EGR 433: Advanced Computer Engineering

# Vector Array / Neuron Processor Design

and other design projects


Advisor **J T Wunderlich** PhD

**Clay Buxton** '20 Computer Engineering

**Kevin Carman** '20 Computer Engineering

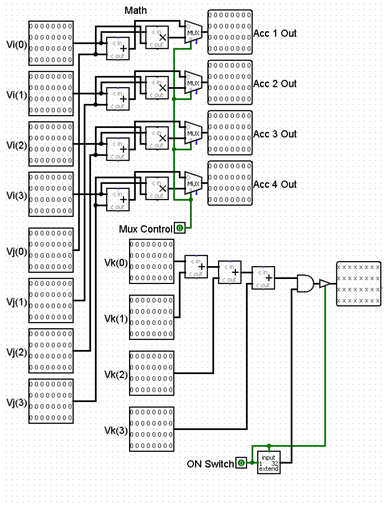
**Derek Manning** '19 Computer Engineering


## Instructions

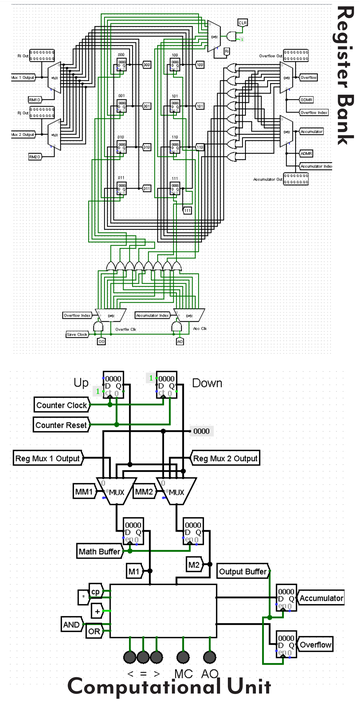
| | |
|---|---|
| 00h | Ri + Counter #1 -> Rk |
| 01h | Ri + Counter #2 ->Rk |
| 02h | Ri + Rj -> Rk |
| 03h | Counter #1 + Counter#2 -> Rk |
| 08h | Ri x Counter #1 -> Rk, Overflow-> Rk+1 |
| 09h | Ri x Counter #2 -> Rk, Overflow -> Rk+1 |
| 0Ah | Ri x Rk -> Rk, Overflow -> Rk+1 |
| 0Bh | Counter #1 x Counter #2 -> Rk, Ovr Rk+1 |
| 10h | Compare Ri with Counter #1 -> Rk +LED |
| 11h | Compare Ri with Counter #2 -> Rk + LED |
| 12h | Compare Ri with Rj -> Rk + LED |
| 13h | Compare Counter #1 with #2 -> Rk + LED |
| 20h | Ri AND Counter #1 -> Rk |
| 21h | Ri AND Counter #2 -> Rk |
| 22h | Ri AND Rj -> Rk |
| 23h | Counter #1 AND Counter #2 -> Rk |
| 24h | Ri OR Counter #1 -> Rk |
| 25h | Ri OR Counter #2 -> Rk |
| 26h | Ri OR Rj -> Rk |
| 27h | Counter#1 OR Counter#2 -> Counter #2 |
| 30h | Clear Ri |
| 40h | MAC<br>Accumulator -> Rk+3, Wrap R0+<br>Overflow -> Rk+4, Wrap R0+<br>Ri x Rj -> Rk, Overflow Rk+1, Wrap R0+<br>(Rk+3) + Rk -> Rk<br>(Rk +4) + (Rk+1) + Carry -> Rk+1 |

## Vector + Neuron Instructions

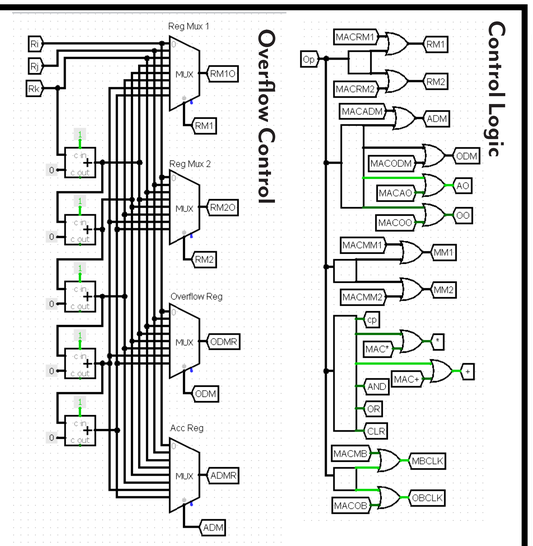
| | |
|---|---|
| 20h | Vi + Vj -> Vk |
| 21h | Vi + Vj -> Vk, Overflow Vk+1 (Wrap R0) |
| 22h | Vi • Vj -> Vk, Overflow Vk+1 (Wrap R0)<br>Vk(1)+Vk(2)+Vk(3)+Vk(4) -> 32 Bit Accumulator |
| 23h | Vi x Vj -> Vk, Overflow -> Vk+1 (Wrap R0)<br>Vk(1)+Vk(2)+Vk(3)+Vk(4) -> 32 Bit Accumulator<br>32 Bit Accumulator -> Neuron Transfer Function |




Vector Math and Accumlator




Computational Unit







The machine have 4 of these cores which process a code stack. Each have the ability to run all of the 26 uniqure functions, and has 6 registers to store information needed for later instructions.

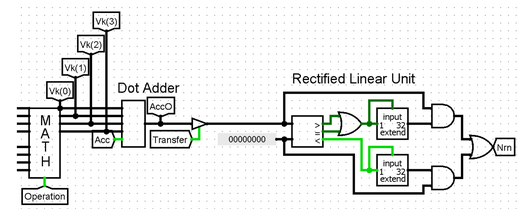
## Core(x4)

## Project Goals

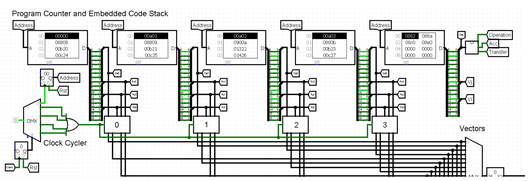
Use Logisim to create a quad core vector array and neuron processor with an embdedded code stack. The processor controlled by a program counter with a master control unit and a finite state machines that implements the simple pipeline of **fetch, decode, execute,** and **write-back** plus any special states.

Our machine needed to be able to process a set of basic instructions in addition to vector mathematics and a neuron transfer function. The computer has 6 registers that act as memory for the machine. Using the inputs Ri and Rj, in addition to 2 counters, the machine can do basic mathematics and logic. This machine also has the ability to do math with vectors using the vector registers Vi and Vj.

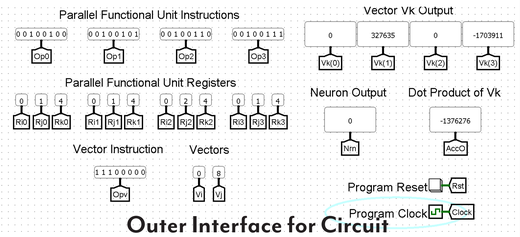
Our machine has 26 unique instructions that can be programmed into an embedded code stack to allow for autonomous execution using a finite state machine. Once the machine has been programmed through the stack, the machine can be set to run and autonomously run through the program as expected from any other computer.




Vector Math and Neuron Transfer




Code Stack and Cores




Outer Interface for Circuit

## Feild Programmable Gate Arrays

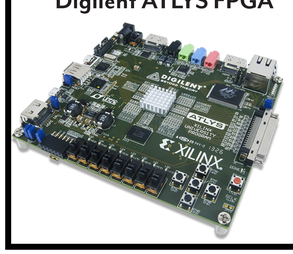
Using ATLYS FPGA's, we were able to implement simplistic variants of the cores. Ultizing the ISE Suite, we designed the ciruicts required for these cores and physically interacted with them using the ATLYS FPGA's. This allowed us to have hands on interaction with our circuit.

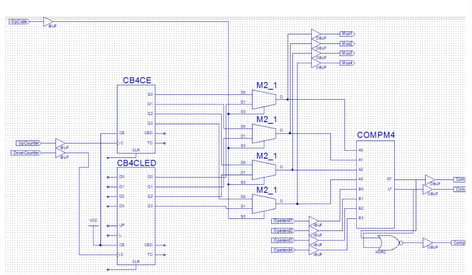
DIGILENT A National Instruments Company

SPARTAN 7

XILINX

Digilent ATLYS FPGA



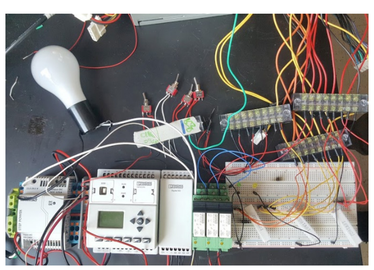



## Industrial Logic Controllers

PHOENIX CONTACT

Using both the Advanced AXC PLCs, with PC Worx, and the Nano Line PLCs, with Nano Navigator Suite, we created circuits that interfaced with the real world.

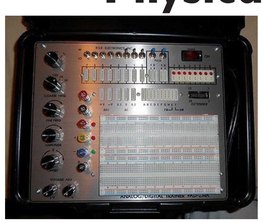
Shown to the left is a circuit that uses an ATX power SUPPLY and A Nano PLC to turn on A lightbulb in differed ways. Using the Advanced AXC PLCs, we did labs we did labs using circuitry similar to logisim and ISE and ladder logic.



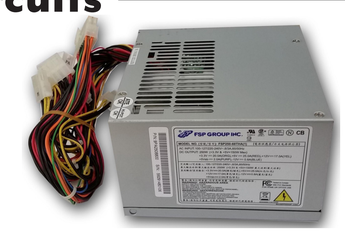
Nano Line PLC

Advanced AXC PLC

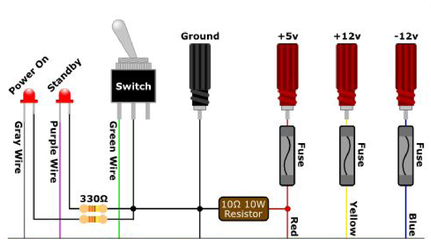
## Physical Circuits



Circuit Trainer



ATX Power Supply




Our ealier labs had us implementing simple parts of the cores in a physical model. These circuits had counters and could do basic math. In a later lab, we used an ATX power supply to power portions of the NanoPLC. We found out the hard way that these ATX power supplies have to have a dummy load to work correclty. That circuit is shown to the left.