Electrical & Computer Engineering

Nov 29, 2001

EE552
High Level
Digital ASIC
Design using
CAD

C-RAM Parallel Computer

Presented By

Shahid Aslam Khan

Sue Ann Ung

Satneev Bhamra

About this Project

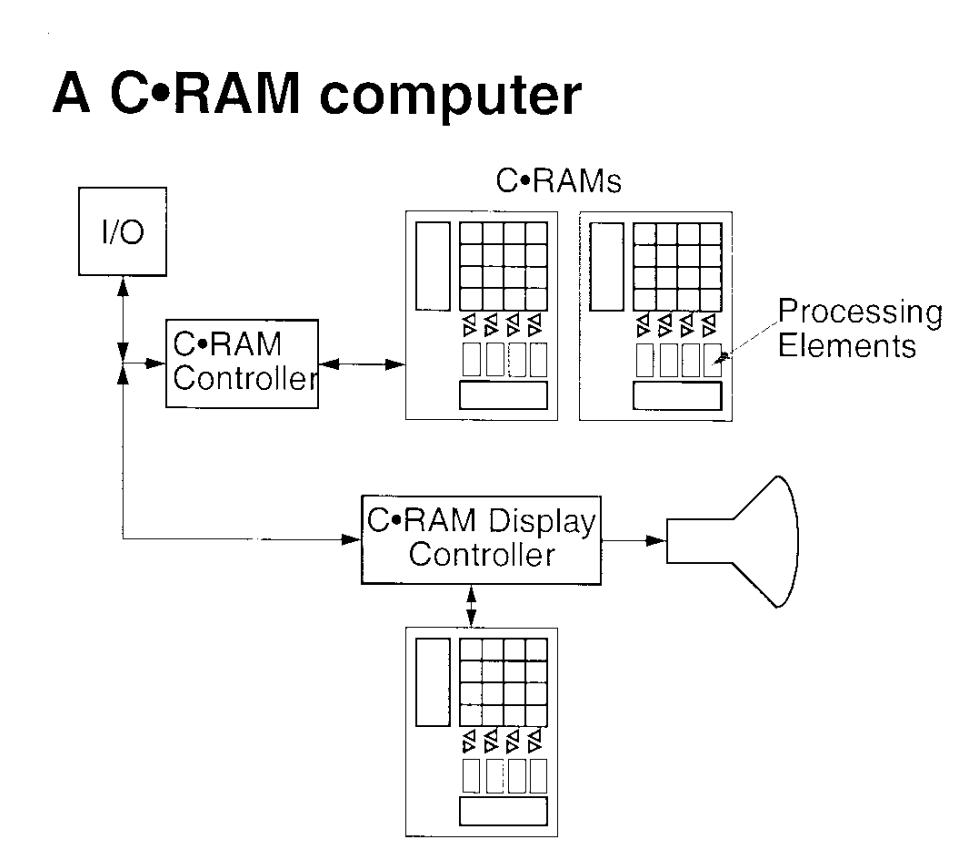
- **Computational RAM (C-RAM)**
- **In a nutshell:**
  - **C-RAM**
    - Memory
    - Processing Elements (PE)
  - **Single Instruction Multiple Data architecture (SIMD)**
    - Parallel Computation
  - **Applications:**
    - Signal and Image Processing
    - Computer Graphics
    - Database, etc.

Outline

- The C-RAM Computer
- The C-RAM Parallel Processor System
- The Sequencer
- The Controller
- C-RAM Architecture
- PE Model
- IO Interface
- Chip Overview
- Achievements
- Acknowledgements
- Demo

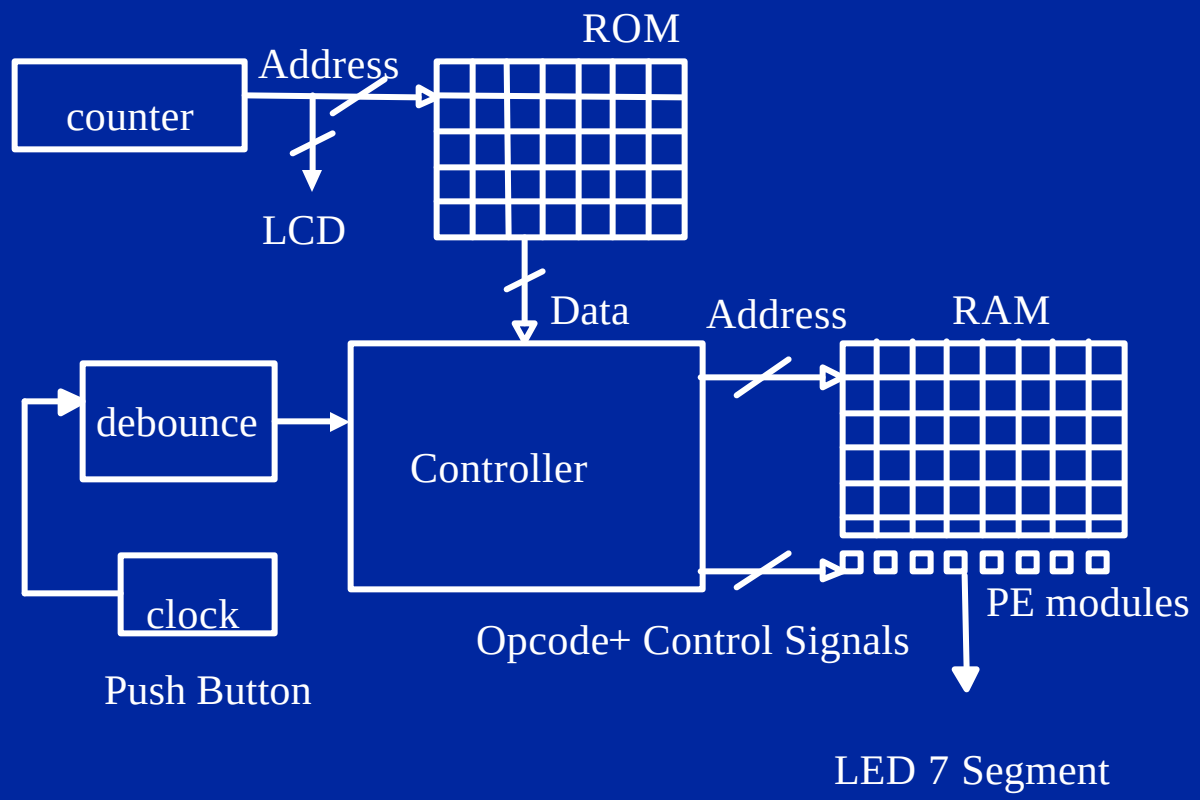
The C-RAM Computer

- **Broken down into 4 components**
  - **control unit**
  - **processing elements**
  - **memory modules**
  - **IO interfaces**



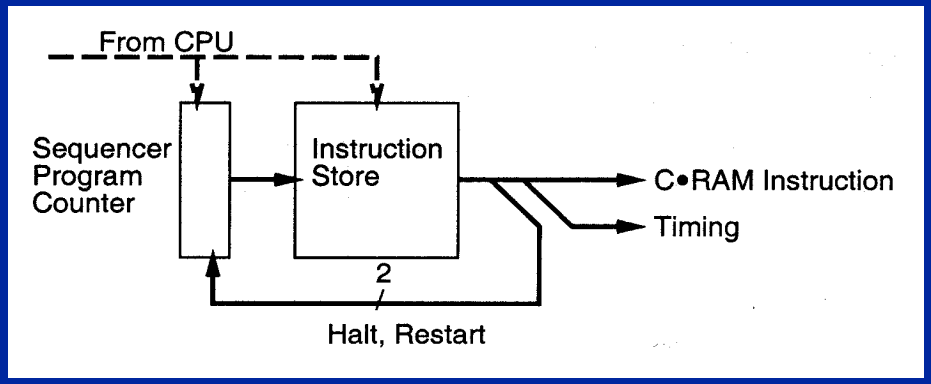
The C-RAM Parallel Processor System

- **Controller**
- **Counter/Sequencer**
- **RAM**
- **IO**

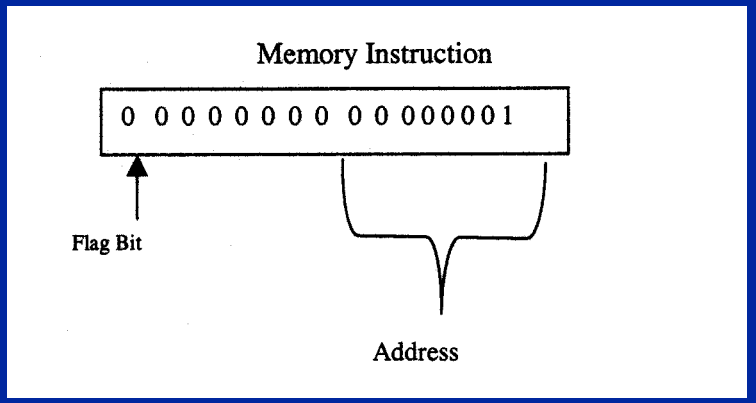


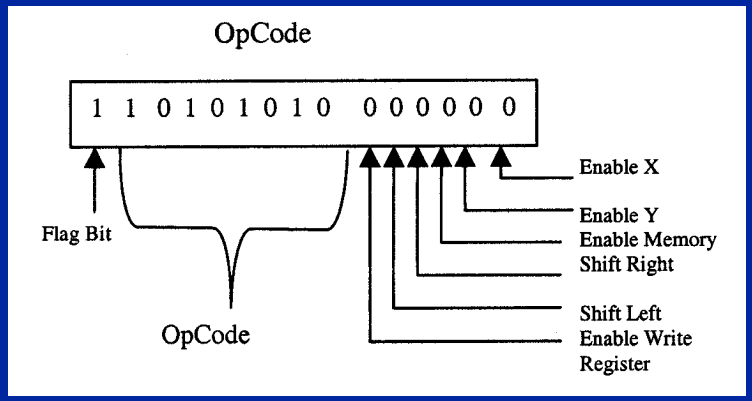
The Sequencer

- **Minimalist Microcoded Sequencer**



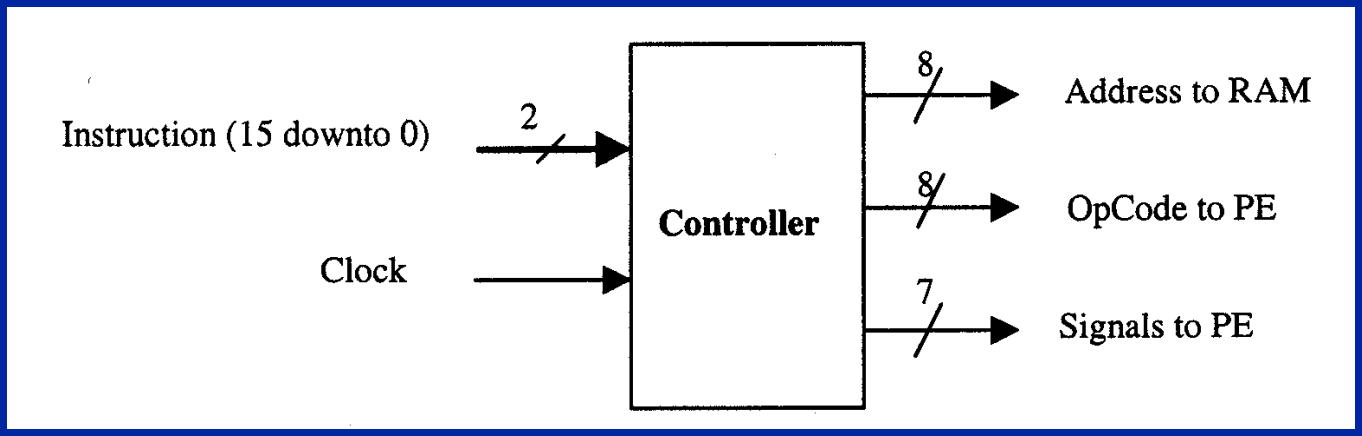
- **Opcodes and Memory instructions**





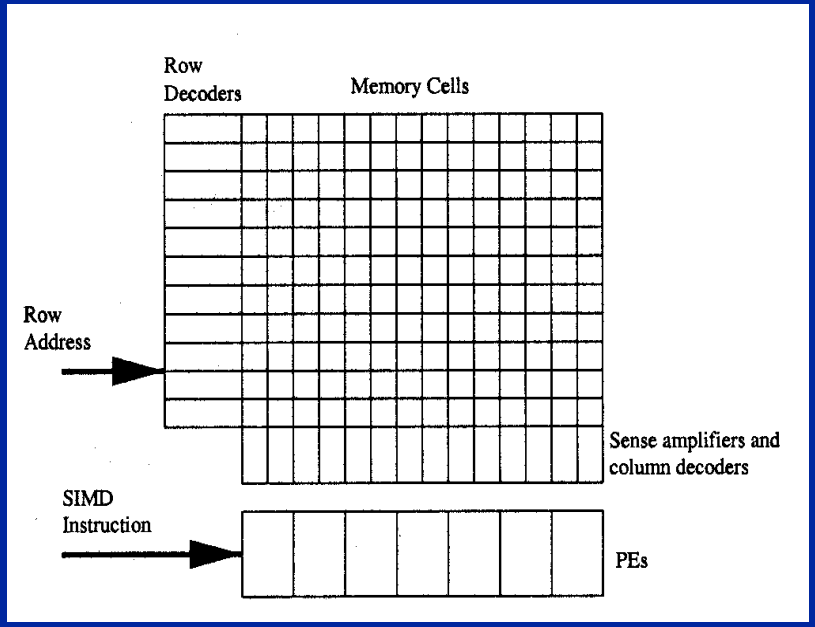
The Controller

- Based on a simple decoder
- Case statement in VHDL code
- Selects address
  - to read from the RAM or
  - to write to the RAM
- Issues opcodes or instructions to the PE



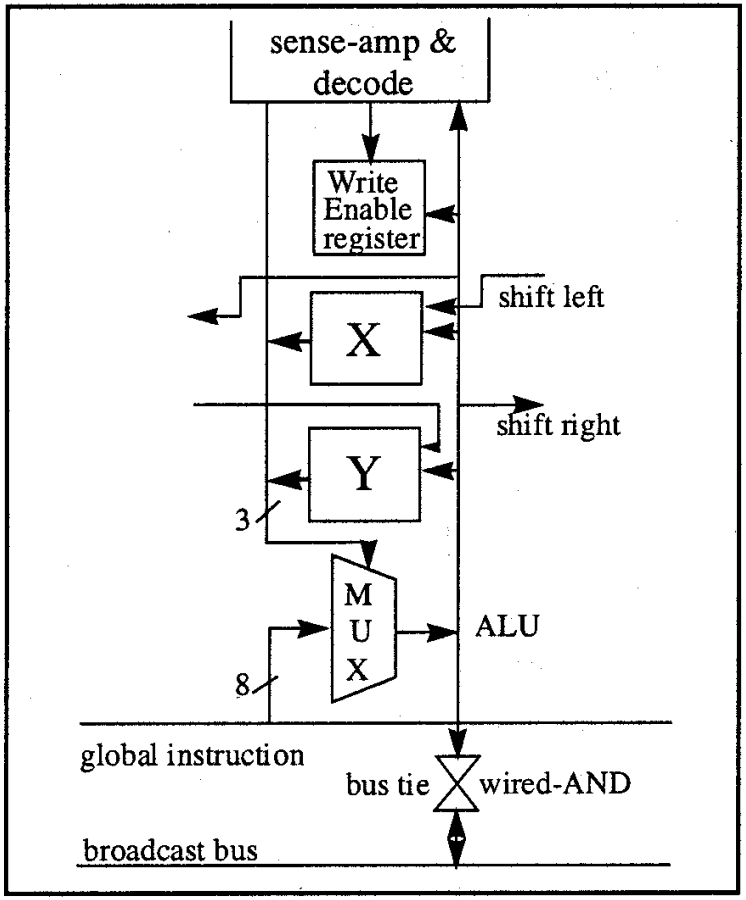
CRAM Architecture

- Implemented using SRAM cells on the Altera Board
- SIMD instructions come from the sequencer in the main controller
- Opcodes are routed to the individual PEs



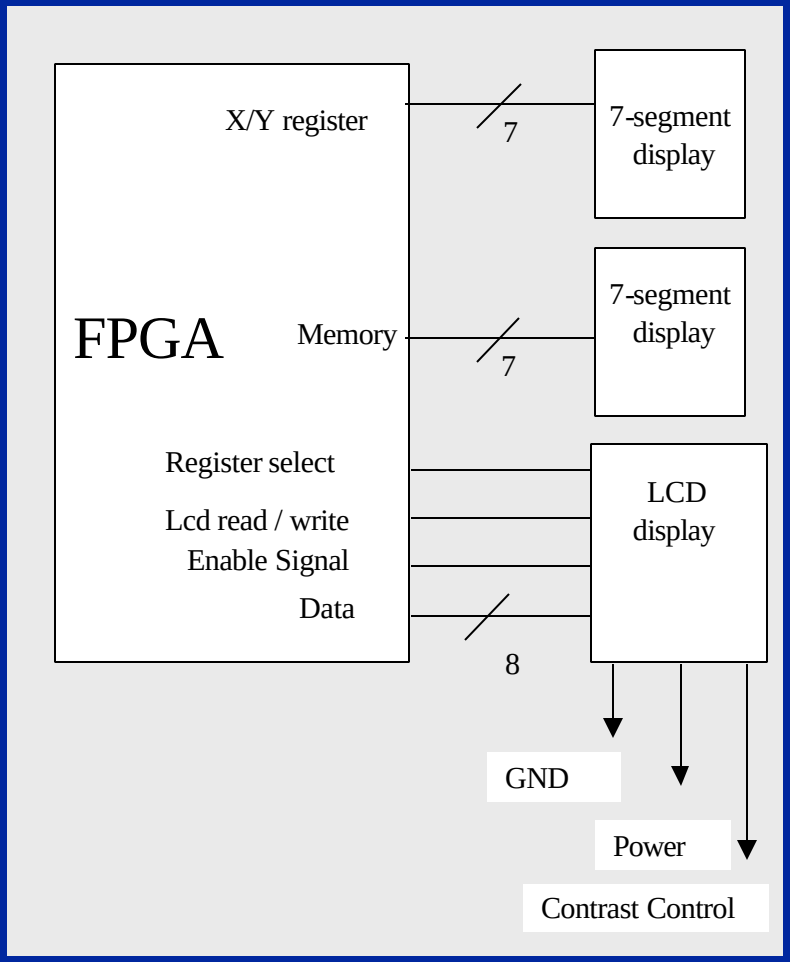
PE Model

- X and Y, 1 bit registers
- ALU
- 8 bit global instruction
- Data shifting
- Conditional Operations



IO Interface

- **Internal ROM / RAM for opcodes and instruction**
- **2-line LCD**
- **2 Seven Segment Display**



Chip Overview

Module	Logic Cells required
PE (1)	12 (1 %)
PE (2)	36 (3 %)
Controller	78 (6%)
I/O	254 (22%)
Total*	344 (30 %)

PE*: only one PE

- **Memory**
  - **256 x 16-bit instruction ROM initialized by .mif file**
  - **256 x 1-bit data RAM initialized by .mif file**
- **IO**
  - **7 segment display to show results of PE elements and memory**
  - **LCD display to show instruction set in the ROM**
- **Performance of C-RAM = 22.17 MHz**
- **Uses only 30% (with 1 PE) of the Altera FLEX10K20 Logic Cells**

Achievements

- **What the system can do now**
  - **More than one PE modules**
  - **Read and Write from/to the RAM**
  - **Logical Functions – Negation, OR, AND etc**
  - **Addition, Subtraction, Copy etc**

- **Improvements that needed to be made**
  - **Add more operations such as multiplication and division**
  - **Use ARM processor as a host to C-RAM module**

Acknowledgements

- **Elliott, Duncan G., Stumm, Michael, Snelgrove, W. Martin, Cojocaru, Christian, McKenzie, Robert (1999). *Computational RAM: Implementing Processors in Memory* (p32-41) IEEE Design and Test of Computers January – March 1999**

- **Elliott, Duncan G. (1998). *Computational RAM: A Memory – SIMD Hybrid***

  Phd Thesis, The University of Toronto

- **Aklilu, Noah, Elliott, Duncan G., Wickman, Curtis A. *A Tightly Coupled Hybrid SIMD/SISD System.***

  MSc Thesis, The University of Alberta

Demo

Addition Algorithm on C-RAM

Elliott, Duncan G. (1998). *Computational RAM: A Memory – SIMD Hybrid*
Phd Thesis, The University of Toronto

```
select B[least_significant_bit]
        X = M          // first bit is a special case with
                          no carry in
select A[least_significant_bit]
        Y = M&X        // Calculate carry
        M = X = X^M    // Calculate sum and write
                          back
for j = least significant bit  + 1..most significant bit
        select B[j]
                X = M^Y   // sum B[j] and carry in
                Y = M&Y   // carry out
        select A[j]
                Y = Y | (M&X)    // carry out
                M = X = X^M      // sum A[j] and partial
                                    sum, write back
end for
```

Instructions

```
0000000000000000; ---Address of B(0)
1101010100000010; --- X = M
1101010100000000; -- disable x
0000000000000100; ---Address of A(0)
1101000000000100; -- Y = M&X
1010110100000000; -- opcode to get X^M
1010110100001000; -- M = X^M enabling
      memory
1010110100000010; -- X = X^M enabling x
      register
0000000000000001; -- Address of B(1)
1011001100000010; -- X = M^Y
1100010000000100; -- Y = M&Y
0000000000000101; -- Address of A(1)
1111011000000100; -- Y = Y|(M&X)
010110100000000; -- opcode to get X^M
1010110100001000; -- M = X^M enabling
      memory
1010110100000010; -- X = X^M enabling x
      register
1010110100000000; -- disable x
0000000000000010; -- Address of B(2)
```

```
00000000 : 1; --Data in B(0)

00000100 : 1; --Data in A(0)

00000001 : 1; --Data in B(1)

00000101 : 1; --Data in A(1)

00000110 : 1; --Data in A(2)

00000010 : 0; --Data in B(2)
```