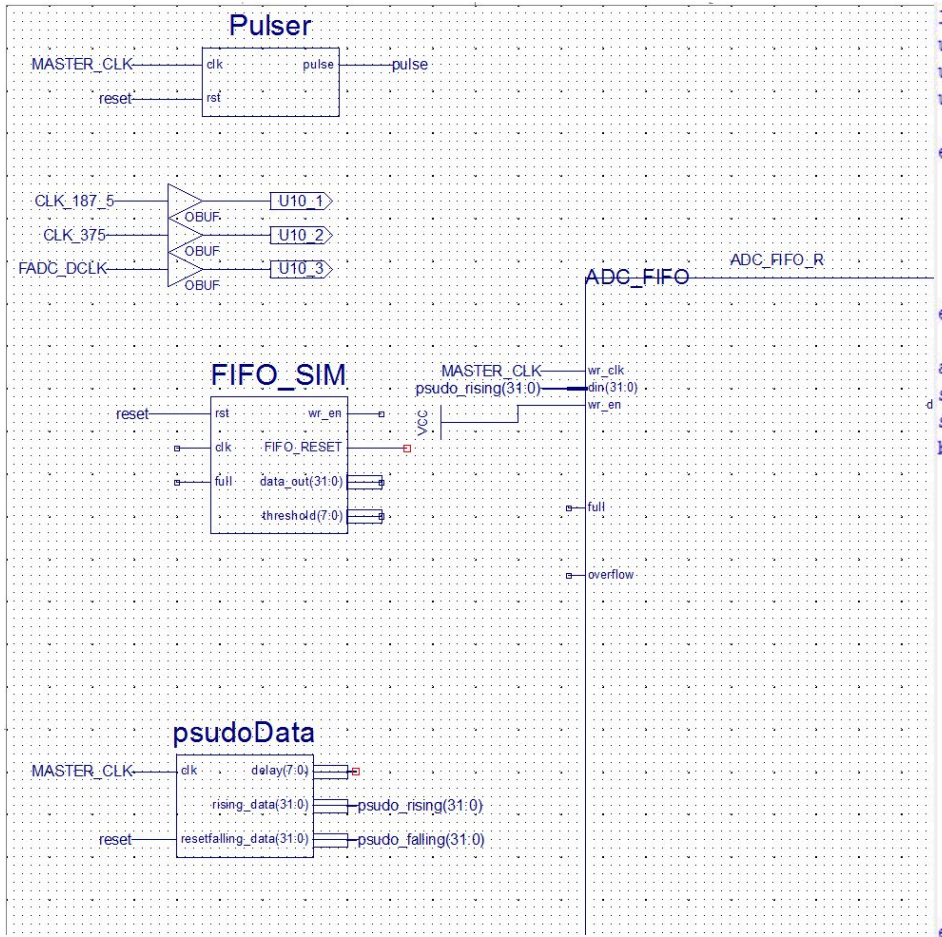
Now make it add two 4-bit numbers



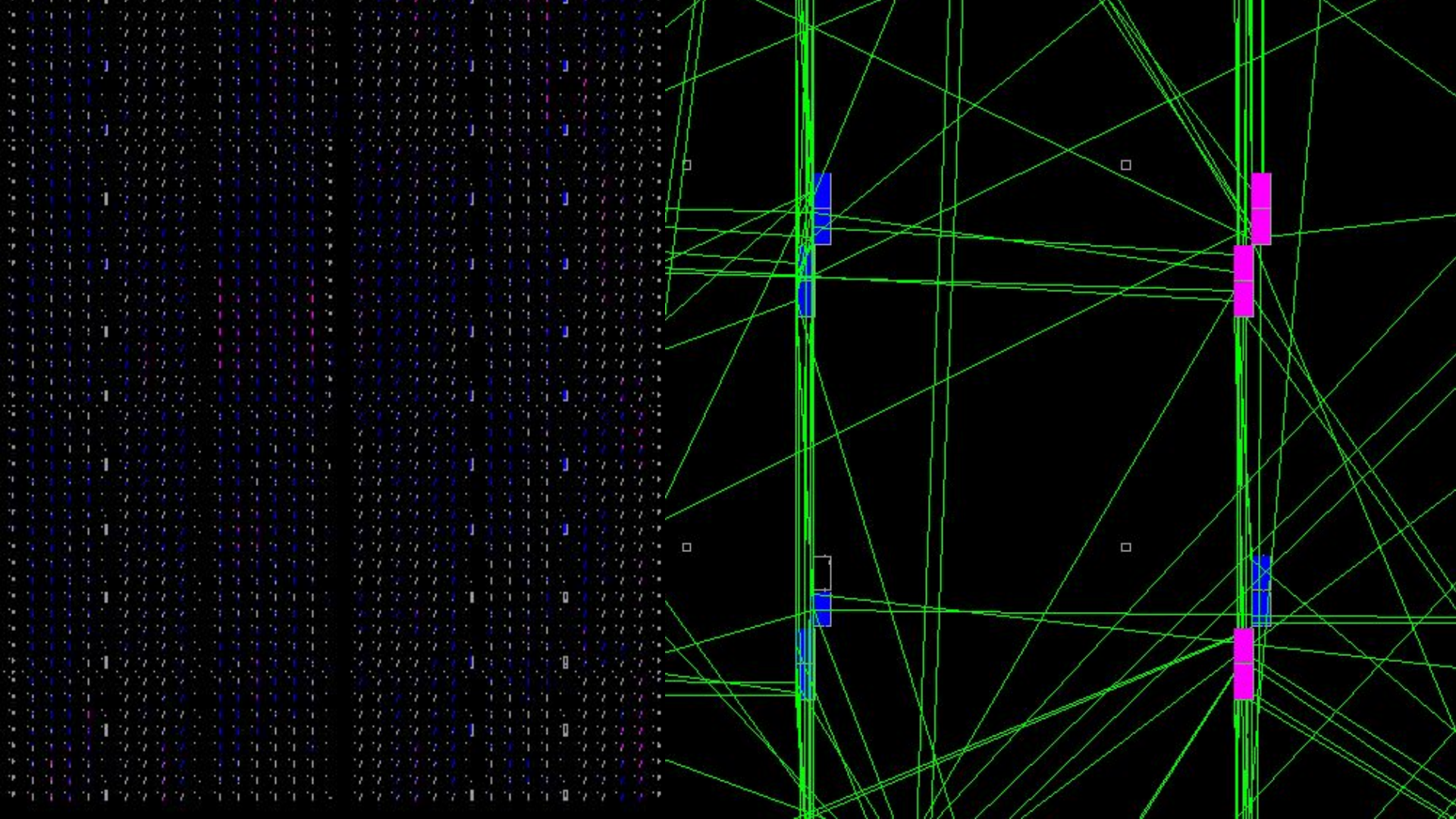
Digital design in the good old days.

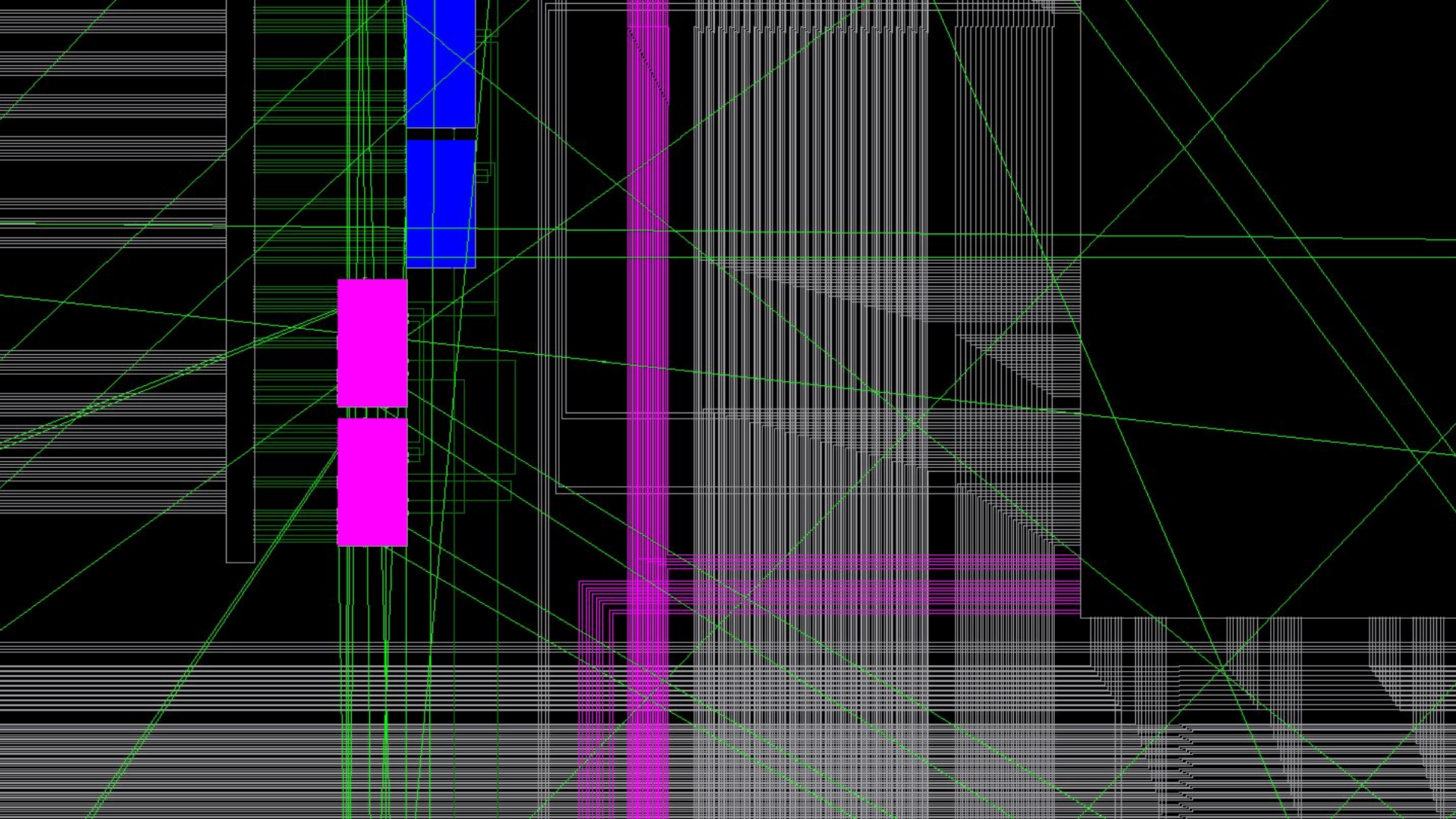
What is an FPGA?

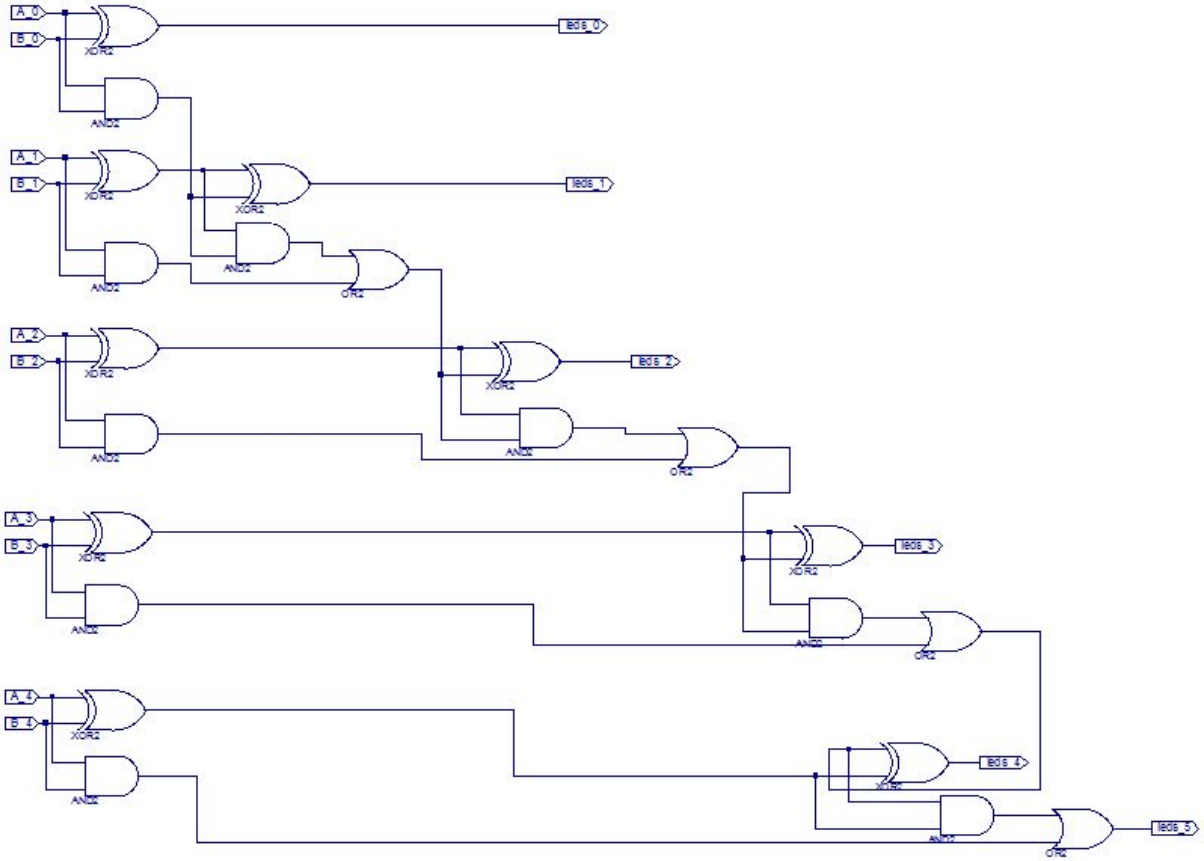
- An FPGA is a collection of configurable logic blocks that can be connected and configured from code.
- The designer makes a digital design on the computer using schematics and special computer languages.
- The design is then compiled and loaded onto flash memory on the board.
- At power up, the FPGA loads the design onto the chip and starts running.
- An FPGA can implement a wide variety of designs. You could implement multiple processors on the same chip (As allowed by the number of resources.)




```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity psudoData is
    Port ( clk : in  STD_LOGIC;
           reset : in  STD_LOGIC;
           delay : out  STD_LOGIC_VECTOR (7 downto 0);
           rising_data : out  STD_LOGIC_VECTOR (31 downto 0);
           falling_data : out  STD_LOGIC_VECTOR (31 downto 0));
end psudoData;

architecture Behavioral of psudoData is
signal counter : unsigned(7 downto 0);
signal counter_f : unsigned(7 downto 0);
begin

   process(clk) begin

      if(reset = '0') then
         if(rising_edge(clk)) then
            counter <= counter + 1;
         end if;

         if(falling_edge(clk)) then
            counter_f <= counter + 1;
         end if;
      else
         counter <= (others => '0');
         counter_f <= (others => '0');
      end if;
   end process;

end Behavioral;
```






A_0
B_0
XOR2
AND2
bb6_0
A_1
B_1
XOR2
XOR2
AND2
AND2
OR2
bb6_1
A_2
B_2
XOR2
XOR2
AND2
AND2
OR2
bb6_2
A_3
B_3
XOR2
XOR2
AND2
AND2
OR2
bb6_3
A_4
B_4
XOR2
AND2
XOR2
AND2
OR2
bb6_4
bb6_5

Pros and Cons of FPGA

- FPGAs are really good at doing multiple things at once.
- They are are really good at signal processing. They can process multiple signals simultaneously very fast.
- Unlike processors, FPGA designs are typically task-specific. This is why we don’t use them as much as processors.

Case Study: BitCoin Mining

ATI Radeon HD 5870

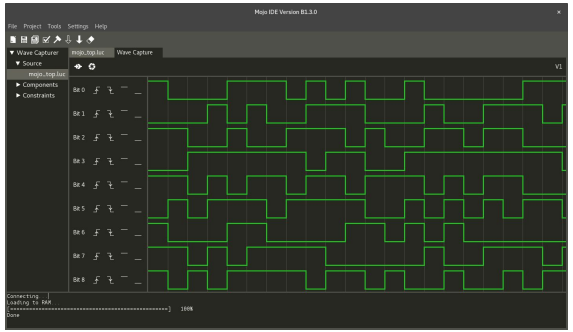
www.weusecoins.com

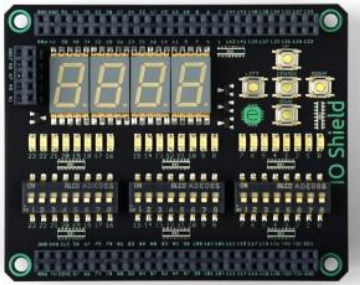
My Project this Summer So Far

- I have been able to write firmware that communicates with the computer to send data out and to allow for input from the computer to set parameters without reprogramming the FPGA.
- Wrote a firmware module that finds peaks based on registers set from the computer through the ethernet connection.
- I am currently in the testing and debugging phase of the project.

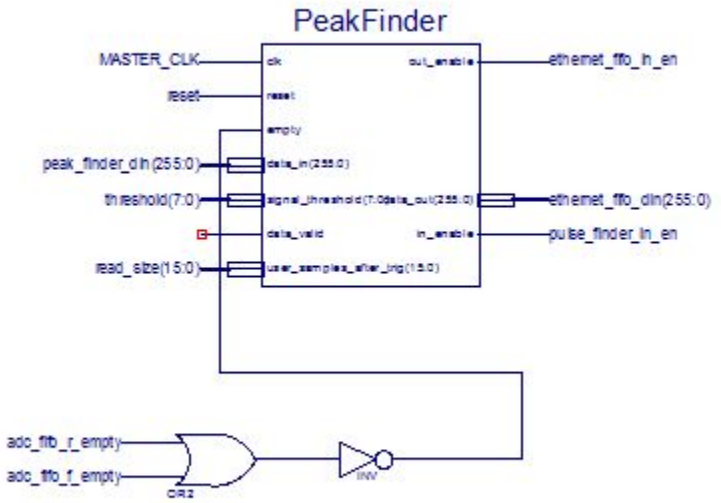
A Bit About the Demo Board

- Mojo Development Environment
- I consider this the “Arduino” of FPGAs.
- Full software package with an IDE and logic analyzer.
- Done over USB so you don’t need an expensive cable.
- Tutorials can be found online for free.

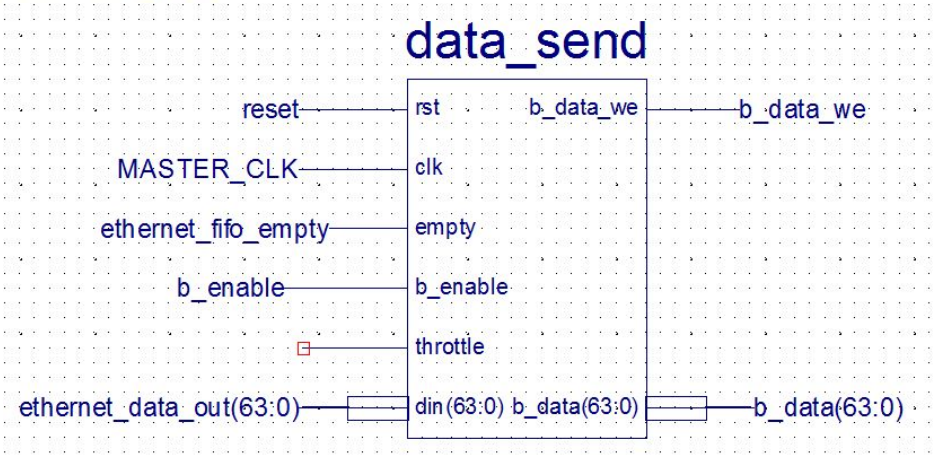




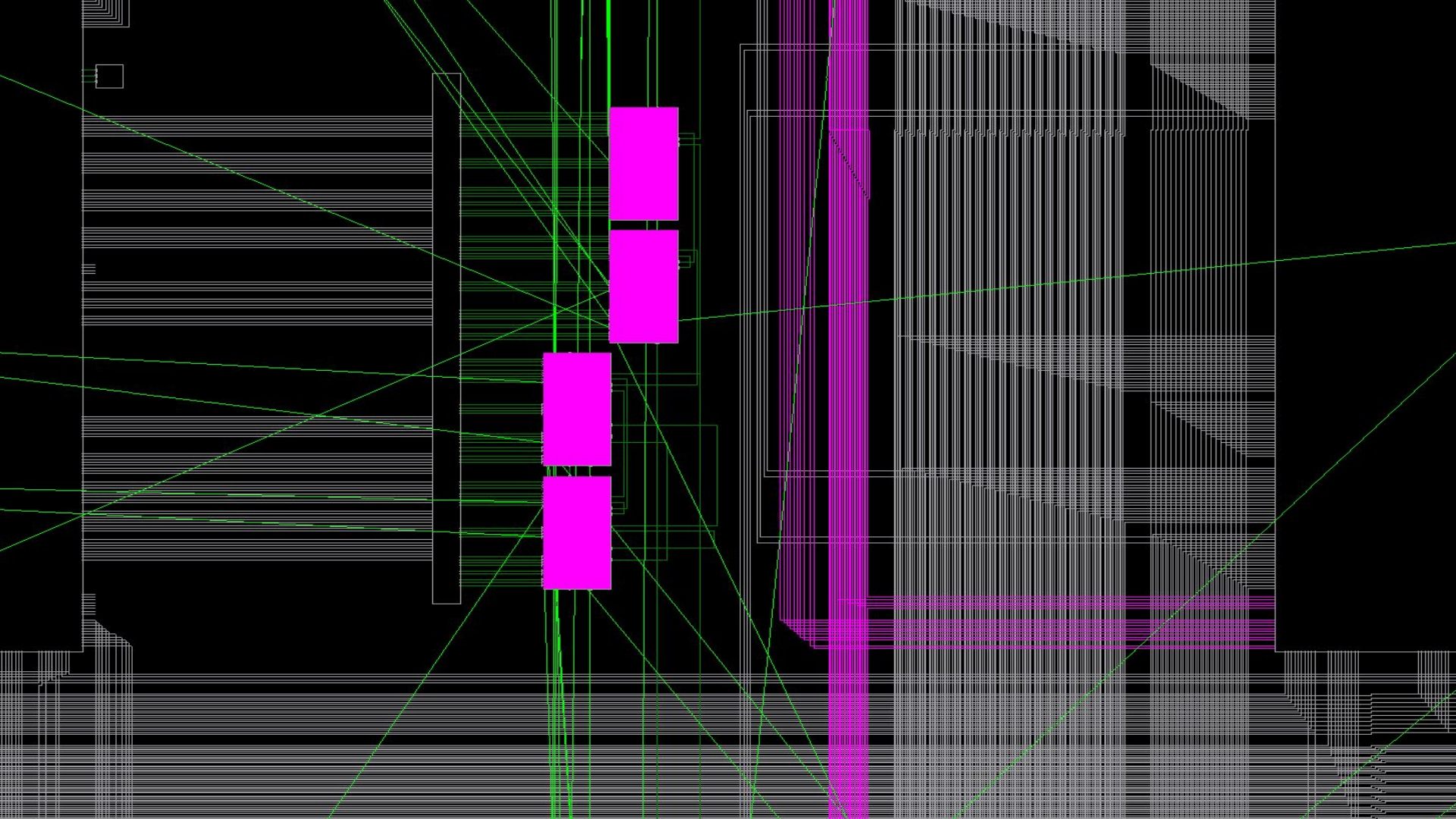
A Few Thousand Words




```
entity PeakFinder is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           data_in : in  STD_LOGIC_VECTOR (255 downto 0);
           signal_threshold : in STD_LOGIC_VECTOR(7 downto 0);
           user_samples_after_trig : in std_logic_vector(15 downto 0);
           empty : in  STD_LOGIC;
           data_valid : in std_logic;
           data_out : out  STD_LOGIC_VECTOR (255 downto 0);
           out_enable : out  STD_LOGIC;
           in_enable : out std_logic);
end PeakFinder;
```



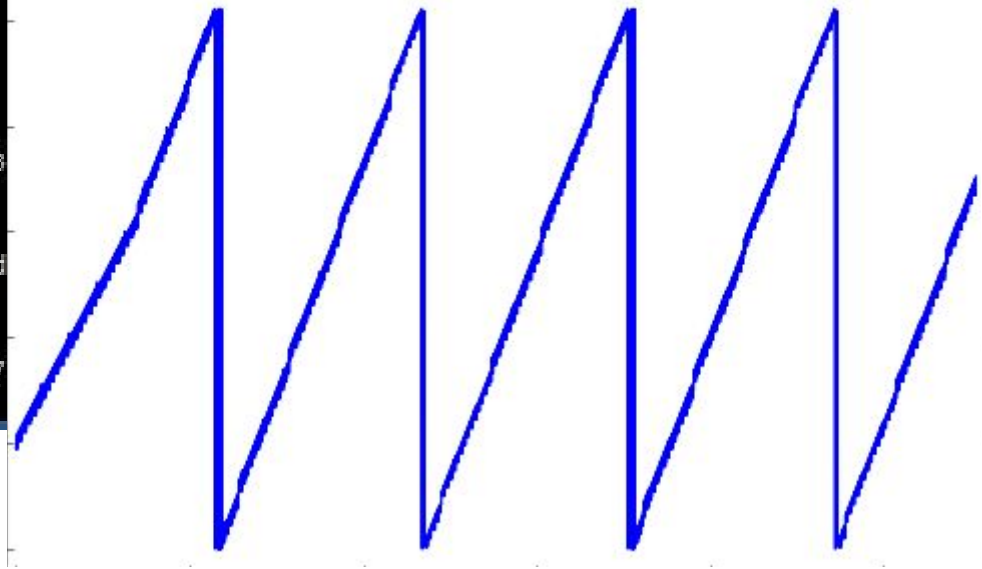

```
process(clk, b_enable, throttle)
begin
   if(rst = '0') then
      if(empty = '0') then
         if(rising_edge(clk)) then
            b_data_we <= '0';

            count_delay <= count_delay + 1;
            if(count_delay = 5) then
               count_delay <= (others => '0');
               b_data_we <= '1';
            end if;
         end if;
      end if;
   else--reset code here
   end if;
end process;
```




/ethernet_overflow 1 1
/b_data_we 1 1
/ethernet_fifo_empty 0 0
/psudo_falling AAAAAAAA AAAAAAAA
/psudo_rising A9A9A9A9 A9A9A9A9
/psudo_rising<0> 1 1
/psudo_rising<1> 0 0
/psudo_rising<2> 0 0
/psudo_rising<3> 1 1
/psudo_rising<4> 0 0
/psudo_rising<5> 1 1
/psudo_rising<6> 0 0
/psudo_rising<7> 1 1
/psudo_rising<8> 1 1
/psudo_rising<9> 0 0
/psudo_rising<10> 0 0
/psudo_rising<11> 1 1
/psudo_rising<12> 0 0
/psudo_rising<13> 1 1
/psudo_rising<14> 0 0
/psudo_rising<15> 1 1

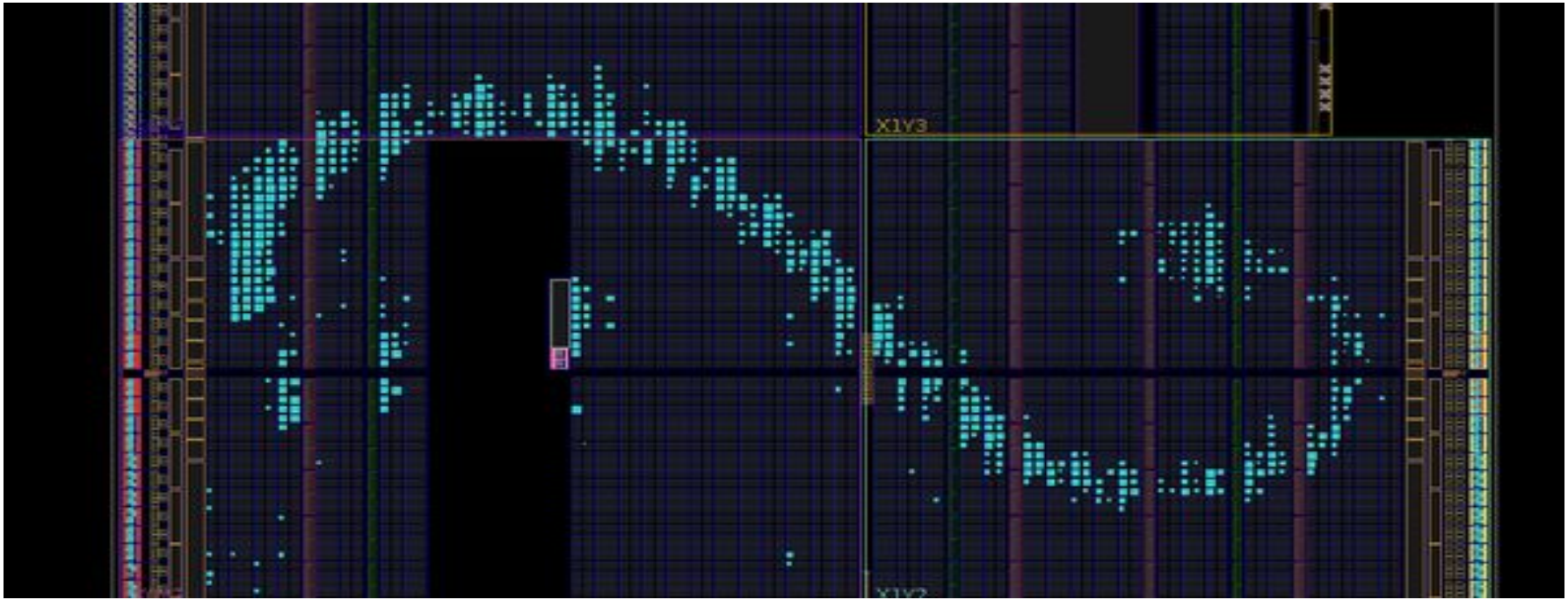
```
Creating Socket:
UDP target IP: 192.168.133
UDP target port: 2001
Send one byte?
Command Completed.
Set burst mode?
Command Completed.
Reset write block?
Command Completed.
Erase write block?
Command Completed.
Request Data?  d
Command Completed.
192.168.133.2  2001 10 0x1 0x2 0xffffffffffffffffL
192.168.133.2  2001 1458 0x2 0x3 0x3173217311730173
data valid
1
192.168.133.2  2001 1458 0x2 0x4 0x30bd20bd10bd00bd
data valid
2
192.168.133.2  2001 1458 0x2 0x5 0x3007200710070007
data valid
```

248	28.358413	192.168.133.2	192.168.133.199	UDP	1500	2001 → 49387 Len=1458
249	28.358414	192.168.133.2	192.168.133.199	UDP	1500	2001 → 49387 Len=1458
250	28.358418	192.168.133.2	192.168.133.199	UDP	1500	2001 → 49387 Len=1458
251	28.358608	192.168.133.2	192.168.133.199	UDP	1500	2001 → 49387 Len=1458
252	28.358610	192.168.133.2	192.168.133.199	UDP	1500	2001 → 49387 Len=1458



```
10111001 11101010 01100011 00101010 10001000 11001100 00000010 00000111
00000100 00000000 00100110 10111001 11101010 01100011 00101010 00000100
00000111 00000011 00000000 00100110 10111001 11101010 01100011 00101010
00000110 00000010 00001110 00010001 11111110 00001001 00000000 00010010
00001111 00000001 00000011 00000000 00000001 00000000 00000000 11111110
00000111 00000000 00010010 10111011 00000001 00000000 00000001 00000001
00000000 00000000
```

Fermilab



Questions?

Special Thanks To

Dr. Chris Stoughton

Ryan Rivera

George Dzuricsko

Laura Thorpe

The Quarknet Program

The Quarknet Students

The Holometer Team