Abstract

The special purpose data processor from CE1911 can be extended with a larger collection of local storage registers. This larger collection allows more complex calculations because more than two numbers can be near the arithmetic circuit for quick access. Historically, this collection is called a register file. Registers files are sized between 4 and 64 registers. Modern processors typically have a register file with 16 or 32 registers.

Large data sets will require even more storage. Consider each pixel on an LCD display. A 1920 pixel x 1280 pixel LCD that displays red/green/blue (RGB) color requires that one byte per color for each pixel be stored. This results in more than 7 millions bytes of data calculated using equation 1.

$$Equation\ (1): \qquad bytes = (1920 * 1280) pixels * 3 \frac{bytes}{pixel} = 7{,}372{,}800\ bytes\ of\ data$$

Standard size units have been defined for computer memory systems addressed with binary numbers. Table 1 shows these binary prefixes as defined by standards written by the Joint Electron Device Engineering Council (JEDEC) and the International Electrotechnical Commission (IEC). In CE1911, the historic abbreviations based on the metric system were introduced. Students are not expected to know the modern names until CE1921.

Table 1: Binary Memory Prefixes

Power-of-2	Modern Name (21st Century)	Modern Abbreviation	Historic Name (metric system)	Historic Abbreviation
2^{10}	kibi	Ki	kilo	K
2^{20}	mebi	Mi	mega	M
2^{30}	gibi	Gi	giga	G
2^{40}	tebi	Ti	tera	T
2^{50}	pebi	Pi	peta	P

The calculation in equation 2 shows the total number of LCD pixel bytes using these prefixes.

$$Equation\ (2): \qquad 7{,}372{,}800\ bytes\ of\ data * \frac{1\ M}{2^{20}} = 7.03\ megabytes = 7.03MB$$

In the modern unit, the answer would be stated as 7.03 mebibytes = 7.03 MiB.

Large data sets like this screen image example are held in random access memory (RAM). The RAM memory can be multiplexed into the special purpose data processor and controlled by controller generated addresses. By adding a register file and a larger RAM memory, the CE1911 special purpose data processor gains significant added functionality through an enhanced memory system.

Register File Structural Design

This file cabinet of registers is accessed using two multiplexer selection signals for data reads. Each selection signals forms the **"address"** of one piece of data that will be used for the ALU calculation. In Figure 1, the output multiplexers choose one of four internal registers based on these **read data address busses** called RDDATA1 and RDDATA0. Because two values are read simultaneously, the register file is said to be a 2-Read register file.

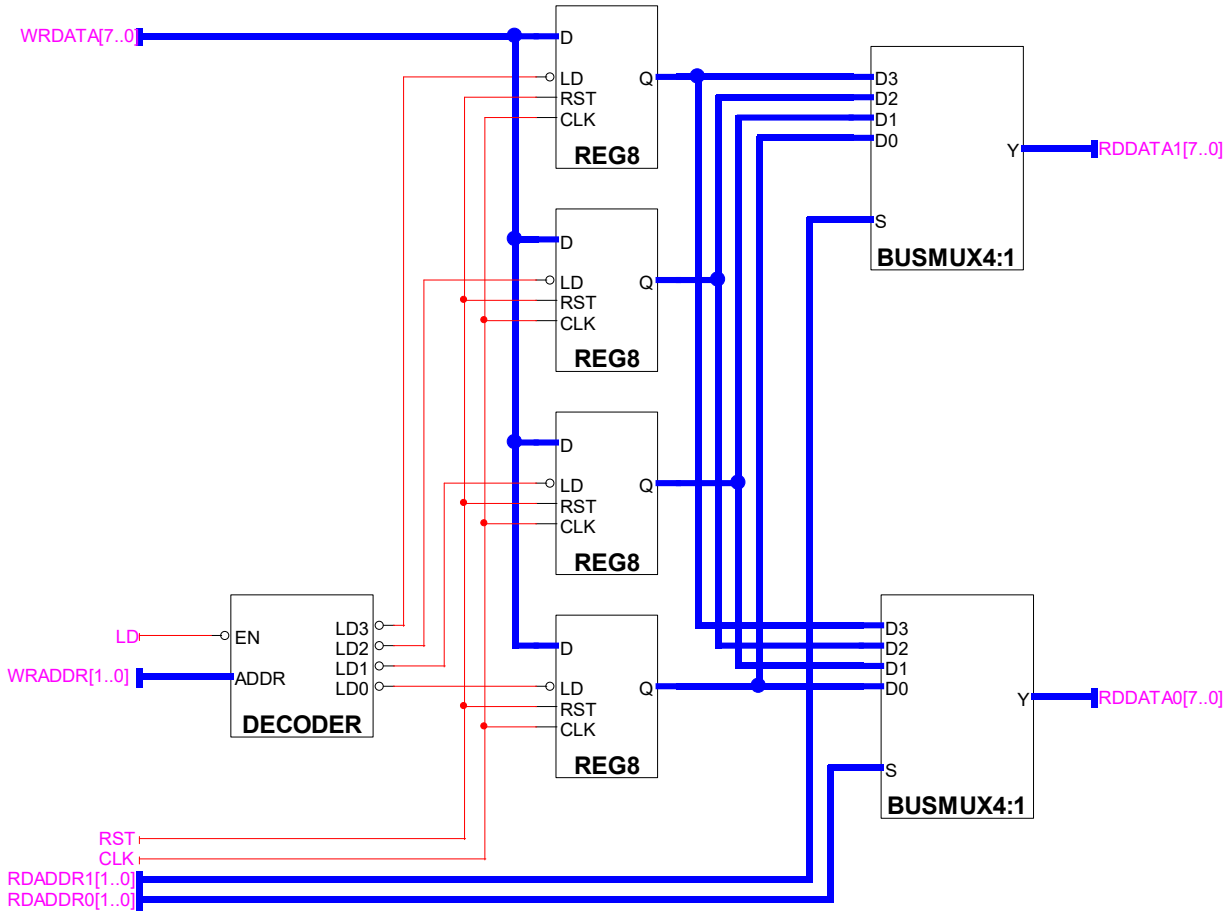

Figure 1: A 2-Read/1-Write Register File

The ALU calculates a single result. This result needs to be stored back into a register for potential use later in the algebraic calculation. Thus, Figure 1 also shows a **write address bus** called WRADDR. This address is decoded to determine which register loads if enabled. Load behavior is selected using the LD control signal. Thus, Figure 1 diagrams a 2-Read/1-Write port register file.

Register File VHDL Design

The structural design in Figure 1 can be created through schematic tools. Alternatively, a VHDL description can be provided to a tool like the Altera Quartus Synthesizer.

```
-- ************************************************************************
-- * CE1911                                                               *
-- * Out-of-class example: posted to web for independent study            *
-- * Name: Dr. Meier                                                      *
-- * Description:                                                         *
-- *                                                                      *
-- *   This file implements a register file with four (4) registers.      *
-- *   It uses integers rather than std_logic_vectors because VHDL        *
-- *   cleanly implements integer-based array indexing.                   *
-- *                                                                      *
-- ************************************************************************
-- *   This register file is two (2) read ports and one (1) write port.   *
-- *   Read addresses control the output multiplexers selecting the       *
-- *   registers to output. A write address controls the register loading*
-- *   input data. A load control signal allows load enable and disable. *
-- ************************************************************************

-- Load the ieee libraries.
-- Using rising_edge to simplify writing clock edge statement.
library ieee;
use ieee.std_logic_1164.all;

entity regfile4 is
port(wrdata: in integer range 0 to 255;
     rdaddr1, rdaddr0, wraddr: in integer range 0 to 3;
      ld,rst,clk: in std_logic;
      rddata1, rddata0: out integer range 0 to 255);
end entity regfile4;

architecture behavioral of regfile4 is
  -- A register file is a collection of registers.
  -- VHDL includes the most basic collection data structure: array.
  -- Using a user defined type to create an array of storage locations.
  type regarray is array(0 to 3) of integer range 0 to 255;
  signal registers: regarray;
begin

  -- Always output requested read data.
  -- The "array index" forms a selection.
  -- Selection forms an output multiplexer.
  rddata1 <= registers(rdaddr1);
  rddata0 <= registers(rdaddr0);

  -- Now handle the register load.
  -- VHDL 2008 all keyword used: enable compiler setting for VHDL2008.
  -- Implementing active-low load.
  update: process(all)
  begin
    if rst = '0' then
        registers(0) <= 0;
         registers(1) <= 0;
         registers(2) <= 0;
         registers(3) <= 0;
    elsif rising_edge(CLK) then
      if ld = '0' then
         registers(wraddr) <= wrdata;
       end if;
    end if;
  end process;

end architecture BEHAVIORAL;
```

RAM Memory VHDL Design

```
-- ************************************************************************
-- * CE1911                                                               *
-- * Out-of-class example: posted to web for independent study            *
-- * Name: Dr. Meier                                                      *
-- * Description:                                                         *
-- *                                                                      *
-- *   This file implements a 256 location random access memory.          *
-- *   It uses integers rather than std_logic_vectors because VHDL        *
-- *   cleanly implements integer-based array indexing.                   *
-- *                                                                      *
-- *   This RAM is smaller than typical. Usually, RAM begins at 1KB and   *
-- *   today reaches into the GB (or GiB) in size.                        *
-- *                                                                      *
-- ************************************************************************
-- *   This RAM has one (1) read ports and one (1) write port.            *
-- *   A read addresses controls the output multiplexer selecting the     *
-- *   memory location to output. A write address controls the RAM load.  *
-- *   A load control signal allows load enable and disable.              *
-- ************************************************************************

-- Load the ieee libraries.
-- Using rising_edge to simplify writing clock edge statement.
library ieee;
use ieee.std_logic_1164.all;

entity ram is
port(addr: in integer range 0 to 255;
     wrdata: in integer range 0 to 255;
     ld,rst,clk: in std_logic;
      rddata: out integer range 0 to 255);
end entity ram;

architecture behavioral of ram is
  -- A RAM is a collection of numbers.
  -- VHDL includes the most basic collection data structure: array.
  -- Using a user defined type to create an array of storage locations.
  type ramarray is array(0 to 255) of integer range 0 to 255;
  signal rammem: ramarray;
begin

  -- Always output requested read data.
  -- The "array index" forms a selection.
  -- Selection forms an output multiplexer.
  rddata <= rammem(addr);

  -- Now handle the RAM load.
  -- VHDL 2008 all keyword used: enable compiler setting for VHDL2008.
  -- Implementing active-low load.
  update: process(all)
  begin
    if rst = '0' then
        for i in 0 to 255 loop
         rammem(i) <= 0;
         end loop;
     elsif rising_edge(CLK) then
       if ld = '0' then
          rammem(addr) <= wrdata;
        end if;
     end if;
   end process;

end architecture BEHAVIORAL;
```

Special Purpose Computer Enhanced by Register File and a small RAM Memory of 256 locations