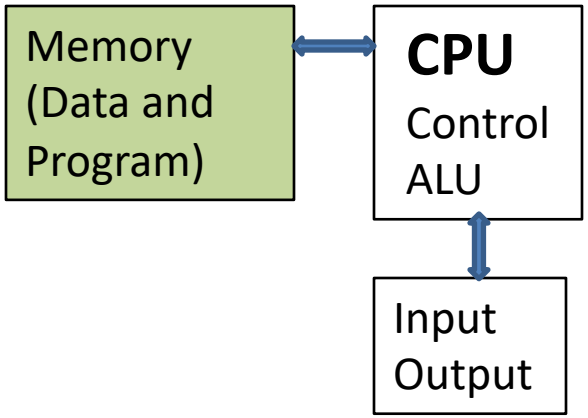
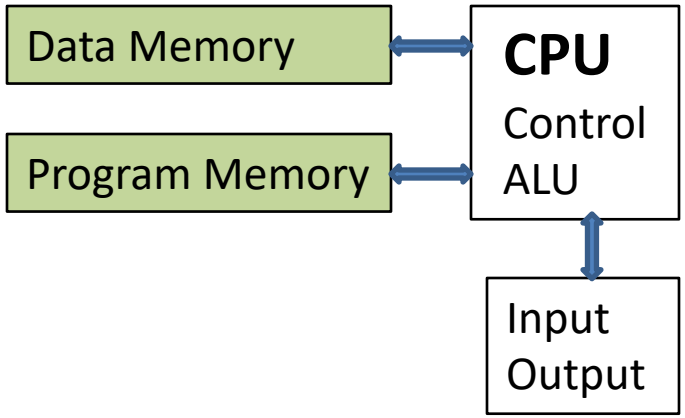
FPGA HARVARD ARCHITECTURE

Von Neumann Architecture



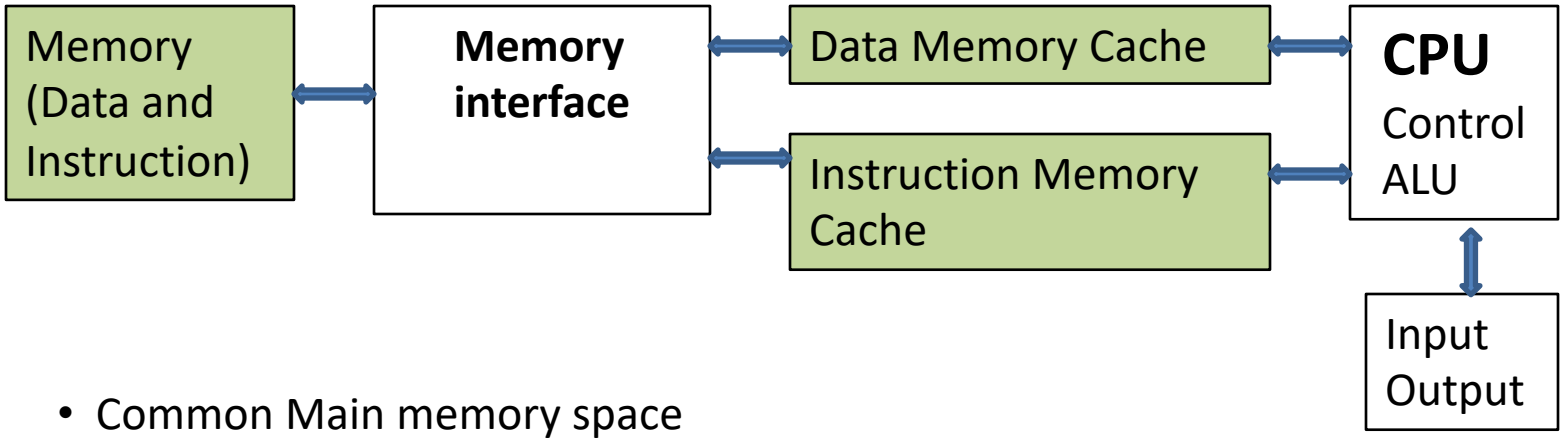
- Shared memory space
- CPU to memory is bottleneck
- Data and instructions have to share the memory interface
- The basic word size is the same for data and instructions

Harvard Architecture



- Separate memory spaces
- Data and instructions access can happen at the same time
- The basic word size can be different for data and instructions

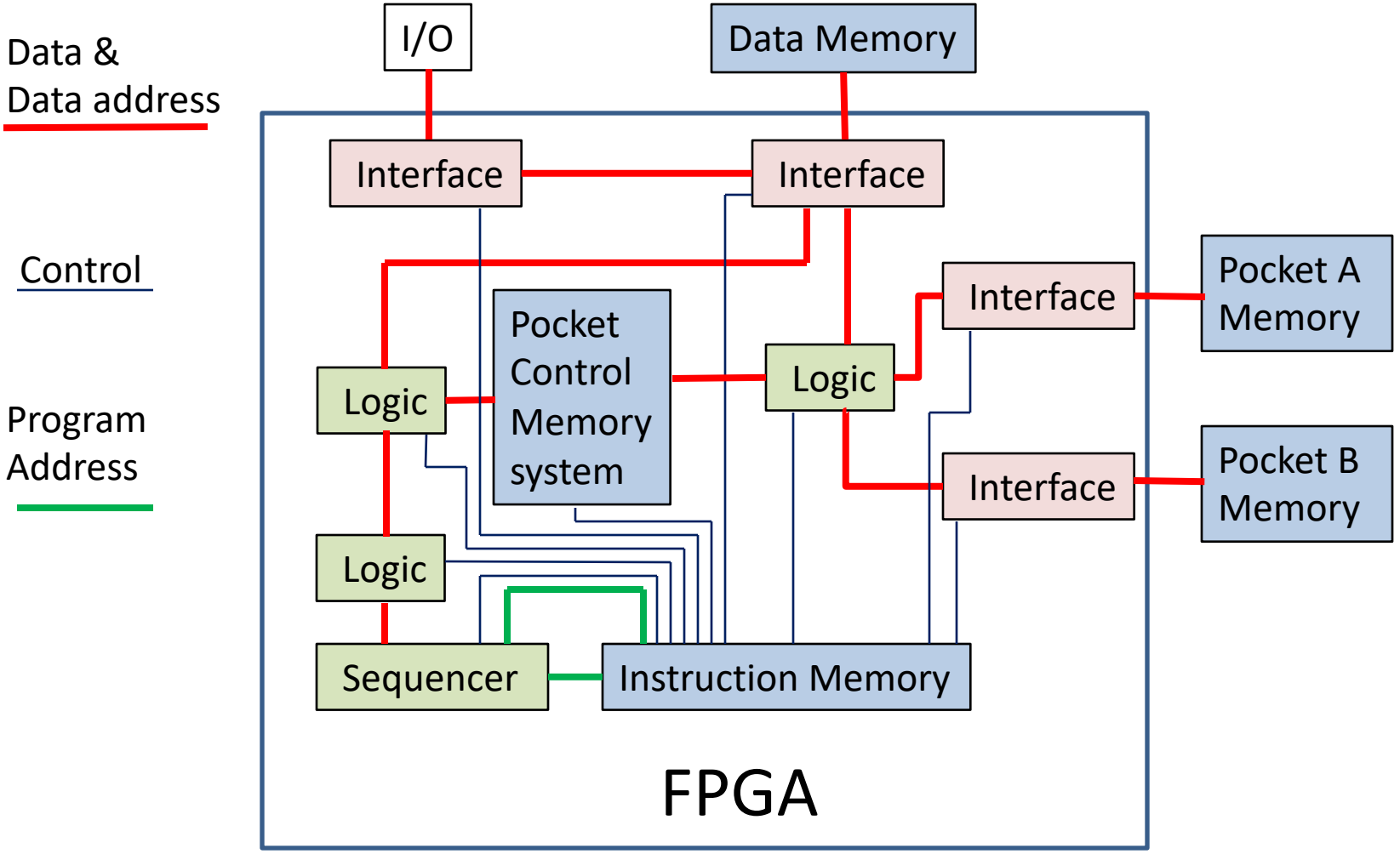
Modified Harvard Architecture



- Common Main memory space
- CPU to memory bottleneck only for things not in the cache
- Separate memory spaces for data cache and instruction cache
- Data and instructions access can happen at the same time
- The basic word size can be different for data and instructions
- Most modern processors use this approach inside the CPU chip

Linear Sort in FPGA

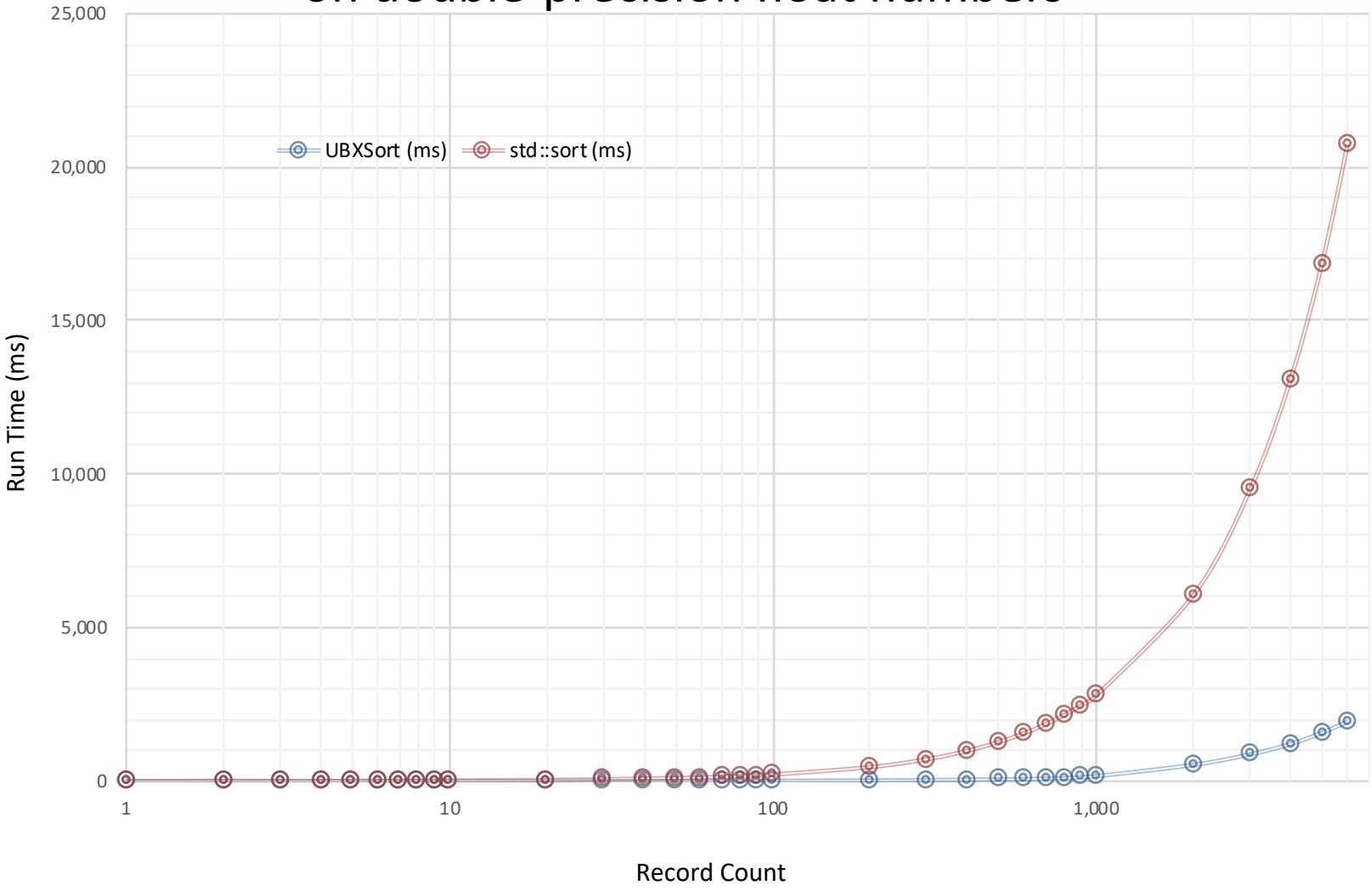
The instruction memory is completely separate from the data memory



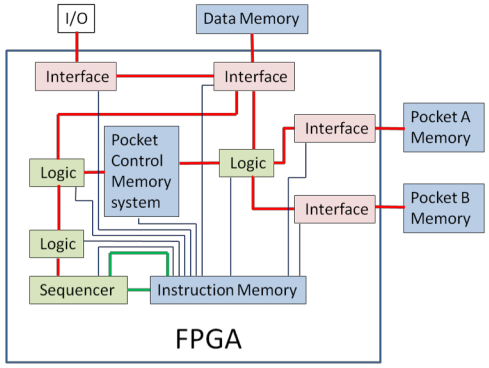
Performance Comparison of Linear UBXSort vs C++ std::sort()

- UBXSort is our patented sorting algorithm, US patent # 5,278,987
- It has a complexity of O(N)
- Typical conventional sorting algorithms have complexity of O(N * log(N))
- The larger the dataset size, the more advantageous of the UBXSort algorithm
- The benchmark test compares the C++ implementation of UBXSort against the sort function included in the standard C++ library
- Test is performed on a machine with Intel(R) Xeon(R) CPU E5-2637 v2 @ 3.50GHz, 16 cores, 512GB RAM
- The host runs on CentOS 6.7, using gcc-4.x, libstdc++.so.6.0.13
- The test meassures the time it takes to sort up to 6 million double precision float numbers

Runtime Comparison: UBXSort vs C++ std:sort() on double-precision float numbers



FPGA Sort Implementation advantage



UBXSort C++ code to sort one character position

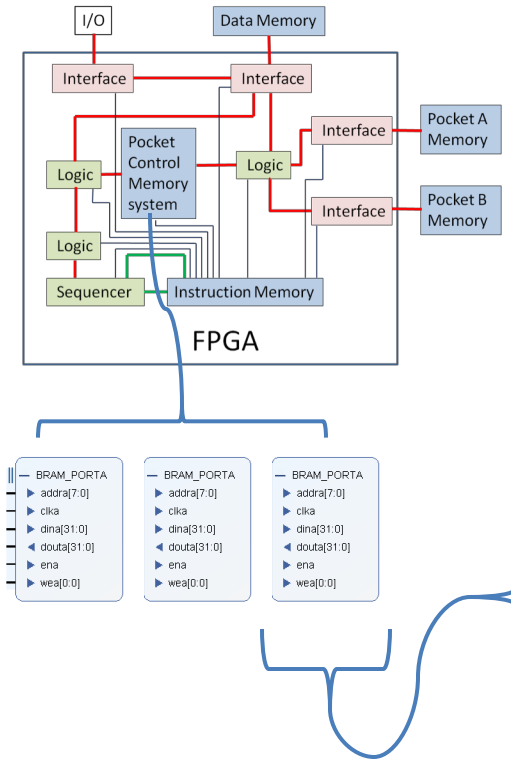
- The C code to sort one character of the key field

- This is the step that takes most of the time. The for loop cycles through all of the records. For each cycle it reads the character to be sorted and does the steps needed to generate the sorted order for that column, which include: memory reads, memory writes, arithmetic functions and decision branch points.

- In the FPGA implementation all of this work is done in the time that it takes to do one memory read, about 12 ns. Therefore the sorting time for 100,000,000 records on 8 character (or 64 bit) field is about 10 seconds.

```
void UBSorter::ProcColumn(unsigned char * pBuf)
{
   int i, j, j0;
   unsigned char c;
   i = startRcd_;
   for (i = 0; i < nRcds_; ++i) {
      c = pBuf[j * stride_];
      if (flags_[c]) { pDest_[ curPocket_[c] ] = j; }
      else {
         iniPocket_[c] = j;
         flags_[c] = true;
      }
      curPocket_[c] = j;
      j = pSrc_[j];
   }

   j0 = -1;
   for (i = 0; i < 256; ++i) {
      if (flags_[i]) {
         if (j0 == -1) {
            j0 = curPocket_[i];
            startRcd_ = iniPocket_[i];
         } else {
            pDest_[j0] = iniPocket_[i];
            j0 = curPocket_[i];
         }
      }
   }
   endRcd_ = j0;
   pDest_[endRcd_] = -1;
```

FPGA Implementation



Verilog or VHDL

```
-- IP VLNV: xilinx.com:ip:blk_mem_gen:8.4
-- IP Revision: 2

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

LIBRARY blk_mem_gen_v8_4_2;
USE blk_mem_gen_v8_4_2.blk_mem_gen_v8_4_2;

ENTITY rcd_mem IS
 PORT (
  clka : IN STD_LOGIC;
  ena : IN STD_LOGIC;
  wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
  addra : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
  dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
  douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
 );
END rcd_mem;

ARCHITECTURE rcd_mem_arch OF rcd_mem IS
  COMPONENT blk_mem_gen_v8_4_2 IS
   GENERIC (
    C_FAMILY : STRING;
    C_XDEVICEFAMILY : STRING;
    C_ELABORATION_DIR : STRING;
    C_INTERFACE_TYPE : INTEGER;

        70 more lines of text

    C_DISABLE_WARN_BHV_RANGE : INTEGER;
    C_COUNT_36K_BRAM : STRING;
    C_COUNT_18K_BRAM : STRING;
    C_EST_POWER_SUMMARY : STRING
```

```
PORT (
    clka : IN STD_LOGIC;
    rsta : IN STD_LOGIC;
    ena : IN STD_LOGIC;
    regcea : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);

    70 more lines of text

    s_axi_dbiterr : OUT STD_LOGIC;
    s_axi_rdaddrecc : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
 END COMPONENT blk_mem_gen_v8_4_2;
 ATTRIBUTE X_CORE_INFO : STRING;
 ATTRIBUTE X_CORE_INFO OF rcd_mem_arch:
      ARCHITECTURE IS "blk_mem_gen_v8_4_2,Vivado 2018.3";
 ATTRIBUTE CHECK_LICENSE_TYPE : STRING;
 ATTRIBUTE CHECK_LICENSE_TYPE OF rcd_mem_arch :
      ARCHITECTURE IS "rcd_mem,blk_mem_gen_v8_4_2,{}";
 ATTRIBUTE CORE_GENERATION_INFO : STRING;
```

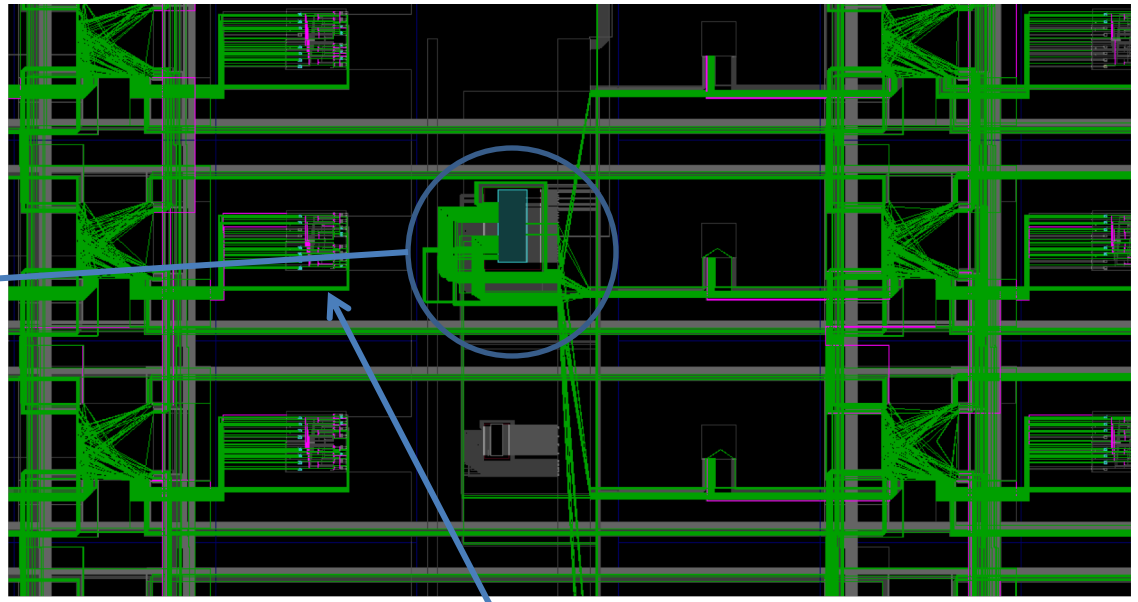
FPGA Chip Implementation

Verilog or VHDL

```
-- IP VLNV: xilinx.com:ip:blk_mem_gen:8.4
-- IP Revision: 2

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

LIBRARY blk_mem_gen_v8_4_2;
USE blk_mem_gen_v8_4_2.blk_mem_gen_v8_4_2;

ENTITY rcd_mem IS
  PORT (
    clka : IN STD_LOGIC;
    ena : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0)
  );
END rcd_mem;

ARCHITECTURE rcd_mem_arch OF rcd_mem IS
  COMPONENT blk_mem_gen_v8_4_2 IS
  GENERIC (
    C_FAMILY : STRING;
    C_XDEVICEFAMILY : STRING;
    C_ELABORATION_DIR : STRING;
    C_INTERFACE_TYPE : INTEGER;

        70 more lines of text

    C_DISABLE_WARN_BHV_RANGE : INTEGER;
    C_COUNT_36K_BRAM : STRING;
    C_COUNT_18K_BRAM : STRING;
    C_EST_POWER_SUMMARY : STRING
```

```
  PORT (
    clka : IN STD_LOGIC;
    rsta : IN STD_LOGIC;
    ena : IN STD_LOGIC;
    regcea : IN STD_LOGIC;
    wea : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
    addra : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    dina : IN STD_LOGIC_VECTOR(31 DOWNTO 0);
    douta : OUT STD_LOGIC_VECTOR(31 DOWNTO 0);

    70 more lines of test

    s_axi_dbiterr : OUT STD_LOGIC;
    s_axi_rdaddrecc : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
  );
END COMPONENT blk_mem_gen_v8_4_2;
ATTRIBUTE X_CORE_INFO : STRING;
ATTRIBUTE X_CORE_INFO OF rcd_mem_arch:
    ARCHITECTURE IS "blk_mem_gen_v8_4_2,Vivado 2018.3";
ATTRIBUTE CHECK_LICENSE_TYPE : STRING;
ATTRIBUTE CHECK_LICENSE_TYPE OF rcd_mem_arch :
    ARCHITECTURE IS "rcd_mem,blk_mem_gen_v8_4_2_0";
ATTRIBUTE CORE_GENERATION_INFO : STRING;
```

Block of memory



The colored lines are a representation of the connections between the logical elements

Slice of 8 LUTs
6 of which are used