Лабораторная работа 3
По курсу: «Компьютерная схемотехника»

Проектирование оперативного запоминающего устройства

1. **Цель лабораторной работы**:
овладеть знаниями и практическими навыками по проектированию устройств памяти, таких как ОЗУ (RAM). Лабораторная работа также служит для овладения навыками программирования и отладки описания RAM на языке VHDL.

2. **Теоретические сведения**

Блок RAM предназначен для быстрого доступа (Access), т.е. запоминания в блоке памяти (Memory) и выдачи n-разрядных слов по произвольному (Random) адресу. В лабораторной работе предполагается, что данное имеет разрядность n от 4 до 16 бит и объем памяти равен M от 512 до 8192 слов.

При проектировании микросхем, если объем RAM невелик (до 1024 бит), то память набирают из отдельных триггеров. В ином случае пользуются библиотеками готовых модулей памяти.

При разработке проекта для ПЛИС готовые модули памяти имеют объем 16, 32, 2048, 4096 бит. В последних сериях ПЛИС объем модуля RAM возрос до 16К или 18К бит. Разрядность модуля RAM может задаваться из ряда: 1,2,4,8 и 16 бит.

Запись данных и адреса в модуль RAM всегда выполняется по фронту синхросерии или сигнала записи, т.е. вход модуля можно рассматривать как вход синхронного регистра. Чтение данного чаще всего выполняется в следующем такте после такта приема адреса. Иногда на выходе модуля RAM стоит синхронный регистр, запоминающий прочитанное слово. Запись и чтение из модуля может выполняться по конвейерному принципу: в одном такте записывается адрес нового данного и выдается прочитанное данное по предыдущему адресу.

Для формирования RAM большого объема собирают систему из нескольких готовых модулей, дешифратора адреса для селекции модуля и выходного мультиплексора.

Различные варианты RAM в лабораторной работе имеют три, две или одну шину. В первом случае шины входного, выходного данного и адреса – раздельны, во втором случае шина входного и выходного данного совмещены и в третьем случае и адрес, и данные передаются по одной шине мультиплексированно.

3. **Элементная база**

При выполнении лабораторной работы предлагается реализовать RAM на базе ПЛИС фирмы Xilinx. В случае, когда объем памяти не превосходит 4096 бит следует использовать модуль памяти RAM16X1 объемом 16 бит, описанный в библиотеке UNISIM. Интерфейс этого модуля выглядит следующим образом

```
entity RAM16X1 is
port ( D : in std_ulogic;   -- входное данное
       WE : in std_ulogic; -- разрешение записи
       WCLK: in std_ulogic;-- синхросигнал для записи
       A0 : in std_ulogic; -- биты адреса
       A1 : in std_ulogic;
       A2 : in std_ulogic;
       A3 : in std_ulogic;
       O  : out std_ulogic); -- выходное данное
end RAM16X1;
```

Запись в модуль происходит по спаду сигнала WE, а чтение выполняется постоянно, если WE=0.

Порты модуля имеют тип std_ulogic. Этот тип – стандартный тип для всех проектов, подготавливаемых с помощью VHDL. Кроме значений логических 0 и 1 он также имеет, так называемые, металогические значения (meta – над, т.е. значения, представляющие "надстройку" над логическими 0 и 1). К их числу относятся значения H, L, называемые слабой единицей и слабым нулем, значение Z, представляющее третье состояние, значения X и W, представляющие собой сильную и слабую ошибку. Значения U и "-" означают "не инициализировано" и "безразлично". Благодаря такому множеству значений, возможно моделирование таких схемных конструкций, как "монтажная" логика, двунаправленная шина с тремя состояниями. Также на них основаны методики моделирования, поиска и локализации неисправностей и ошибок в проектах.

Модуль памяти на 256 шестнадцатиразрядных слов имеет следующий интерфейс.

```
entity RAMB4_S16 is
    port (DI: in STD_LOGIC_VECTOR (15 downto 0);--входное данное
        EN  : in STD_ULOGIC;  -- разрешение обращенияе
        WE  : in STD_ULOGIC;  -- сигнал записи
        RST : in STD_ULOGIC;  -- асинхронный сброс
        CLK : in STD_ULOGIC;  -- сигнал синхросерии
        ADDR: in STD_LOGIC_VECTOR (7 downto 0);--адрес
        DO  : out STD_LOGIC_VECTOR (15 downto 0) );--выходное данное
    end RAMB4_S16;
```

Аналогичный интерфейс имеют модуль RAMB4_S1 4096 однобитных данных , модуль RAMB4_S2 2048 двухразрядных слов, RAMB4_S4 - 1024 четырехразрядных слов и RAMB4_S8 - 512 восьмиразрядных слов. При этом шины адреса и данных имеют соответствующую разрядность. Запись в модуль происходит по фронту синхросигнала CLK при WE=1, а чтение – также по фронту синхросигнала, если WE=0.

В блоках RAM с двумя и одной шиной используются двунаправленные шины. Действие таких шин основано на том, что несколько источников шины электрически подключены к одной линии через тристабильные буферы. Корректность функционирования шины достигается за счет того, что одновременно к шине подключается с логическими уровнями 0 или 1 не более, чем один буфер, а остальные буферы включены в третье – высокоомное состояние, обозначенное как 'Z'. В ПЛИС фирмы Xilinx применяется тристабильный буфер, который имеет следующий интерфейс.

```
    entity BUFT is
    port( O: out   STD_ULOGIC; -- выход
          I: in    STD_ULOGIC; -- вход
          T: in    STD_ULOGIC);-- управляющий вход, если=1, то
    end BUFT;                   -- на выходе - 'Z'
```

4. **Примеры описания RAM**

Рассмотрим пример проектирования RAM объемом 1024 шестнадцатиразрядных слов с одной шиной адреса и данных. Объявление объекта для такой RAM выглядит следующим образом.

```
    library IEEE;
    use IEEE.STD_LOGIC_1164.all;
    library UNISIM;
    use UNISIM.all;
    use work.CNetlist.all;
    entity RAM is
     port(CLK : in BIT;  -- синхровход
          R : in BIT;  -- сброс
          WR: in BIT;  -- сигнал записи
          AE: in BIT;  -- сигнал фиксации адреса
          OE: in BIT;  -- сигнал выдачи прочитанного слова
          AD : inout STD_LOGIC_VECTOR(15 downto 0) );-- адрес/данное
    end RAM;
```

Здесь сперва объявлено, что используются стандартная библиотека IEEE и библиотека моделей компонентов фирмы Xilinx. Из первой библиотеки берется тип STD_LOGIC и STD_LOGIC_VECTOR, необходимые для организации тристабильной шины и связи с библиотечными компонентами, а также функции преобразования из типа STD_LOGIC в тип BIT и обратно. Из второй библиотеки берутся компоненты модулей RAM и тристабильного буфера.

4.1**. Поведенческая модель RAM.**

Поведенческая модель RAM представляет собой процесс, который не особо отличается от процесса, описывающего регистр, но в котором вместо вектора битов применяется массив векторов.

```
architecture BEH of RAM is
    type MEM1KX16 is array(0 to 1023) of BIT_VECTOR(15 downto 0);
    constant RAM_init: MEM1KX16:=  --начальное состояние памяти
    (X"0000",X"0000",X"0000",X"0000",X"0000",X"0000",X"0000",X"0000",
            others=> X"0000");
    signal addr,do: BIT_VECTOR(15 downto 0);
begin
RG_ADDR:process(CLK,R) begin  --регистр адреса
          if R='1' then
               addr<=X"0000";
          elsif CLK='1' and CLK'event and AE='1' then
               addr<= To_bitvector(AD);
          end if;
     end process;
       ------ собственно блок памяти -----------------
RAM1K:process(CLK,addr,addri)
           variable RAM: MEM1KX16:= RAM_init;
           variable addri:natural;
        begin
           addri<= BIT_TO_INT(addr(9 downto 0));
           if CLK='1' and CLK'event then
                  if WE = '1' then
                     RAM(addri):= To_bitvector(AD);  -- запись
                  end if;
                  if R='1' then
                     do<= X"0000";
                  else
                     do<= RAM(addri);
                  end if;
           end if;
        end process;
    -- тристабильный выходной буфер --------------------
TRI:AD<= To_StdLogicVector(do) when OE='1'
         else "ZZZZZZZZZZZZZZZZ";
end BEH;
```

Для задания памяти требуемого объема был введен тип MEM1KX16, представляющий собой массив 1К шестнадцатиразрядных слов. Константа RAM_init этого типа представляет начальное состояние RAM. Если блок RAM предназначен для хранения программ, то коды этой константы могут быть кодами команд программы, исполняемой после включения процессора.

В процессе RG_ADDR описан регистр адреса, в который записывается адрес чтения или записи с общей входной-выходной шины по сигналу AE. Переменная RAM типа MEM1KX16 является основой блока RAM, описанного в процессе RAM1K. Доступ к ячейкам памяти – элементам массива RAM – выполняется по значению целой переменной addri. Эта переменная получается из вектора адреса addr после преобразования его типа. Так как элементы – битовые векторы, а входные и выходные данные типа STD_LOGIC_VECTOR, то для их преобразования используются функции To_bitvector и To_StdLogicVector из библиотеки IEEE. Как и в блоках памяти ПЛИС, по сигналу R блок RAM выдает нулевое значение.

В операторе с меткой TRI описан выходной буфер с тремя состояниями, который по сигналу OE выдает считанное данной, а иначе – вектор из элементов, означающих третье состояние.

Данное описание относится к моделям, описанным синтезируемым стилем. Но компилятор-синтезатор может выполнить память на отдельных триггерах, что влечет за собой большие аппаратные затраты.

4.2. Структурная модель RAM.

Рассмотрим проектирование RAM на базе PLMT и блоков памяти RAMB4_S16. Блок RAM должен содержать четыре блока памяти RAMB, мультиплексор MUX считанных данных, входы которого соединены с выходами блоков памяти, регистр адреса RGA выходной тристабильный буфер (рис. 3.1.). Дешифратор DC двух старших разрядов адреса выдает отдельный сигнал записи на каждый из блоков RAMB.

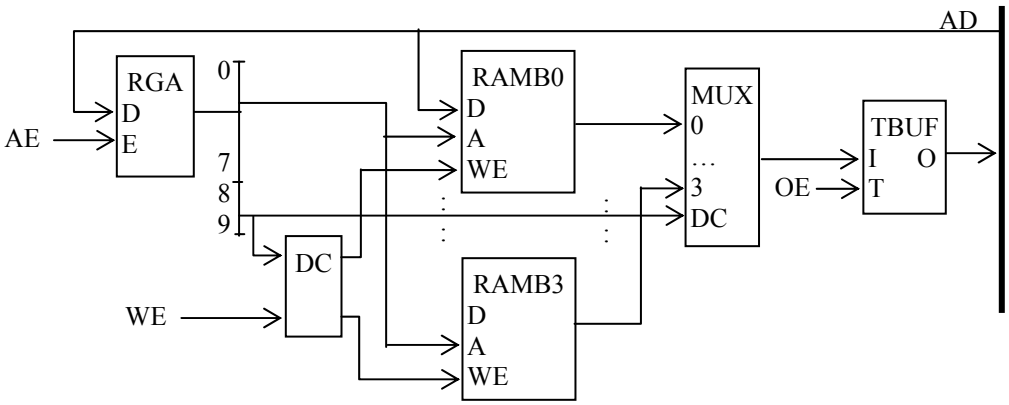

Рис.3.1. Структура блока RAM

Дешифратор DC состоит из четырех PLM, каждая из которых декодирует два разряда адреса. Первая из них описывается следующим образом при C = WE, B=A_9, A=A_8.

```
architecture PLM_DC0 of PLM_3 is
    begin
     Y<=(C and not B and not A) after td;
end PLM_DC0;
```

Остальные PLM дешифратора - PLM_DC1, PLM_DC2,PLM_DC3- описываются аналогично.

Мультиплексор MUX состоит из шестнадцати PLM со следующим описанием при F=D_3, E=D_2, D = D_1, C = D_0, B=A_1, A=A_0.

```
architecture PLM_MUX of PLM_6 is
    begin
     Y<=(C and not B and not A)  -- 0-й вход
         or (D and not B and A)  -- 1-й вход
         or (E and B and not A)  -- 2-й вход
         or (F and B and A)      -- 3-й вход
                    after td;  --задержка элемента
    end PLM_MUX;
```

Структурная модель RAM описывается как следующая архитектура.

```
architecture STR of RAM is
  type DARR_STD is array(0 to 3) of STD_LOGIC_VECTOR(15 downto 0);
   signal do_std:DARR_STD;      --выходы RAM
   constant one:STD_LOGIC:='1';
   constant gnd:bit:='0';  -- нулевой бит
   signal DO0,DO1,DO2,DO3:bit_vector(15 downto 0);--выходы RAM
   signal CLK_std,rst: STD_LOGIC;
   signal we:bit_vector(3 downto 0);
   signal we_std:STD_LOGIC_VECTOR(3 downto 0);
   signal addr: BIT_VECTOR(9 downto 0);
   signal addr_std:STD_LOGIC_VECTOR(9 downto 0);
   signal dom,adi: bit_vector(15 downto 0);
```

```
   signal dom_std:STD_LOGIC_VECTOR(15 downto 0);
   signal oe_std:STD_LOGIC;

   component RAMB4_S16 is  -- блок памяти
    port (DI: in STD_LOGIC_VECTOR (15 downto 0);--входное данное
        EN  : in STD_ULOGIC;  -- разрешение обращения
        WE  : in STD_ULOGIC;  -- сигнал записи
        RST : in STD_ULOGIC;  -- асинхронный сброс
        CLK : in STD_ULOGIC;  -- сигнал синхросерии
        ADDR: in STD_LOGIC_VECTOR (7 downto 0);--адрес
        DO  : out STD_LOGIC_VECTOR (15 downto 0) );--выходное данное
    end component;
    component BUFT is  --тристабильный буфер
    port( O: out   STD_ULOGIC; -- выход
          I: in    STD_ULOGIC; -- вход
          T: in    STD_ULOGIC);-- управляющий вход,если=1,то Z
    end component;
    component FDRE  is port (Q:out  bit; --триггер
          D :   in   bit;
          C :   in   bit;
          CE:   in   bit;
          R :   in   bit);
    end component;

begin
      adi<=To_BitVector(AD);
   RG_A:for i in 0 to 9 generate  --регистр адреса --------------
      U_RGA: FDRE port map(C=>CLK,R=>R,CE=>AE,D=>adi(i),Q=>addr(i));
     end generate;
-- Дешифратор адреса --------------------------------
   U_DC0: entity PLM_3(PLM_DC0)
       port map(C=>WR,B=>addr(9), A=>addr(8),Y=>we(0));
   U_DC1: entity PLM_3(PLM_DC1)
       port map(C=>WR,B=>addr(9), A=>addr(8),Y=>we(1));
   U_DC2: entity PLM_3(PLM_DC2)
       port map(C=>WR,B=>addr(9), A=>addr(8),Y=>we(2));
   U_DC3: entity PLM_3(PLM_DC3)
       port map(C=>WR,B=>addr(9), A=>addr(8),Y=>we(3));

      CLK_std<= To_StdULogic(CLK);
      RST<= To_StdULogic(R);
      addr_std<= To_StdLogicVector(addr);
   RAMS: for i in 0 to 3 generate  -- блоки памяти --------------
         wr_std(i)<= To_StdULogic(WR(i));
     U_RAM: RAMB4_S16 port map (CLK=>CLK_std,RST=>rst, EN=>one,
              WE  =>WR(i),
              DI=>AD,                      --входное данное
              ADDR=> addr_std(7 downto 0),--адрес
              DO  => do_std(i));           --выходное данное
     end generate;
   do0<= To_BitVector(do_std(0));
   do1<= To_BitVector(do_std(1));
   do2<= To_BitVector(do_std(2));
   do3<= To_BitVector(do_std(3));
   oe_std<= not To_StdULogic(OE);
RAM_ENV:for i in 0 to 15 generate
   U_MUX: entity PLM_6(PLM_MUX) -- Мультиплексор --------------
       port map (F=>do3(i),E=>do2(i),D=>do1(i),C=>do0(i),
                 B=>addr(9),A=>addr(8),Y=>dom(i));
       dom_std(i)<= To_StdULogic(dom(i));
   U_TRI: BUFT       -- Тристабильный буфер --------------------
       port map(T=> oe_std,I => dom_std(i), O=>AD(i));
  end generate;
end STR;
```

Поскольку в лабораторных работах основным типом является битовый тип и производные от него типы, а вставляемые компоненты имеют порты типа `Std_Logic`, то в описании RAM многократно применяются функции преобразования типов и из битового в `Std_Logic` и обратно, как, например, `To_StdULogic`, `To_BitVector`. Описание RAM можно было бы сократить, если функции преобразования типов вставлять прямо при связывании сигналов и портов в операторе вставки компонента. Но большинство компиляторов-синтезаторов распознает это как ошибку.

4.3. Испытательный стенд для RAM

Рассмотрим испытательный стенд - объект `RAM_TB` – для архитектуры `RAM(STR)`, у которой эталонной моделью является архитектура `RAM(BEH)`. Испытания заключаются в подаче случайных чисел и сигналов управления на обе модели и сравнении состояния общих шин моделей. При этом 16- разрядные числа AD поступают с генератора случайных чисел на входы обеих моделей через тристабильные буферы, а случайные сигналы WE, OE и AE вырабатываются другим генератором случайных чисел. Для сравнения результатов моделирования используется такой же объект компаратора, как и в лабораторной работе 2.

```
rst<='1', '0' after 220 ns;
wr<= '1' when dc(1 downto 0)="00" else '0';
ae<= '1' when dc(1 downto 0)="01" else '0';
oe<= '1' when dc(1 downto 0)="10" else '0';
AD1<= To_StdLogicvector(di) when oe='0' else (others=>'Z');
AD2<= To_StdLogicvector(di) when oe='0' else (others=>'Z');
```

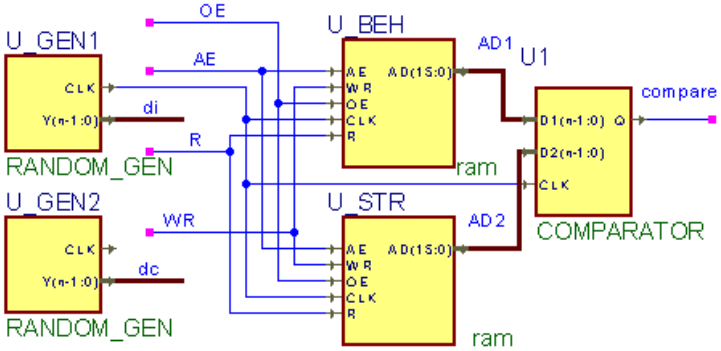

Рис. 3.2. Испытательный стенд для ICTR

5. **Порядок проведения лабораторной работы**

В соответствии с номером варианта, выбирается задание на выполнение лабораторной работы. Параметры задания включают:

- тип логического элемента (PLM или LUT);
- максимальное число термов PLM или количества входов LUT (4 или 5);
- разрядность и объем результирующей схемы RAM;
- количество шин RAM – 1,2 или 3.

Выполнение лабораторной работы имеет 3 стадии: разработка поведенческой модели RAM, разработка структурной модели RAM и разработка испытательного стенда с проверкой функционирования RAM.

5.1. **Разработка поведенческой модели RAM**.

Поведенческая модель RAM описывается с помощью операторов процесса с использованием операций с целыми числами и функций из пакета `Cnetlist` . При этом используется редактор и компилятор VHDL, входящие в состав пакета Active HDL.

После того, как ICTR описана в виде архитектуры ICTR(BEH), она тестируется путем ручной подачи входных тестовых значений при моделировании этой архитектуры.

5.2. **Разработка структурной модели RAM.**

Структурная поведенческая модель RAM описывается структурным стилем, как это показано выше в примере. При этом используются компоненты из файла

`Cnetlist_Lib.VHD`. Для этого также используется редактор и компилятор VHDL, входящие в состав пакета Active HDL.

5.3. **Разработка испытательного стенда и тестирование моделей.**

За образец испытательного стенда берется его пример, описанный в п.4.3. Он дорабатывается под требования конкретного испытуемого объекта.

При тестировании моделей по графикам сигналов определяется правильность функционирования моделей и измеряются задержки сигналов между входами и выходами структурной модели. Полученные графики сигналов переносятся в файл отчета с помощью функций выделения и сохранения в "кармане". По результатам тестирования формулируются выводы по лабораторной работе.

6. Отчет по лабораторной работе.

Отчет по лабораторной работе должен содержать:

- цель работы,
- описание варианта RAM,
- ход синтеза моделей PLM или содержимого LUT,
- тексты описаний поведенческой и структурной моделей RAM,
- графики сигналов, снятых на испытательном стенде,
- значение аппаратных затрат,
- измеренные задержки сигналов,
- выводы.

7. Вопросы по лабораторной работе.

Каково функциональное назначение RAM?

Каким образом управляют режимом работы RAM?

Какими способами задают RAM в проектах на VHDL?

Как проектировать RAM небольшого объема?

Как проектировать RAM объемом больше 1024 бит?

Для чего нужен тип `std_ulogic`?

Объясните работу тристабильной шины.

Покажите два способа задания тристабильного буфера в VHDL.

Как соединять порты и сигналы в проектах VHDL, которые имеют различные типы?

Почему рекомендуется все триггеры RAM подключать к одному источнику синхросерии?