Использование различных типов памяти при проектировании учебного микропроцессорного ядра для реализации в базисе ПЛИС

Андрей СТРОГОНОВ,
д. т. н.
andreis@hotmail.ru
Сергей ЦЫБИН
chipcec@comch.ru

Предлагается повторно переработать проект микропроцессорного ядра из работ [1, 2] в базисе ПЛИС APEX20KE и Stratix III компании Altera с использованием САПР ПЛИС Quartus II версии 8.1, с целью изучения особенностей использования различных видов памяти. В качестве микропроцессорного ядра применяется автомат с циклом работы в два такта из работы [1].

Особенности современной цифровой микроэлектроники обуславливают преимущественное использование синхронных интерфейсов устройств разного типа, в том числе и памяти. При попытке реализовывать асинхронный интерфейс в ряде случаев получаются схемы с различными задержками распространения сигналов («гонки фронтов», «перекосы»), которые не выявляются средствами моделирования, но оказывают существенное негативное влияние на характеристики проекта, загружаемого в ПЛИС, вплоть до неработоспособности схемы или чередующихся неисправностей.

Семейство ПЛИС Stratix III позволяет реализовать на одном кристалле булевы логические функции с помощью адаптивных логических блоков, блоки памяти, при этом память может быть реализована без затрат основной логики, и блоки цифровой обработки сигналов. ПЛИС APEX20KE содержит лишь встроенные блоки памяти (встроенное ОЗУ/ПЗУ).

ПЛИС Stratix имеют структуру памяти TriMatrix, составленную из встроенных блоков памяти RAM трех видов: блоки MLAB (Memory LAB, емкость блока 320 бит, блок может быть представлен как простое двухпортовое ОЗУ); блоки M9K (емкость блока 9216 бит); блоки M144K (емкостью 147 456 бит). Также в ПЛИС Stratix III встроена дополнительная собственная память (встроенное реконфигурируемое ОЗУ). Блоки MLAB необходимы для реализации сдвиговых регистров, буферов FIFO, линий задержек для цифровых фильтров и др. Блоки M9K используются как блоки памяти общего назначения. Блоки M144K нужны для хранения исполняемого кода синтезируемых процессорных ядер, для реализации буферов большого объема в задачах видеообработки сигналов. Все блоки памяти TriMatrix поддерживают синхронный режим работы. Блоки M9K и M144K в ПЛИС Stratix не поддерживают асинхронную память, а блок MLAB поддерживает только асинхронный режим чтения данных. Каждый из блоков памяти с помощью мегафункции altsyncram может быть сконфигурирован как RAM: 1 PORT, RAM: 2 PORT, RAM: 3 PORT, ROM: 1 PORT, ROM: 2 PORT, shift register (RAM-based), FIFO [3].

Рассмотрим два варианта построения микропроцессорного ядра: с асинхронным ПЗУ (рис. 1а) с использованием устаревших серий ПЛИС, например, ПЛИС APEX20KE, и синхронным ПЗУ (рис. 1б) с использованием ПЛИС Stratix III. Вариант микропроцессорного ядра с асинхронным ПЗУ рассмотрен в работе [2] с использованием САПР ПЛИС Quartus II версии 2.0. В обоих случаях емкость ПЗУ — 256 слов на 16 бит, то есть входная адресная шина — 8-разрядная (address [7.. 0]), а выходная шина данных (q [15.. 0]) — 16-разрядная. Файл прошивки ПЗУ приведен в примере 1 (mif — файл).

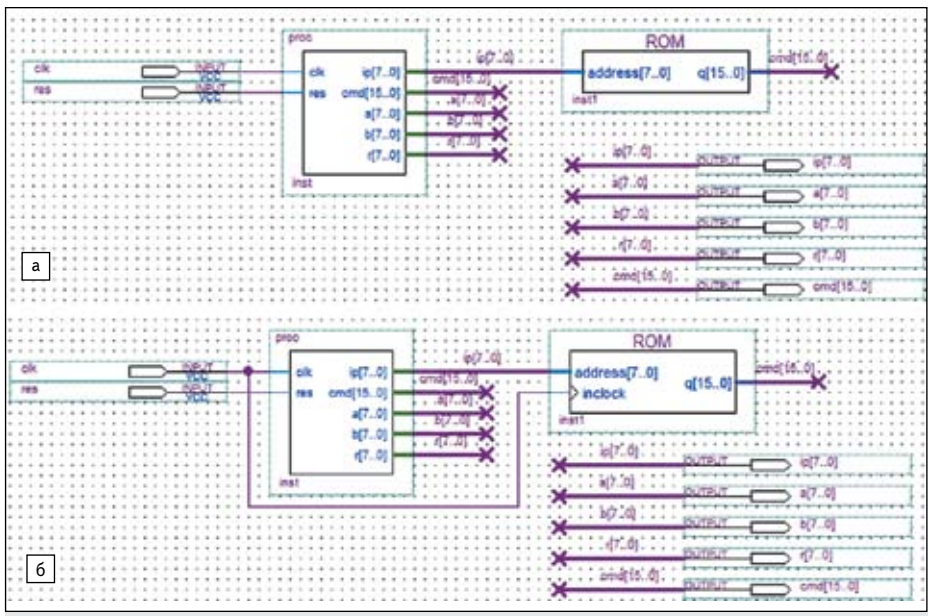

Рис. 1. Микропроцессорное ядро:
а) с асинхронным ПЗУ в базисе ПЛИС APEX20KE; б) с синхронным ПЗУ в базисе ПЛИС Stratix III

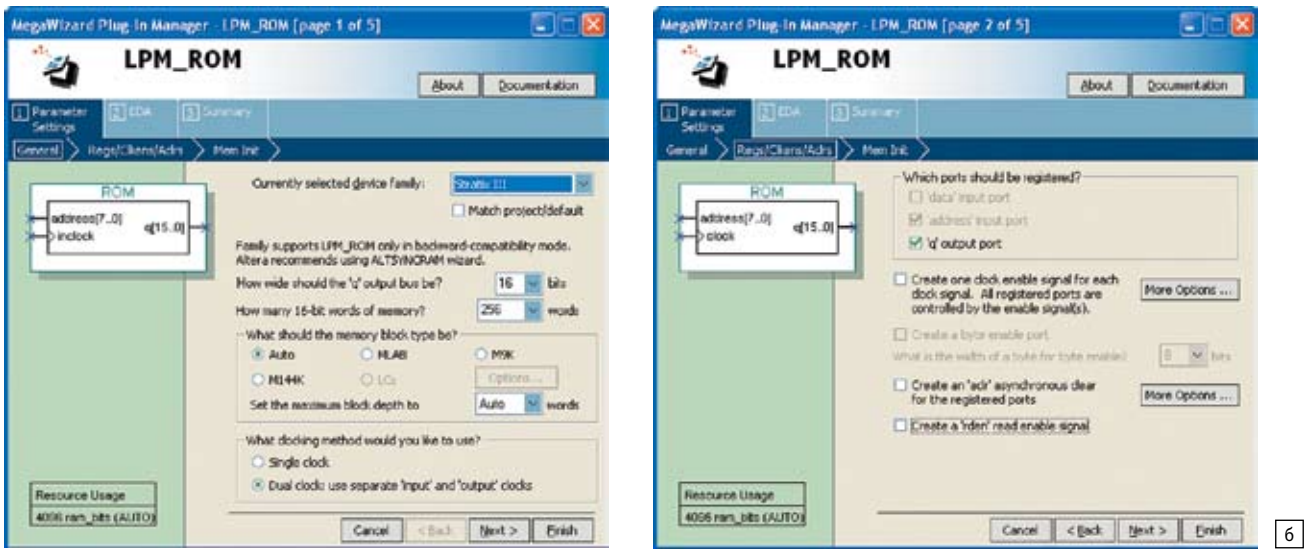

Рис. 2. Менеджер по работе с мегафункцией ПЗУ LPM_ROM ПЛИС Quartus II версии 8.1 для ПЛИС Stratix III в режиме совместимости: а) шаг 1; б) шаг 2

```
-- Quartus II generated Memory Initialization File (.mif)
WIDTH=16;
DEPTH=256;
ADDRESS_RADIX=HEX;
DATA_RADIX=HEX;
CONTENT BEGIN
        000:    0401;
        001:    0511;
        002:    0305;
        003:    0512;
        004:    0402;
        005:    0403;
        006:    0404;
        007:    0604;
        008:    0406;
        009:    0407;
        00A:    0600;
      [00B.. 0FF]:  0000;
END;
```

Пример 1. Файл прошивки ПЗУ

Мегафункция LPM_ROM для ПЛИС Stratix III работает в режиме совместимости. Поэтому в случае повторной переработки проектов, выполненных на устаревших сериях ПЛИС, например, на APEX20KE, при смене серии ПЛИС на Stratix III (рис. 2а, галочка Match project/default снята) возникает предупреждение, что мегафункция LPM_ROM будет сконфигурирована на синхронный режим работы. Возникнет предупреждение о рекомендации к использованию мегафункции altsyncram (ROM: 1 PORT). Фактически произойдет замещение мегафункции LPM_ROM на мегафункцию ROM: 1 PORT.

При настройке мегафункции для ПЛИС на Stratix III пользователь должен самостоятельно выбрать один из типов блоков памяти или выбрать тип Auto. Для ПЛИС APEX20KE доступен только тип Auto (мегафункция LPM_ROM), что объясняется архитектурными особенностями данной ПЛИС, а именно встроенными блоками памяти. Если выбрать тип Auto для ПЛИС Stratix III, то по умолчанию доступны 4096 бит памяти (это зависит от разрядности адресной и выходной шины данных).

Галочку с 'address' input port снять нельзя, то есть адресный порт по умолчанию настраивается как регистр. (На условном обозначении появляется указатель динамического входа — треугольник, то есть адресация к словам ПЗУ осуществляется по переднему фронту синхроимпульса inclock, а выходной порт q — асинхронный) (рис. 2б, мегафункции LPM_ROM для ПЛИС Stratix III, шаг 2). Это говорит о том, что ПЗУ будет сконфигурировано на синхронный режим работы. По желанию, работу выходного порта q можно также синхронизовать сигналом outclock, таким образом можно реализовать режим двойного тактирования.

Использование внутренних триггеров адаптивных логических модулей в качестве памяти ПЗУ в мегафункции для ПЛИС на Stratix III недопустимо, поэтому опция LC (логические элементы, задействуется основная логика: таблицы перекодировок и внутренние триггеры логических элементов) недоступна. Опция LC доступна при разработке ОЗУ без предварительной конфигурации. В случае выбора опции MLAB и режима Auto доступны в качестве памяти 20 таблиц перекодировок (LUT-таблица, таблица перекодировок или логическая таблица, служит для реализации комбинационных функций и может быть упрощенно представлена как столбец ОЗУ с мультиплексором) + 7 MLAB + 24 триггера. При использовании одного из трех типов блоков памяти MLAB, M9K и M144K дополнительный режим Auto

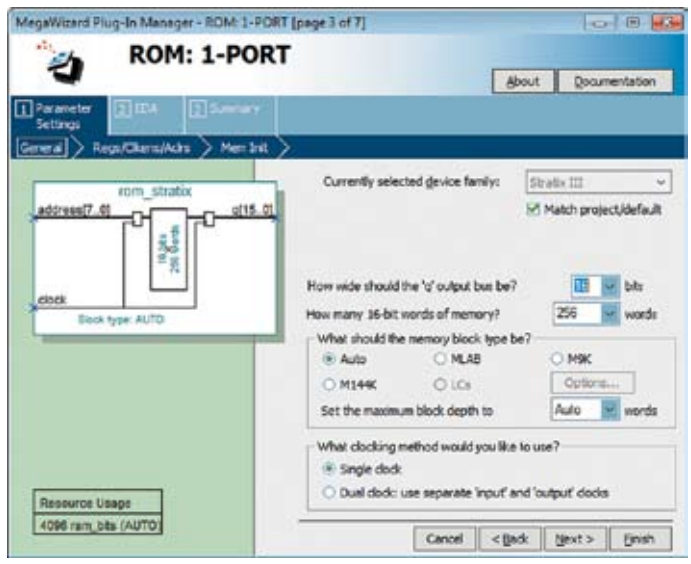

Рис. 3. Менеджер по работе с мегафункцией однопортовой памяти ПЗУ LPM_ROM ПЛИС Quartus II версии 8.1 для ПЛИС Stratix III (ROM: 1 PORT)

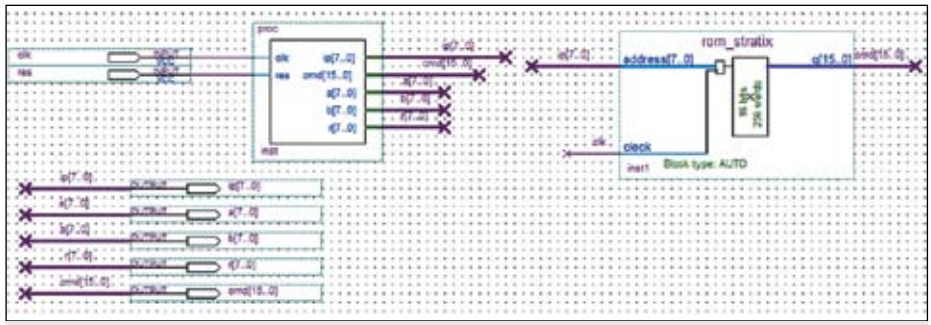

Рис. 4. Микропроцессорное ядро с синхронным ПЗУ (синхронный режим адресации и асинхронный режим чтения) в базисе ПЛИС Stratix III, созданное с помощью мегафункции ROM: 1 PORT

назначает максимально доступный к использованию объем памяти. Число используемых слов для каждого блока памяти может быть ограничено пользователем.

Чтобы избежать данного предупреждения и повторно переработать проект с использованием мегафункции однопортового ПЗУ (ROM: 1 PORT) для ПЛИС Stratix III, необходимо удалить из проекта блок памяти ПЗУ, созданный с помощью мегафункции LPM_ROM, и заново с помощью «мастера» MegaWizard Plugin сгенерировать блок памяти (рис. 3), сменив предварительно в Assignments/Device серию ПЛИС APEX20KE на Stratix III. На рис. 4 показано микропроцессорное ядро с синхронным ПЗУ в базисе ПЛИС Stratix III, созданное с помощью мегафункции ROM: 1 PORT. Сравнивая рис. 2 и 3, видим, что условное обозначение мегафункции ROM: 1 PORT несколько отличается от обозначения мегафункции LPM_ROM в режиме совместимости. Это означает, что все входы в память и выходы из нее «защелкиваются» в регистрах.

На рис. 5 и 6 показано функциональное и временное моделирование работы микропроцессорного ядра с асинхронным ПЗУ в базисе ПЛИС APEX20KE (а) и синхронным (б) ПЗУ в базисе ПЛИС Stratix III с использованием мегафункции ПЗУ LPM_ROM САПР Quartus II версии 8.1.

Для того чтобы обеспечить переносимость проекта с одной серии ПЛИС на другую без использования мегафункций, рассмотрим использование языка VHDL для проектирования ПЗУ. Пример 2 демонстрирует проектирование асинхронного ПЗУ на языке VHDL с использованием элементов поведенческого описания (используются абстрактные логические структуры, такие как циклы и процессы). На рис. 7 показаны результаты функционального моделирования в базисе ПЛИС APEX20KE. Пример 3 демонстрирует проектирование синхронного ПЗУ на языке VHDL [4], а на рис. 8 показаны результаты функционального

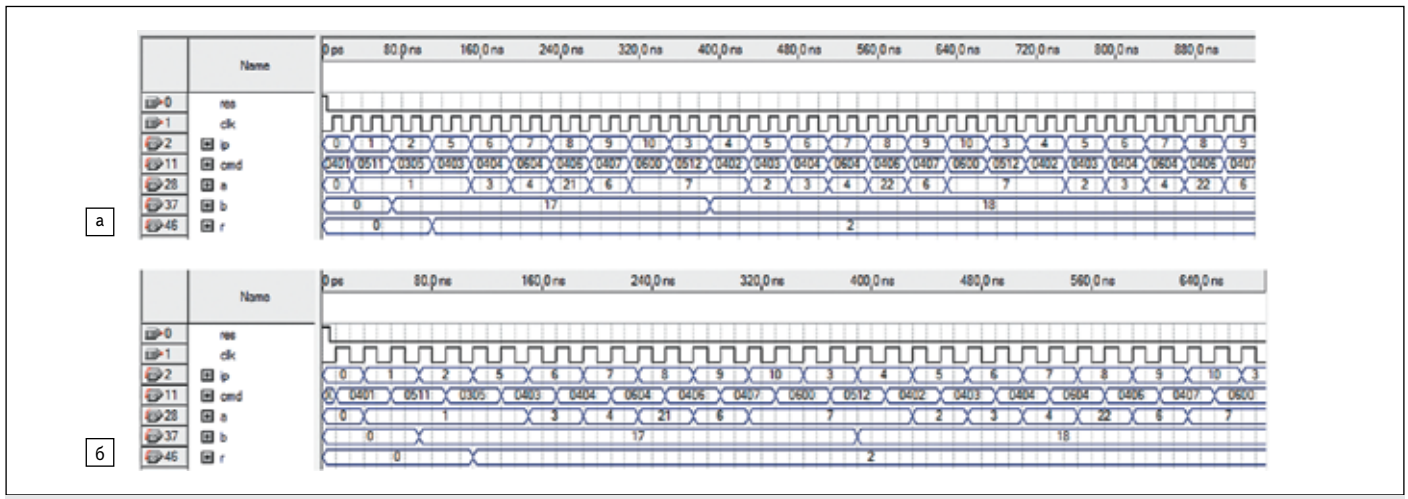

Рис. 5. Функциональное моделирование работы микропроцессорного ядра: а) с асинхронным ПЗУ в базисе ПЛИС APEX20KE; б) с синхронным ПЗУ в базисе ПЛИС Stratix III

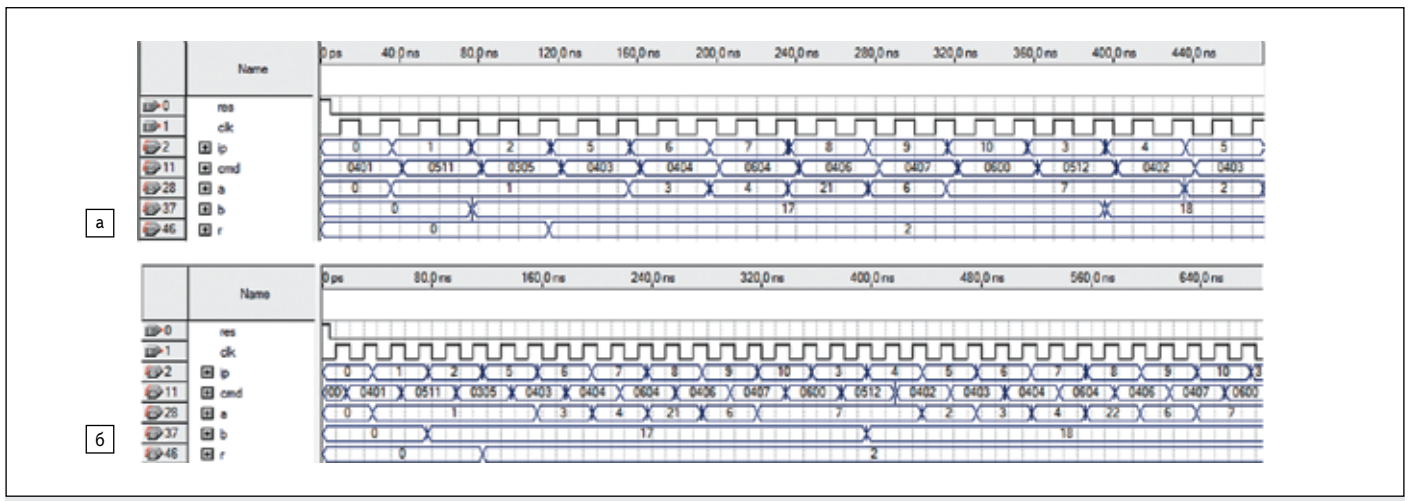

Рис. 6. Временное моделирование работы микропроцессорного ядра: а) с асинхронным ПЗУ в базисе ПЛИС APEX20KE; б) с синхронным ПЗУ в базисе ПЛИС Stratix III

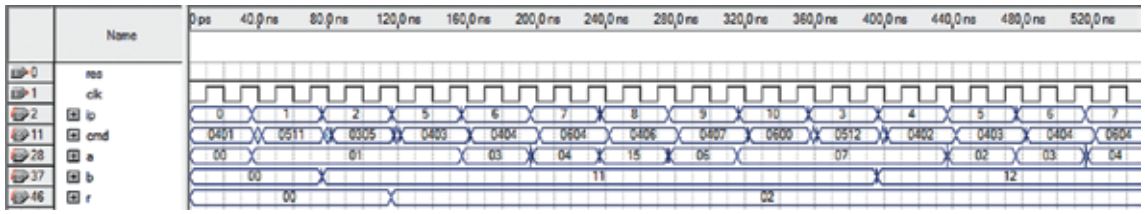

Рис. 7. Функциональное моделирование работы микропроцессорного ядра с асинхронным ПЗУ на языке VHDL (ПЛИС APEX20KE)

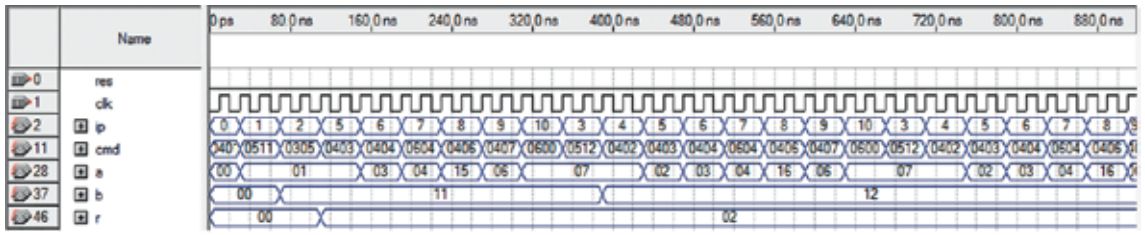

Рис. 8. Функциональное моделирование работы микропроцессорного ядра с синхронным ПЗУ на языке VHDL (ПЛИС Stratix III)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
ENTITY rom_syn IS
   PORT (
      clk: IN std_logic;
      addr: IN std_logic_vector (7 DOWNTO 0);
      rom_out: OUT std_logic_vector (15 DOWNTO 0));
END rom_syn;
ARCHITECTURE a OF rom_syn IS
   TYPE T_UFIX_16_256 IS ARRAY (255 DOWNTO 0) of unsigned (15 DOWNTO 0);
BEGIN
   PROCESS (addr)
      -- local variables
      VARIABLE data_temp: T_UFIX_16_256;
   BEGIN
      FOR b IN 0 TO 255 LOOP
         data_temp (b):= to_unsigned (0, 16);
      END LOOP;
      data_temp (0)    := to_unsigned (1025, 16);
      data_temp (1)    := to_unsigned (1297, 16);
      data_temp (2)    := to_unsigned (773, 16);
      data_temp (3)    := to_unsigned (1298, 16);
      data_temp (4)    := to_unsigned (1026, 16);
      data_temp (5)    := to_unsigned (1027, 16);
      data_temp (6)    := to_unsigned (1028, 16);
      data_temp (7)    := to_unsigned (1540, 16);
      data_temp (8)    := to_unsigned (1030, 16);
      data_temp (9)    := to_unsigned (1031, 16);
      data_temp (10)   := to_unsigned (1536, 16);
   rom_out <= std_logic_vector (data_temp (to_integer (unsigned (addr))));
   END PROCESS;
END a;
```

Пример 2. Описание асинхронного ПЗУ на языке VHDL

моделирования в базисе ПЛИС Stratix III. Проект микропроцессора с синхронным ПЗУ на языке VHDL работоспособен как на старых, так и на новых сериях ПЛИС фирмы Altera. Пример 4 показывает описание асинхронного ПЗУ с использованием элементов потокового описания (представление на уровне регистровых передач). Оператор case обеспечивает параллельную обработку и используется для выбора одного варианта из нескольких в зависимости от условий.

Отредактируем исходный код языка VHDL управляющего автомата микропроцессорного ядра, приведенного в работе [1], с использованием перечислимых типов. Применение этих типов преследует две цели: улучшение смысловой читаемости программы, а также более четкий и простой визуальный контроль значений. Наиболее часто перечислимый тип используется для обозначения состояний конечных автоматов.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.numeric_std.all;
ENTITY ozu_sin IS
   PORT (
      clk : IN std_logic;
      reset : IN std_logic;
      addr : IN std_logic_vector(7 DOWNTO 0);
      rom_out : OUT std_logic_vector(15 DOWNTO 0));
END ozu_sin;
ARCHITECTURE a OF ozu_sin IS
TYPE T_UFIX_16_256 IS ARRAY (255 DOWNTO 0) of unsigned (15 DOWNTO 0);
SIGNAL data, data_next: T_UFIX_16_256;
BEGIN
   PROCESS (reset, clk)
      VARIABLE b : INTEGER;
   BEGIN
      IF reset = '1' THEN
         FOR b IN 0 TO 255 LOOP
            data(b) <= to_unsigned(0, 16);
         END LOOP;
      ELSIF clk'EVENT AND clk= '1' THEN
         FOR b IN 0 TO 255 LOOP
            data(b) <= data_next(b);
         END LOOP;
      END IF;
   END PROCESS;
   PROCESS (addr)
      -- local variables
   VARIABLE data_temp : T_UFIX_16_256;
   BEGIN
      FOR b IN 0 TO 255 LOOP
         data_temp(b) := to_unsigned(0, 16);
      END LOOP;
      data_temp(0)    := to_unsigned(1025, 16);
      data_temp(1)    := to_unsigned(1297, 16);
      data_temp(2)    := to_unsigned(773, 16);
      data_temp(3)    := to_unsigned(1298, 16);
      data_temp(4)    := to_unsigned(1026, 16);
      data_temp(5)    := to_unsigned(1027, 16);
      data_temp(6)    := to_unsigned(1028, 16);
      data_temp(7)    := to_unsigned(1540, 16);
      data_temp(8)    := to_unsigned(1030, 16);
      data_temp(9)    := to_unsigned(1031, 16);
      data_temp(10)   := to_unsigned(1536, 16);
      rom_out <= std_logic_vector(data_temp(to_integer(unsigned(addr))));
      FOR c IN 0 TO 255 LOOP
         data_next(c) <= data_temp(c);
      END LOOP;
   END PROCESS;
END a;
```

Пример 3. Описание синхронного ПЗУ на языке VHDL (поведенческий уровень)

Перечислимый тип объявляется путем перечисления названий элементов-значений. Объекты, тип которых объявлен как перечислимый, могут содержать только те значения, которые указаны

при перечислении, следовательно, количество всех возможных значений конечно. Объявление перечислимого типа имеет вид:

```
TYPE имя_типа IS (название_элемента [, название_элемента]);
Type State_type IS (stateA, stateB, stateC);
VARIABLE State: State_type;
.
State:= stateB;
```

В данном примере объявляется переменная State, допустимыми значениями которой являются stateA, stateB, stateC. Пример 5 демонстрирует использование перечислимого типа для проектирования управляющего автомата.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity proc is
port (ip: inout std_logic_vector(7 downto 0);
      cmd: inout std_logic_vector(15 downto 0);
      clk,res: in std_logic;
      a: inout std_logic_vector(7 downto 0);
      b,r: inout std_logic_vector(7 downto 0));
end proc;
architecture a of proc is

TYPE state_values IS (st0, st1);
signal state, next_state: state_values;

begin
process(clk)
begin
if (res = '1') then
          a <="00000000";
          b <="00000000";
          ip <="00000000";
          r <="00000000";
elsif clk'event and clk='1' then
case state is
when st0=> next_state <=st1;
case conv_integer(cmd) is
when 0=> ip <= ip+1;
when 256 to 511 =>ip<=cmd(7 downto 0);
when 512 to 767 =>if conv_integer(a)=0
                  then ip<=cmd(7 downto 0);
                  else ip<=ip+1;
                  end if;
when 768 to 1023 =>r<=ip; ip<=cmd(7 downto 0);
when 1024 to 1279 => a<=cmd(7 downto 0); ip<=ip+1;
when 1280 to 1535 => b<=cmd(7 downto 0); ip<=ip+1;
when 1536 =>ip<=r+1;
when 1537=>a<=b; ip<=ip+1;
when 1538=>b<=a; ip<=ip+1;
when 1539=>a<=b;b<=a; ip<=ip+1;
when 1540=>a<=a+b; ip<=ip+1;
when 1541=>a<=a-b; ip<=ip+1;
when 1542=>a<=a and b; ip<=ip+1;
when 1543=>a<=a or b; ip<=ip+1;
when 1544=>a<=a xor b; ip<=ip+1;
when 1545=>a<=a-1; ip<=ip+1;
when others=>ip<=ip+1;
end case;
when st1=> next_state<=st0;
end case;
end if;
end process;
end a;
```

Пример 5. Описание управляющего автомата микропроцессорного ядра с использованием перечислимого типа на языке VHDL с циклом работы в два такта

Изучая рис. 5–8, видим, что функциональное и временное моделирование подтверждают правильность работы микропроцессорного ядра, реализованного в базисах ПЛИС APEX20KE и Stratix III САПР ПЛИС Quartus II версии 8.1. Независимо от типа используемой памяти (синхронный или асинхронный режим работы) процессор работает

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.numeric_std.ALL;

ENTITY ROM IS
   PORT( clk : IN std_logic;
         Reset : IN std_logic;
         Addr : IN std_logic_vector(7 DOWNTO 0);
         Cmd : OUT std_logic_vector(15 DOWNTO 0)
         );
END ROM;
ARCHITECTURE rtl OF ROM IS
   -- Signals
   SIGNAL s : unsigned(7 DOWNTO 0);
   SIGNAL Lookup_Table_out1 : unsigned(15 DOWNTO 0);
BEGIN
   s <= unsigned(addr);
   PROCESS(s)
   BEGIN
      CASE s IS
         WHEN "00000000" => Lookup_Table_out1 <= "0000010000000001";
         WHEN "00000001" => Lookup_Table_out1 <= "0000010100010001";
         WHEN "00000010" => Lookup_Table_out1 <= "0000001100000101";
         WHEN "00000011" => Lookup_Table_out1 <= "0000010100010010";
         WHEN "00000100" => Lookup_Table_out1 <= "0000010000000010";
         WHEN "00000101" => Lookup_Table_out1 <= "0000010000000011";
         WHEN "00000110" => Lookup_Table_out1 <= "0000010000000100";
         WHEN "00000111" => Lookup_Table_out1 <= "0000011000000100";
         WHEN "00001000" => Lookup_Table_out1 <= "0000010000000110";
         WHEN "00001001" => Lookup_Table_out1 <= "0000010000000111";
         WHEN "00001010" => Lookup_Table_out1 <= "0000011000000000";
         WHEN OTHERS => Lookup_Table_out1 <= "0000000000000000";
      END CASE;
   END PROCESS;
   cmd <= std_logic_vector(Lookup_Table_out1);
END rtl;
```

Пример 4. Описание асинхронного ПЗУ на языке VHDL (уровень регистровых передач)

Таблица. Используемые ресурсы ПЛИС Stratix III EP3SL50F484C2

Проект	Логическое приложение		Полный объем памяти, бит
	Комбинационные ALUT	Специализированные логические регистры	
Синхронное ПЗУ, ПЛИС Stratix III, мегафункция LPM_ROM, тип Auto	109	33	4096
Синхронное ПЗУ*, ПЛИС APEX20KE EP20K30ETC144, мегафункция LPM_ROM, тип Auto	188 LE **		4096
Синхронное ПЗУ, ПЛИС Stratix III, мегафункция ROM: 1 PORT, тип Auto	109	33	4096
Синхронное ПЗУ, ПЛИС Stratix III, мегафункция ROM: 1 PORT, тип Auto (управляющий автомат, пример 5)	94	32	4096
Асинхронное ПЗУ на языке VHDL (пример 4)	111	33	–
Синхронное ПЗУ на языке VHDL (управляющий автомат, пример 5)	113	33	–

Примечание. * — приводится для сравнения;
** — логический элемент устаревших серий ПЛИС, содержащий 4-входовую LUT и программируемый регистр.

в два такта. По первому такту синхроимпульса происходит выборка и дешифрование команды, а по второму такту — непосредственная отработка команды, например, выдача результатов в регистры общего назначения или загрузка новых команд, как в случае с «прыжковыми» командами, к примеру, команды CALL или RET.

Для разработки микропроцессорных ядер в базисе ПЛИС Stratix III в качестве синхронного ПЗУ необходимо использовать универсальную мегафункцию altsyncram, которая может быть сконфигурирована как ROM: 1 PORT (мегафункция altsyncram может быть использована и для конфигурирования ОЗУ), что позволяет снизить нагрузку по числу используемых логических элементов (таблица) и обеспечить синхронный режим работы ПЗУ. Если управляющий автомат и ПЗУ будут спроектированы с использованием языка VHDL, то в этом случае будет задействована основная логика: таблицы перекодировок (ALUT) и внутренние триггеры логических элементов (logic registers). Из анализа представленной таблицы можно сделать вывод, что в простейшем случае наиболее оптимальным оказывается использование языка VHDL, а блоки памяти используются неэффективно.

Литература

1. Тарасов И. Проектирование конфигурируемых процессоров на базе ПЛИС. Часть II // Компоненты и технологии. 2006. № 3.
2. Строгонов А. Проектирование учебного процессора для реализации в базисе ПЛИС // Компоненты и технологии. 2009. № 3.
3. www.altera.com. Stratix III Device Handbook, vol 1, software version 9.0, document version 1.9, July 2009.
4. Строгонов А., Буслов А. Проектирование учебного процессора для реализации в базисе ПЛИС с использованием системы MATLAB/Simulink // Компоненты и технологии. 2009. № 5.