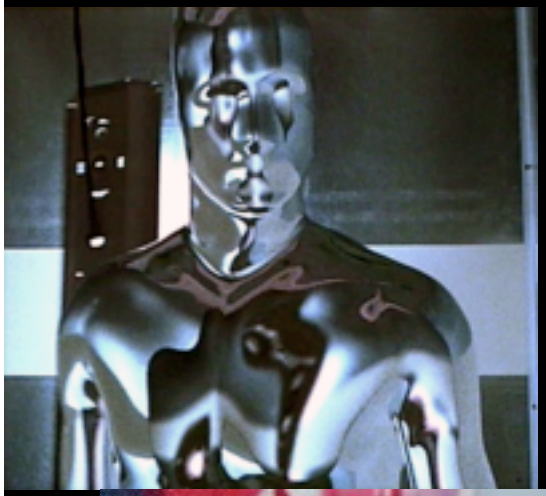
# Reconfigurable Computing

Rafael Arce Nazario
Computer Science Department
UPR – Río Piedras

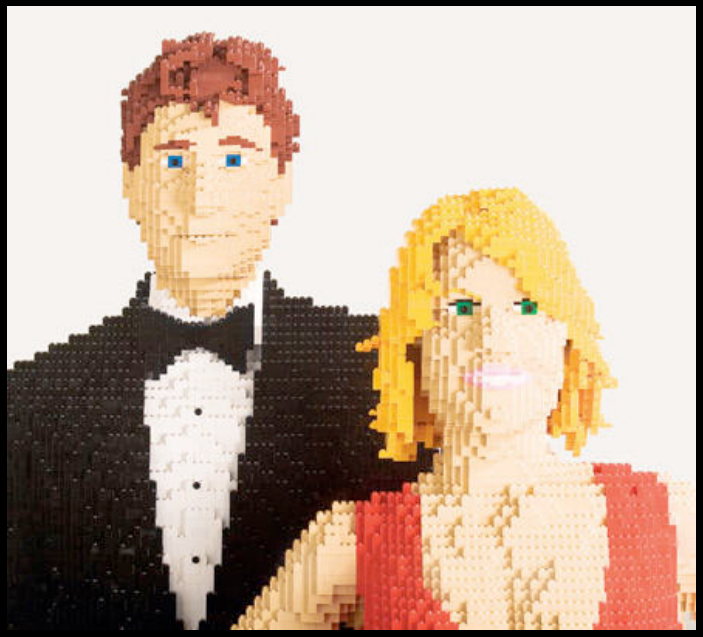
What does the word reconfigurable remind you of?

# A Transformer?

# T-1000 from Terminator 2?

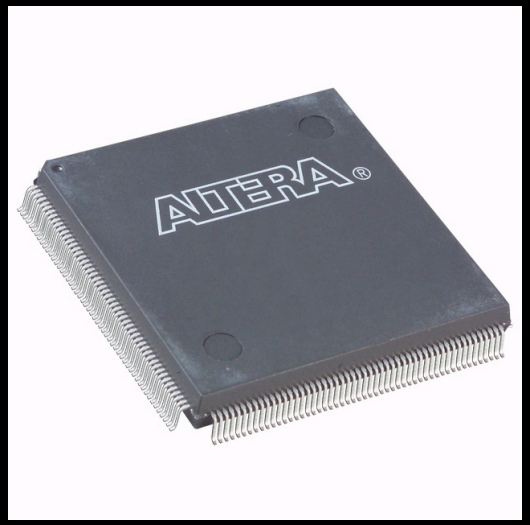



# Lego?



# In the context of computation circuits

- Reconfigurability = capacity of a circuit TO change the configuration and connections of its internal components to achieve different computational tasks.

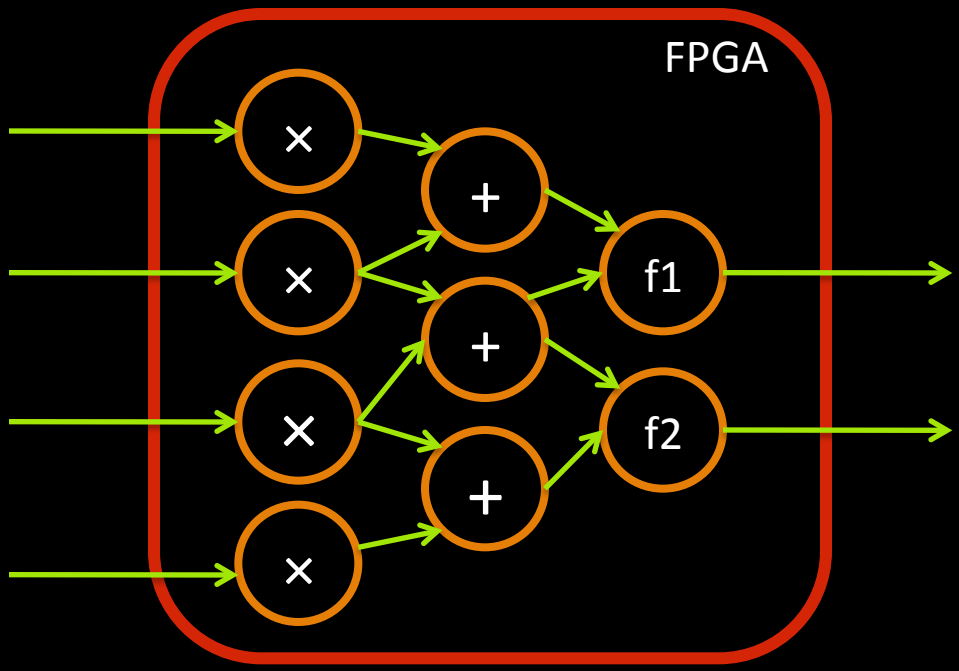



FPGA
Field Programmable
Gate Array

On the outside it looks
like a common chip….

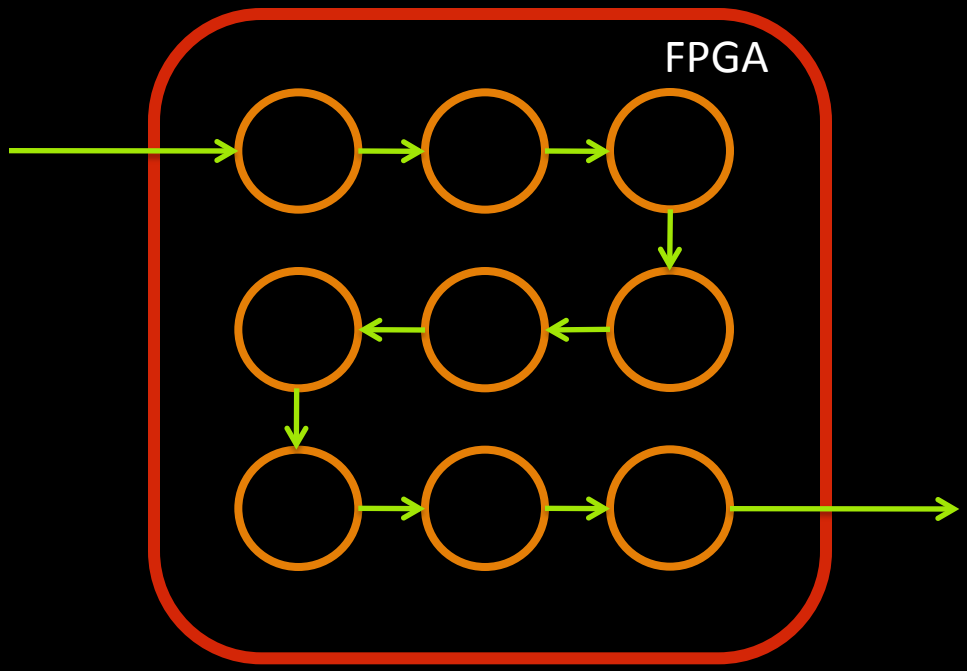
# Inside . . .

- The necessary circuitry to become …



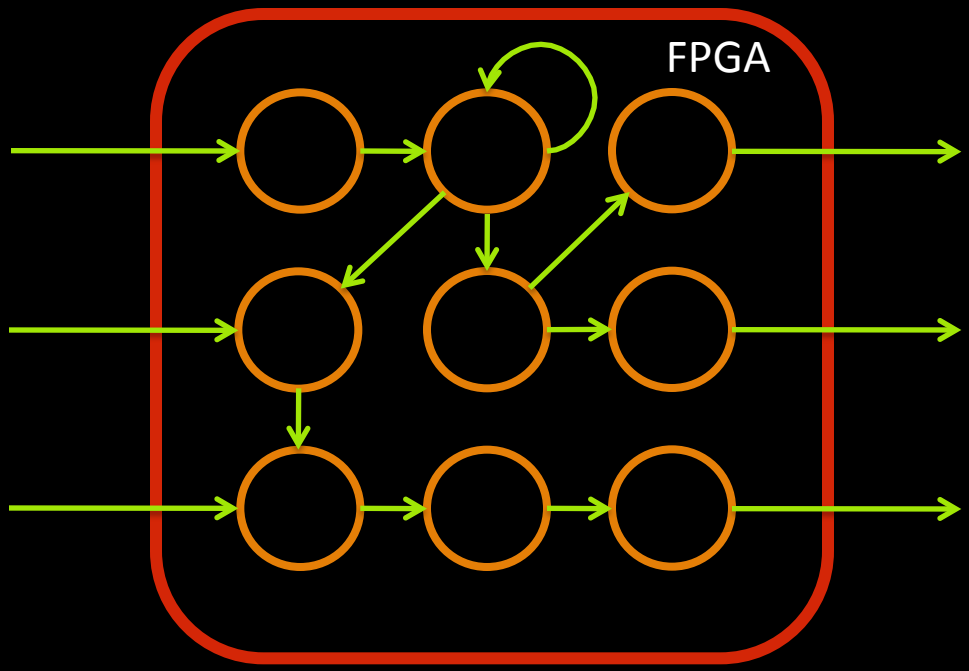
# Inside . . .

- or this …



# Inside . . .

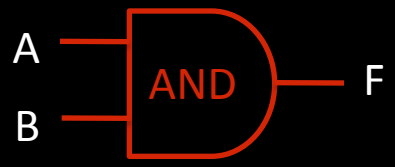
- or even this …



# How can it do that?

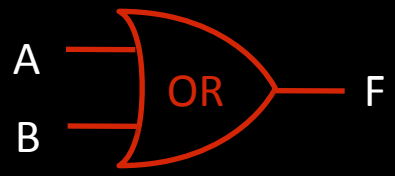
- Electronic computers use electrical signals to compute.
- During computation each signal can assume one of two values: 0 or 1 (signals are binary)
- Boolean gates receive one or more binary input signal(s) and 'compute' output signal(s).

An example of a Boolean gate:



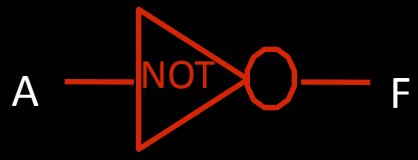

| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Other basic Boolean gates




| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |


| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



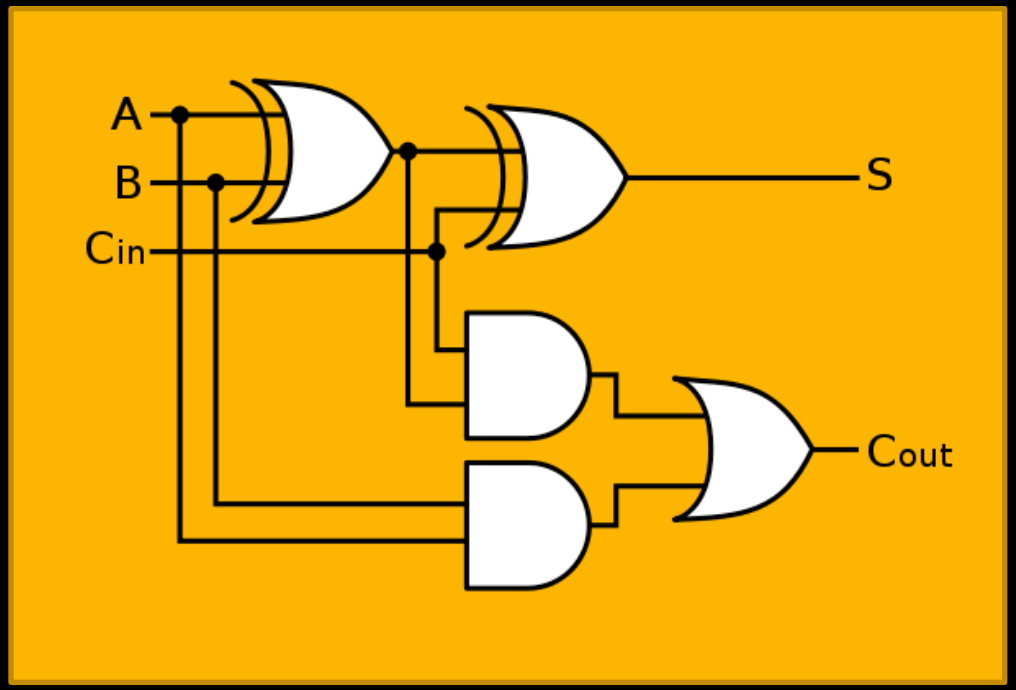

| A | F |
|---|---|
| 0 | 1 |
| 1 | 0 |

- The amazing thing is: by combining these simple gates we can accomplish any operation (arithmetic, etc.)

# Full adder

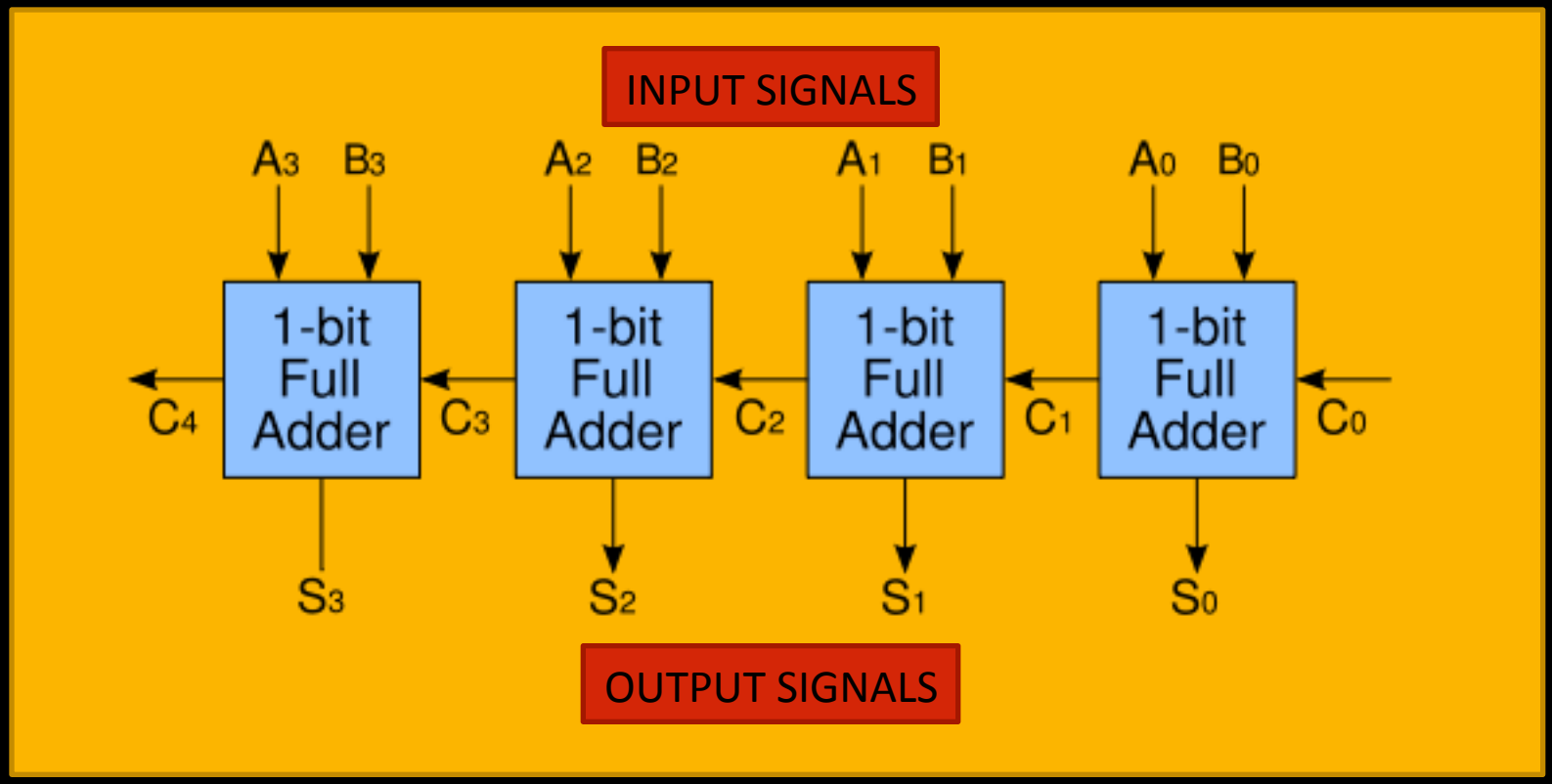
- Sums any two bits A and B and Carry bits

FULL ADDER




| A | B | Cin | Cout | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

# 4-bit adder




Basically, a chain of full adders

# By further combining Boolean gates we can accomplish…

- Multipliers
- Square root operators
- Comparisons
- Decision/Control Structures
- and lots more

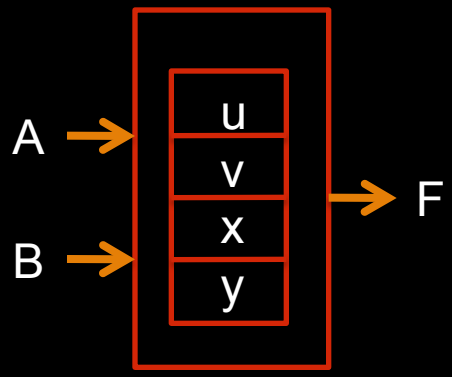
Intel i5 contains 100's of millions of gates inside

- They are the basis of all computational circuits: microprocessors, dedicated processing circuits, etc.

Ross Freeman:
Inventor of the FPGA

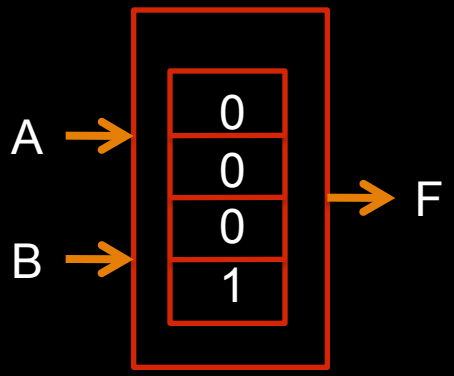
Hey, I know a circuit that can behave as an AND, OR, NOT, XOR, gate depending on how I 'program' it. Its called a look up table.

Rafael Arce
Mortal apprentice

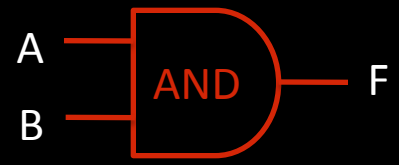


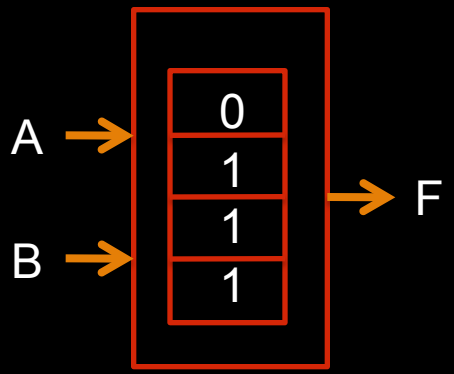

| A | B | F |
|---|---|---|
| 0 | 0 | u |
| 0 | 1 | v |
| 1 | 0 | x |
| 1 | 1 | y |

# The fabulous look-up table

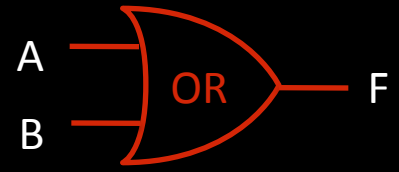



| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

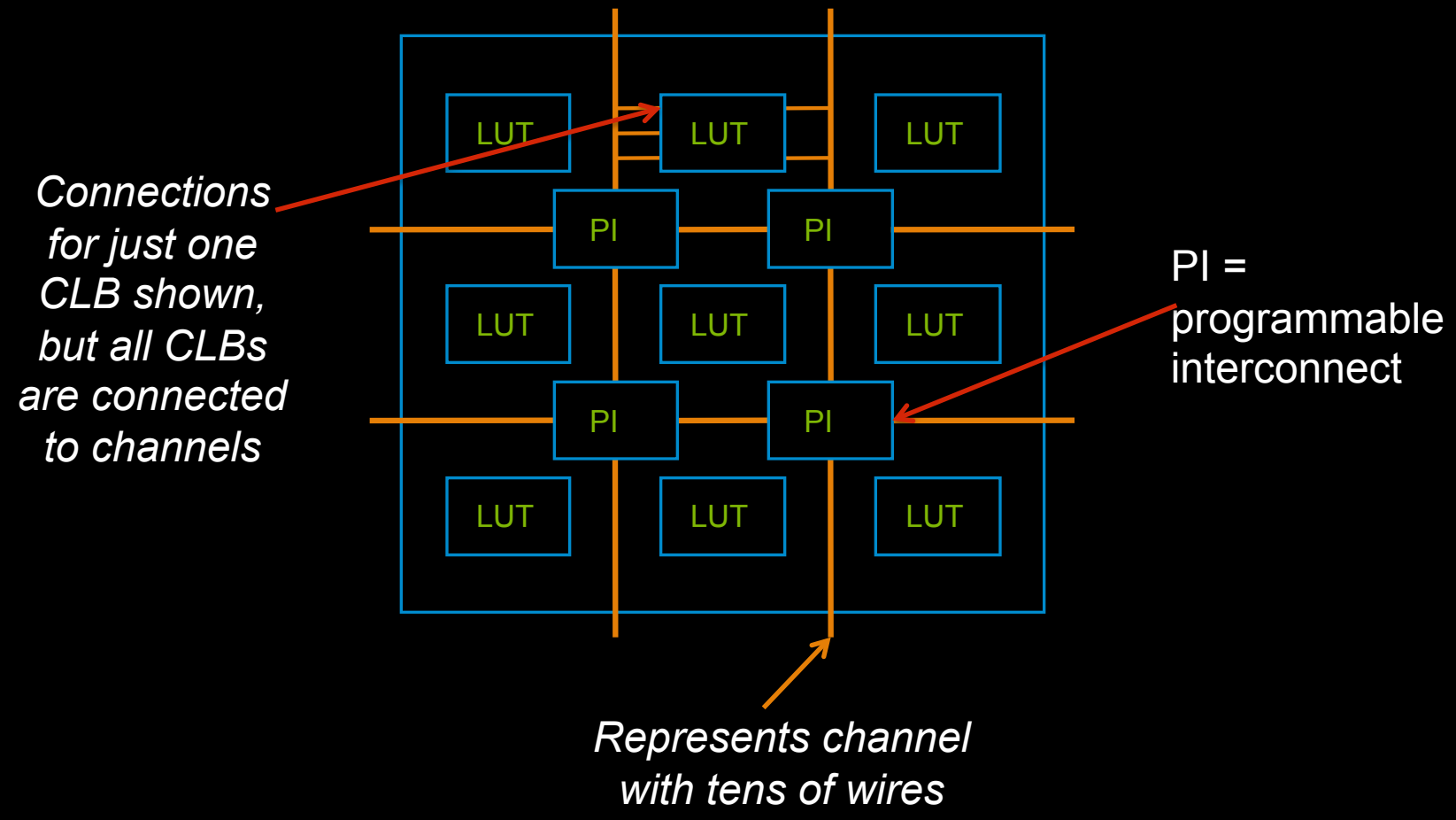



| A | B | F |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |



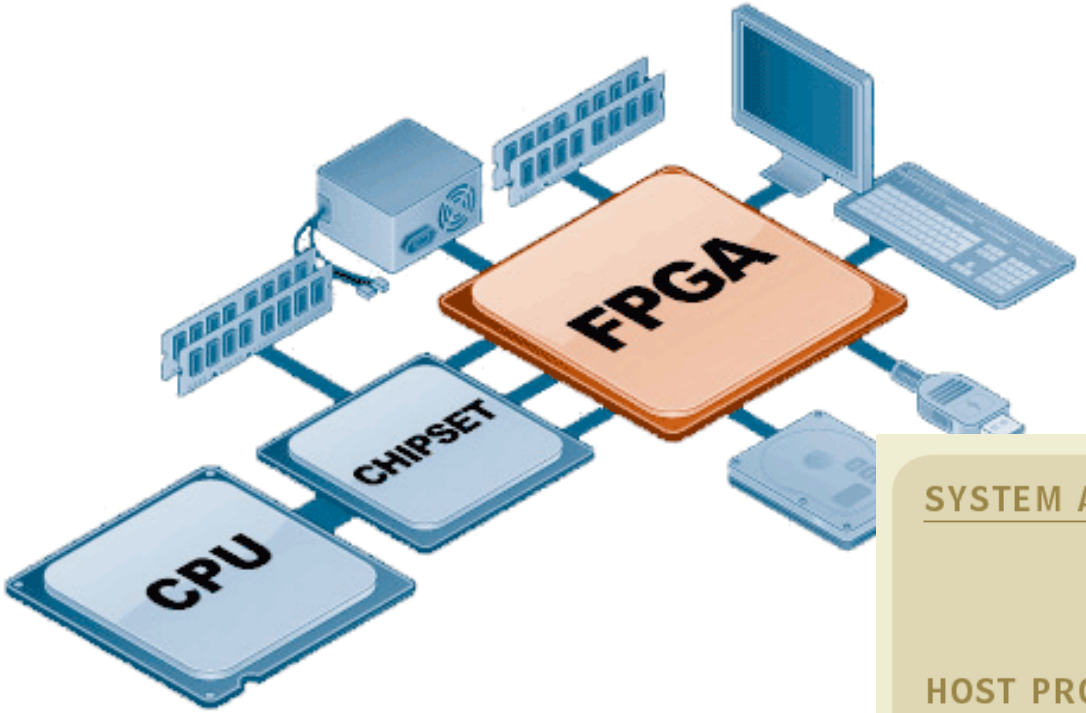
# LUTs and Interconnections



# What FPGAs are good for..

- Application specific data types:
  - Genomic data
  - Encryption
  - Network packets
  - Integer data
- Not so good for:
  - Floating point data, control intensive (sequential processing)

# Reconfigurable High Performance Computers





## SYSTEM ARCHITECTURE

| | |
|---|---|
| | Dual-socket computer with one Intel® host processor and one Convey HC-1 custom coprocessor |

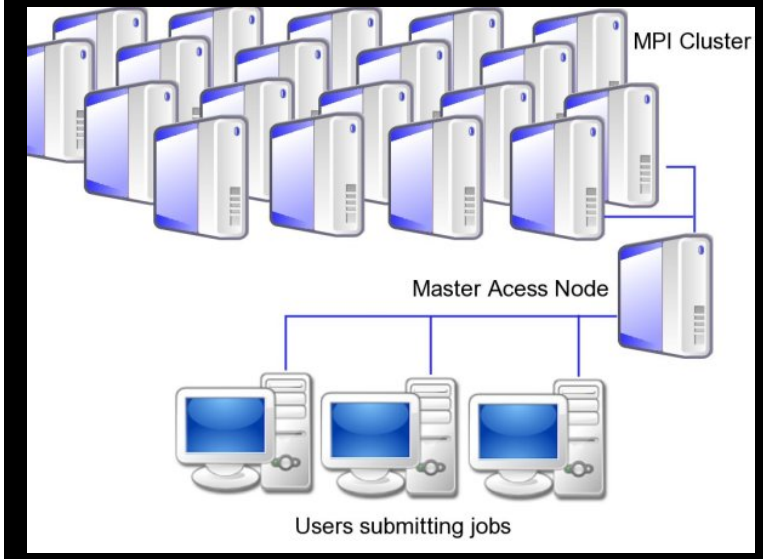
## HOST PROCESSOR

| | |
|---|---|
| PROCESSOR | Intel® Xeon® 2.13 GHz dual-core processor |
| HOST MEMORY | DIMMs: minimum 4, maximum 16<br>Max memory: 128GB |
| CHIPSET | Intel® 5400 MCH (1066 MHz FSB) |

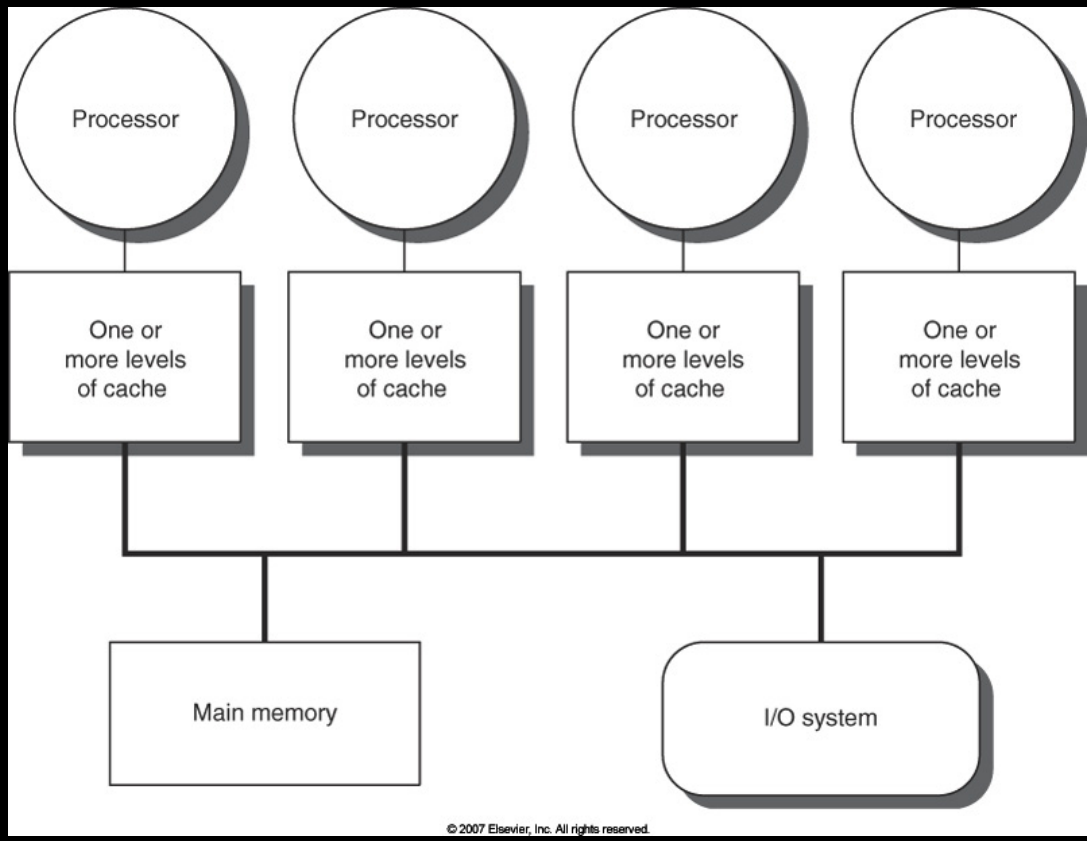
## CONVEY COPROCESSOR

| | |
|---|---|
| PROCESSOR | Based on Xilinx® Virtex® 5 FPGA |
| COPROCESSOR MEMORY | DIMMs: minimum 8, maximum 16<br>DIMM types: Standard PC-5300 DDR2 (max 128GB) ECC Convey Scatter-Gather SGDIMMs (max 32GB) ECC |

# Typical High Performance Computer Architectures

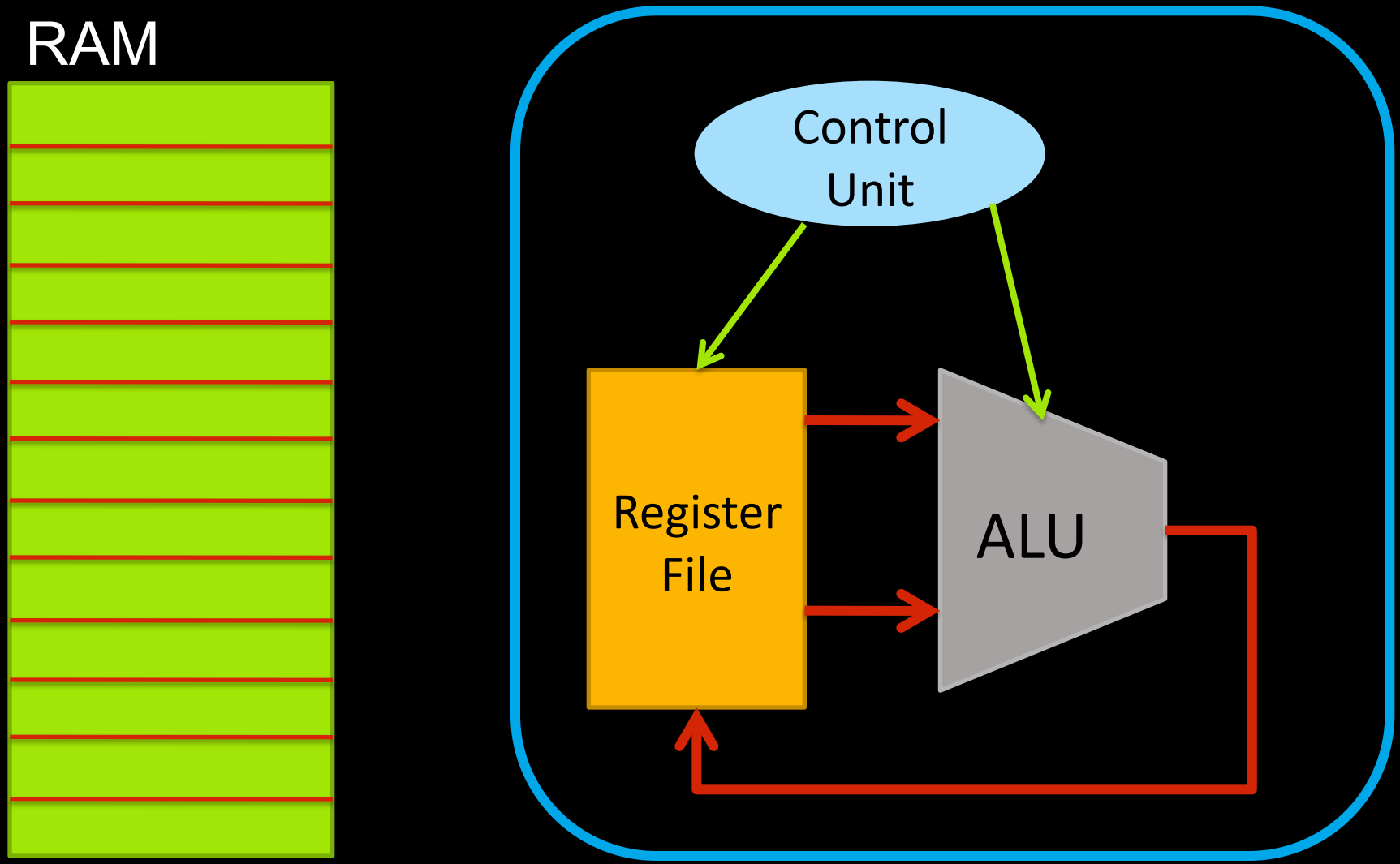



A cluster
(example of a distributed memory system)




shared memory computer

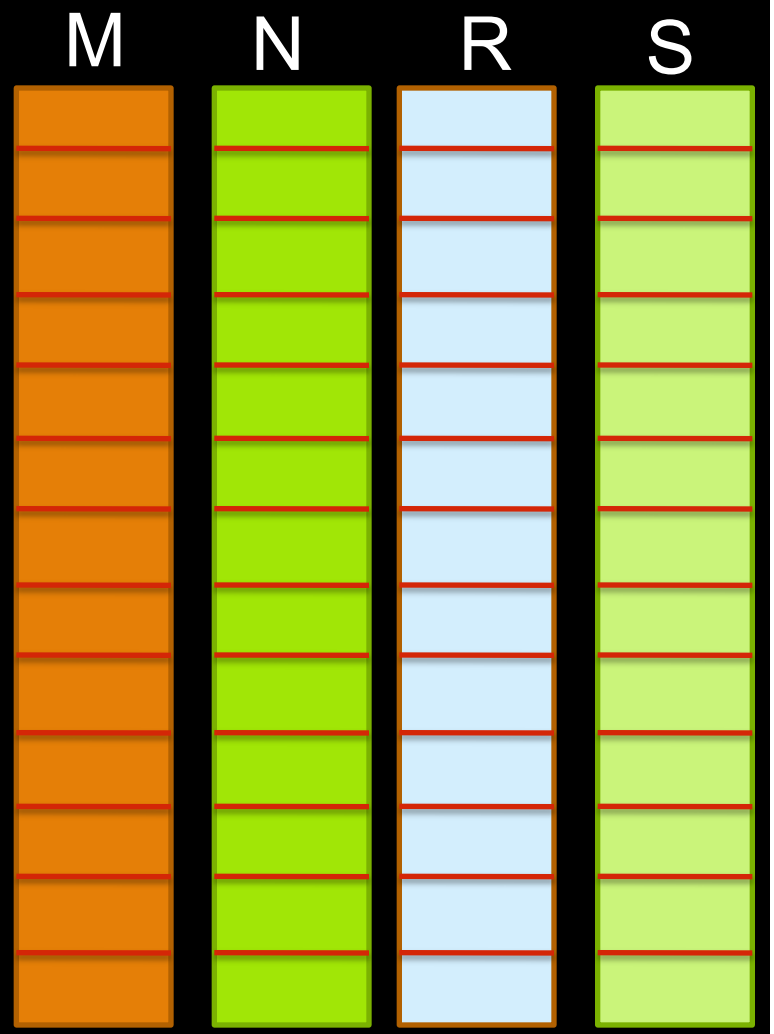
# How a stored program machine computes



# Example

- Perform the following operation over each corresponding element of the vectors M,N,R,S:

  f[i] = m[i]*n[i] + r[i]*s[i]

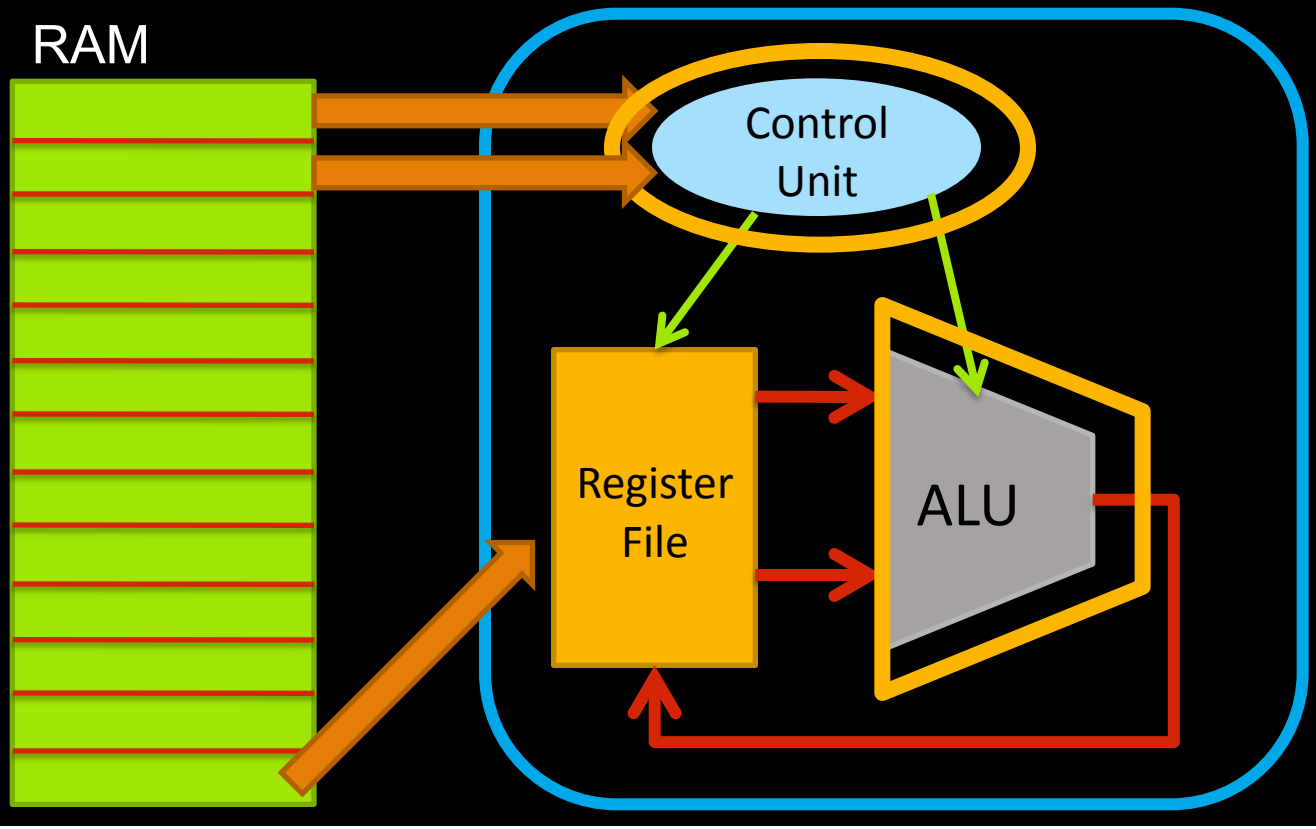


# The program tells the computer what to perform

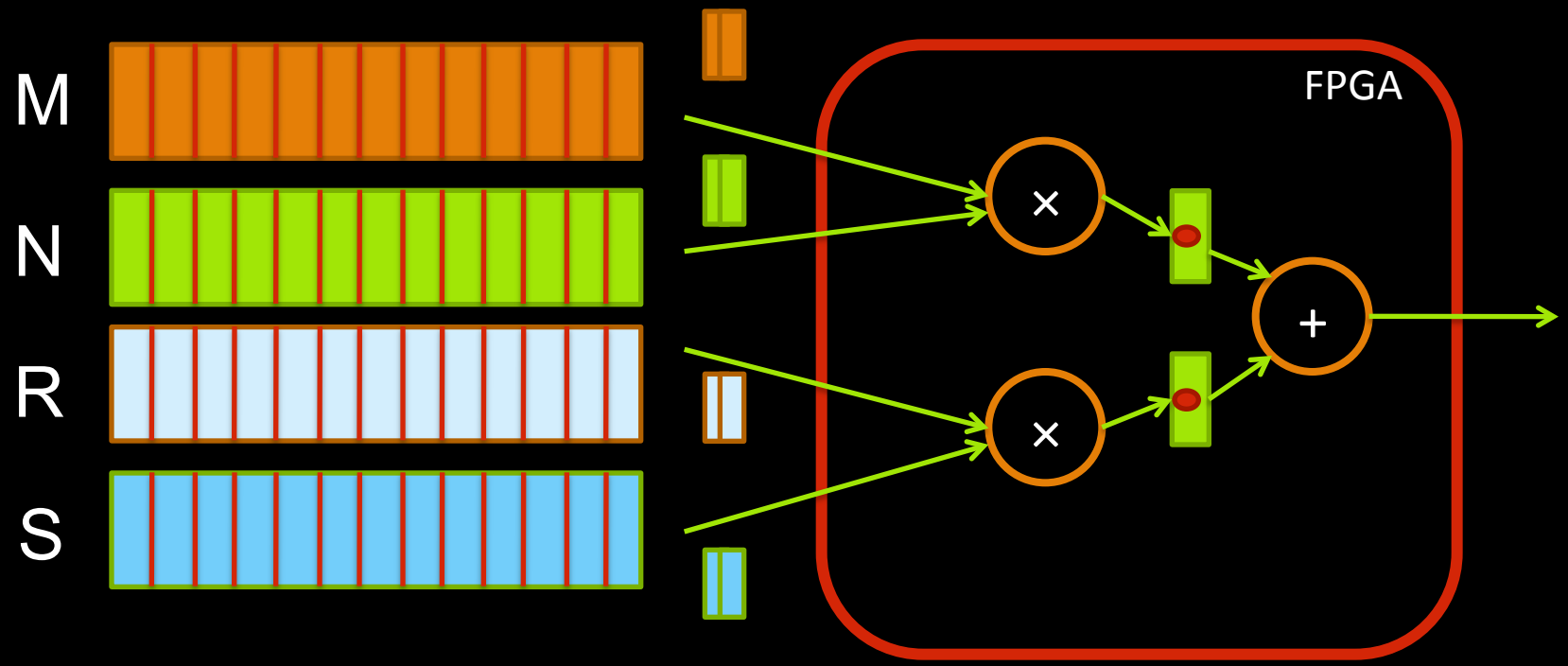
```
i = 0
while i <= 1000
  load m[i] to reg1
  load n[i] to reg2
  mul reg1, reg2 to reg3
  load r[i] to reg1
  load s[i] to reg2
  mul reg1, reg2 to reg4
  add reg3, reg4 to reg5
  save reg5 to f[i]
  increment i
end for
```

To execute any program, the stored-program computer repeats the following:

(1) fetch an instruction from memory
(2) decode it
(3) execute it
(4) repeat from (1)



# How an FPGA computes




- Does away with the stored program concept
- Can complete the computation of each index on each cycle

# What I do, related to RC

- Manage a project to use newly purchased RCs for projects in biology, mathematics, engineering.
- Design custom architectures for applications.
- Automating algorithm translation to reconfigurable devices.

## Get involved: Contact rafael.arce@upr.edu

- Learn about one of the paradigms that will allows to maintain performance improvements in computation.
- Help us achieve 10x-100x speedups in applications in bioinformatics and signal processing.

# Embrace hardware

Typical perception
of hardware

My perception

# Sample program to perform a*b+c.

```
// Options: -cpp
Mitrion-C 1.5;
type AttRAM = typedef mem bits:128[0x40000];

(AttRAM, AttRAM) main (AttRAM a0, AttRAM b0) {
    (a1, b1) = foreach(e in <0 .. 262143>) {
        (m, a1)     = memread(a0, e);
        int:32[4] a = m;
        int:128 res = a[0] * a[1] + a[2];
        b1          = memwrite(b0, e, res);
    }(a1, b1);
} (a1, b1);
```

# Thanks!

- Questions

- Rafael Arce Nazario
  rafael.arce@upr.edu
  ccom.uprrp.edu/~rarce/rclab