Проектирование микропроцессорных ядер в САПР ПЛИС WebPACK ISE фирмы Xilinx

Андрей СТРОГОНОВ, д. т. н.
andreis@hotmail.ru
Максим МОТЫЛЕВ
polikort@rambler.ru
Артем БУСЛОВ
buslov-artem@yandex.ru

Предлагается повторно переработать проект микропроцессорного ядра из работ [1, 2] в базисе ПЛИС фирмы Xilinx с использованием САПР WebPACK ISE, с целью изучения эффективности упаковки логики ПЛИС различных технологических поколений и архитектур фирм Xilinx и Altera соответствующими средствами САПР. В основе микропроцессорного ядра используется управляющий автомат с циклом работы в два такта из [3, 4]. В качестве интегральной оценки эффективности использования ресурсов ПЛИС может выступать коэффициент заполнения кристалла микропроцессорным ядром или его время компиляции [5, 6]. На эффективность упаковки логики ПЛИС наиболее значимое влияние оказывает архитектура межсоединений.

Реализуем микропроцессорное ядро в ПЛИС Spartan II фирмы Xilinx с использованием структурного описания на языке VHDL. Оператор component является параллельным. Синтаксис оператора:

```
label: component-name port map (port1=>signal1, port=>signal2,
…, portn=>signaln);
```

Здесь ***component-name*** — имя объекта, который должен быть использован в теле архитектуры. Для каждого оператора component с обращением к названному объекту создается одна копия этого объекта, каждая копия должна иметь уникальную метку ***label***. Ключевое слово ***port map*** вводит список, посредством которого портам названного объекта ставятся в соответствие сигналы данной архитектуры. Каждый порт объекта связывается с сигналом посредством оператора "=>", соответствия могут следовать в любом порядке.

До обработки внутри архитектуры компонент должен быть декларирован объявлением компонента в определении архитектуры. Синтаксис объявления компонента на языке VHDL:

```
component component-name
port ( signal-names: mode signal-type;
       signal-names: mode signal-type;
       ….
       signal-names: mode signal-type);
end component;
```

Используемые в архитектуре компоненты должны быть заранее определены или быть библиотечными элементами.

Пример 1 демонстрирует структурное описание проекта на языке VHDL. Описание управляющего автомата с циклом работы в два такта приведено в работе [1]. Пример 2 показывает содержимое прошивки асинхронного ПЗУ, используемое при компиляции проекта в различные серии ПЛИС. На рис. 1 показана структурная схема микропроцессорного ядра в редакторе ***Project Navigator*** САПР ПЛИС Xilinx ISE 9.1i.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity cpu is
Port ( cmd : out STD_LOGIC_VECTOR (15 downto 0);
       clk : in STD_LOGIC;
       res : in STD_LOGIC;
       a : out STD_LOGIC_VECTOR (7 downto 0);
       b : out STD_LOGIC_VECTOR (7 downto 0);
       r : out STD_LOGIC_VECTOR (7 downto 0));
end cpu;
architecture Behavioral of cpu is
component proc
port ( ip : out STD_LOGIC_VECTOR (7 downto 0);
       cmd : in STD_LOGIC_VECTOR (15 downto 0);
       clk : in STD_LOGIC;
       res : in STD_LOGIC;
       a : inout STD_LOGIC_VECTOR (7 downto 0);
       b : inout STD_LOGIC_VECTOR (7 downto 0);
       r : inout STD_LOGIC_VECTOR (7 downto 0));
end component;
component pzu
port ( q : out STD_LOGIC_VECTOR (15 downto 0);
       addr : in STD_LOGIC_VECTOR (7 downto 0));
end component;

signal cmd_1 : STD_LOGIC_VECTOR (15 downto 0);
signal addr_1 : STD_LOGIC_VECTOR (7 downto 0);
signal a_1 : STD_LOGIC_VECTOR (7 downto 0);
signal b_1 : STD_LOGIC_VECTOR (7 downto 0);
signal r_1 : STD_LOGIC_VECTOR (7 downto 0);
begin
U1: proc port map (ip => addr_1, cmd => cmd_1, clk => clk, res => res,
a => a_1, b => b_1, r => r_1);
U2: pzu port map (addr => addr_1, q => cmd_1);
       a <= a_1;
       b <= b_1;
       r <= r_1;
       cmd <= cmd_1;
end Behavioral;
```

Пример 1. Структурный стиль описания микропроцессорного ядра на языке VHDL

```
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 2. Файл прошивки асинхронного ПЗУ

Проведем верификацию проекта с использованием симулятора языка VHDL ModelSim SE 6.0. Для этого, используя редактор ***Project Navigator***, для модуля верх-

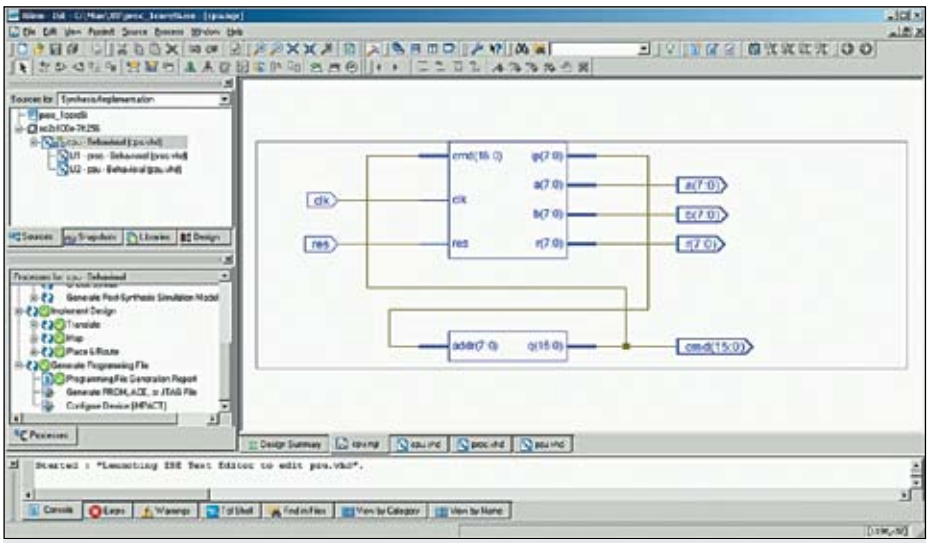

Рис. 1. Структурная схема микропроцессорного ядра в редакторе Project Navigator САПР ПЛИС Xilinx ISE 9.1i

него уровня проекта создадим тестовый файл ***VHDL Test Bench***. В нем зададим сигнал тактирования и, используя конструкцию языка ***VHDL "wait for"***, опишем сигналы на входах автомата в интересующим промежутке времени (пример 3).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
ENTITY test_cpu IS
END test_cpu;
ARCHITECTURE behavior OF test_cpu IS
        COMPONENT cpu
    PORT(
        cmd : OUT std_logic_vector(15 downto 0);
        clk : IN std_logic;
        res : IN std_logic;
        a : OUT std_logic_vector(7 downto 0);
        b : OUT std_logic_vector(7 downto 0);
        r : OUT std_logic_vector(7 downto 0));
    END COMPONENT;
      --Inputs
      signal clk : std_logic := '0';
      signal res : std_logic := '0';
      --Outputs
      signal cmd : std_logic_vector(15 downto 0);
      signal a : std_logic_vector(7 downto 0);
      signal b : std_logic_vector(7 downto 0);
      signal r : std_logic_vector(7 downto 0);
BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: cpu PORT MAP (
            cmd => cmd,
            clk => clk,
            res => res,
            a => a,
            b => b,
            r => r
            );
      -- Clock process definitions
process
        begin
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
end process;
      -- Stimulus process
      stim_proc: process
      begin
        wait for 10 ns;
        res <= '0';
        wait;
      end process;
END;
```

Пример 3. Тестовый файл для моделирования микропроцессорного ядра в симуляторе языка ModelSim 6.0

С помощью тестового файла промоделируем проект в среде ModelSim SE PLUS 6.0 фирмы Mentor Graphics Corporation и убедимся в правильности функционирования проектируемого устройства (рис. 2).

Для оценки эффективности упаковки логики ПЛИС зарубежных фирм Altera и Xilinx с архитектурами FPGA и CPLD, САПР Quartus II Version 8.1 и Xilinx ISE 9.1i, скомпилируем проект в кристаллы с архитектурой FPGA — APEX 20K EP20K30E (Altera), Stratix III EP3SL50 (Altera), Spartan-IIE XC2S100E (Xilinx), Virtex5 XC5vlx30 (Xilinx) и с архитектурой CPLD — Max II EPM570ZM100C6 (Altera) и CoolRunner2 XC2C384 (Xilinx). Сравнительные характеристики исследуемых ПЛИС отображены в таблице 1. Подробное сравнение ресурсов ПЛИС архитектур FPGA и CPLD фирм Altera и Xilinx приведено на сайте [7].

Трассировочная архитектура MultiTrack, используемая в ПЛИС Stratix III, обеспечивает связь между различными кластерами логических элементов и характеризуется определенным количеством шагов (hop), необходимых для того, чтобы соединить один конфигурируемый логический блок (LAB) с другим. Чем меньше количество шагов, тем выше производительность и легче оптимизация архитектуры с помощью инструментов САПР. Архитектура трассировки межсоединений MultiTrack обеспечивает бóльшую доступность ко всем окружающим КЛБ с помощью меньшего числа связей, что позволяет увеличить производительность, снизить энергопотребление и оптимизировать упаковку логики.

Результаты размещения проекта в кристаллы различных архитектур и производителей показаны в таблицах 1, 2 и отображены на рис. 3–6. В таблицах 1 и 2 используются следующие условные обозначения: ЛЭ — логический элемент; ТП — таблицы перекодировок (LUT-таблицы); МЯ — макроячейки; ВБП — встроенные блоки памяти.

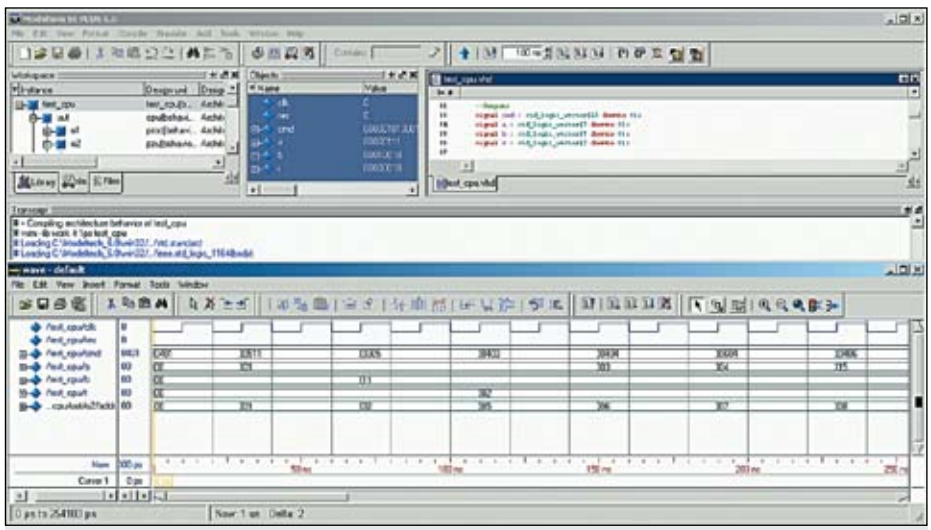

Рис. 2. Временные диаграммы работы микропроцессорного ядра в симуляторе языка VHDL ModelSim SE PLUS 6.0

Таблица 1. Функциональные характеристики ПЛИС фирм Altera и Xilinx

Фирма	Altera			Xilinx		
Архитектура	FPGA		CPLD	FPGA		CPLD
Серия и название ПЛИС	APEX20K EP20K30E	Stratix III EP3SL50	Max II EPM570ZM100C6	Spartan-IIE XC2S100E	Virtex5 XC5vlx30	CoolRunner2 XC2C384
Напряжение питания ядра, В	1,8	1,1	1,8	1,8	1	1,8
ЛЭ/МЯ*	1200	47 500	–/440	1536	9600	–/384

Примечание. * — для ПЛИС с архитектурой CPLD

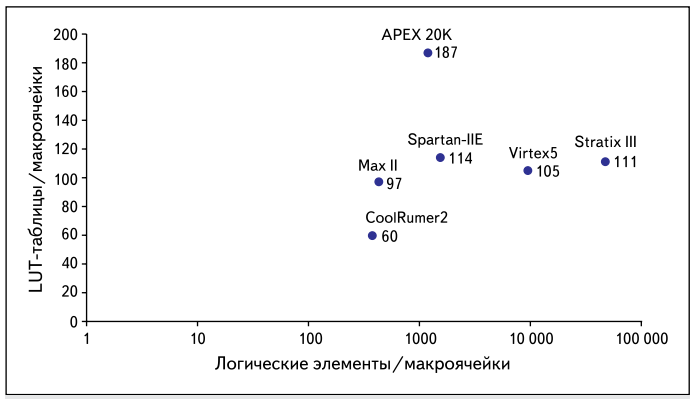

Рис. 3. Задействованные ресурсы ПЛИС фирм Altera (САПР Quartus II Version 8.1) и Xilinx (САПР ISE 9.1i) для реализации комбинационных функций при компиляции микропроцессорного ядра

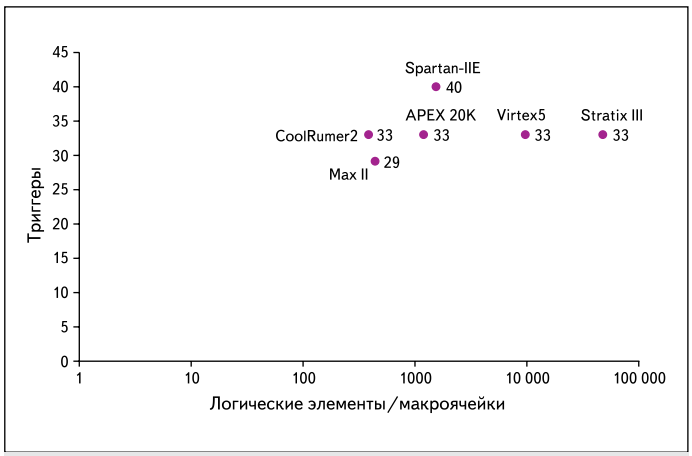

Рис. 4. Задействованные ресурсы ПЛИС фирм Altera (САПР Quartus II Version 8.1) и Xilinx (САПР ISE 9.1i) для реализации последовательностных функций при компиляции микропроцессорного ядра

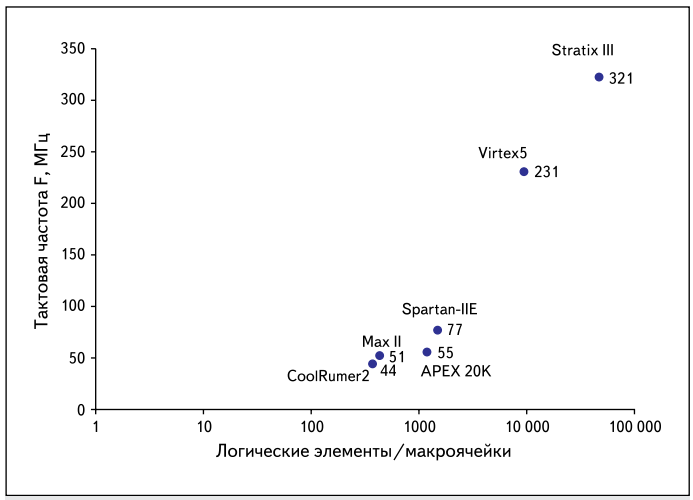

Рис. 5. Максимальная тактовая частота для работы проекта микропроцессорного ядра в базисе ПЛИС фирм Altera (САПР Quartus II Version 8.1) и Xilinx (САПР ISE 9.1i)

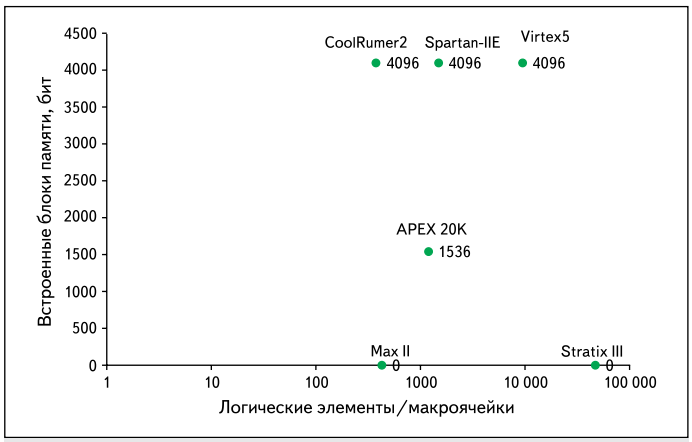

Рис. 6. Задействованные ресурсы встроенной памяти ПЛИС фирм Altera (САПР Quartus II Version 8.1) и Xilinx (САПР ISE 9.1i) микропроцессорного ядра

Проанализировав полученные результаты, можно сделать вывод, что для проектирования встраиваемой микропроцессорной системы на основе пользовательского микропроцессорного ядра с управляющим автоматом с циклом работы в два такта оптимальными характеристиками обладает семейство ПЛИС Virtex5 с архитектурой FPGA, а из архитектур CPLD — ПЛИС Max II. По эффективности упаковки логики наилучшие характеристики обеспечивает ПЛИС Stratix III.

Таблица 2. Ресурсы ПЛИС, задействованные при компиляции проекта микропроцессорного ядра

Семейство и название кристалла	APEX 20K EP20K30E	Stratix III EP3SL50	Max II 570ZM100C6	Spartan-IIE XC2S100E	Virtex5 XC5vlx30	CoolRunner2 XC2C384
ТП/МЯ*	187	111	97	114	105	–/60
D-триггеры	33	33	29	40	33	33
Частота, МГц	55	321	51	77	231	44
ВБП, бит	1536	0	0	4096	4096	4096

Примечание. * — CPLD семейства MAX II имеют LUT-based архитектуру.
Эквивалентный логический объем CPLD MAX II в макроячейках пересчитывается из объема в ЛЭ с коэффициентом 1,3 (например, для EPM570: 440 Typical Macrocell = 570 Logical Elements/1,3)

Литература

1. Строгонов А. Проектирование учебного процессора для реализации в базисе ПЛИС // Компоненты и технологии. 2009. № 3.
2. Строгонов А., Цыбин С. Использование различных типов памяти при проектировании учебного микропроцессорного ядра для реализации в базисе ПЛИС // Компоненты и технологии. 2009. № 12.
3. Тарасов И. Проектирование конфигурируемых процессоров на базе ПЛИС. Часть I // Компоненты и технологии. 2006. № 2.
4. Тарасов И. Проектирование конфигурируемых процессоров на базе ПЛИС. Часть II // Компоненты и технологии. 2006. № 3.
5. Verma S. How to perform meaningful benchmarks on FPGAs from different vendors — www.pldesignline.com
6. OpenCore Stamping and Benchmarking Methodology — www.altera.com
7. www.altera.ru