УДК 004.3

С.Б. ОСТРОУМОВ[1], Ю.Н. ПРОХОРОВА[1], А.А. АНДРАШОВ[1], А.Д. ГЕРАСИМЕНКО[2]

[1] ***Национальный аэрокосмический университет им. Н.Е. Жуковского «ХАИ», Харьков***
[2] ***Научно-производственное предприятие «Радий», Кировоград***

О ТЕСТИРОВАНИИ ПРОГРАММНО-АППАРАТНЫХ СРЕДСТВ ДЛЯ ПЛИС-ОРИЕНТИРОВАННЫХ КРИТИЧЕСКИХ ПРИЛОЖЕНИЙ

Предлагаются элементы методики тестирования компонентов Quartus и Nios фирмы Altera, используемых для разработки ПЛИС проектов критического применения, а также статического анализа программного кода, реализующего логику технологического процесса на ядре Nios. Анализируются результаты тестирования компонентов, используемых при создании ПТК ИУС АЭС.

ПТК ИУС АЭС, Nios, Quartus II, системы автоматизированного проектирования (САПР)

Введение

В связи с расширением применения ПЛИС в критических приложениях (так, на предприятии «Радий» за последние 5 лет введено в эксплуатацию более 3300 микросхем) актуальными являются вопросы тестирования компонентов, используемых при проектировании таких систем. Естественно, существует некоторая степень доверия к фирмам-лидерам в области производства и продаж микросхем ПЛИС (Altera, Xilinx и др.). Однако для критических систем (системы аварийной защиты атомного реактора [1], к которой предъявляются более жесткие требования [2, 3]), вопросы методики тестирования библиотечных компонентов рас-смотрены недостаточно. Синтез подобных систем, основанных на ПЛИС-технологии, осуществляется с помощью САПР. В частности для разработки системы аварийной защиты используется САПР фирмы Altera Quartus II. При этом используются IP-ядра двух типов: библиотечные компоненты Quartus II (мегафункции) и компоненты шины Avalon (компоненты SoPC Builder, Nios) [4].

Целью статьи является разработка элементов методики и анализ результатов тестирования компонентов ПЛИС-проектов, к которым предъявляются особые требования при разработке критических приложений.

Элементы методики тестирования

Возможны два варианта тестирования IP-ядер: функциональное тестирование методом «черного ящика» и тестирование структуры методом «белого ящика» [5]. Для тестирования методом «белого ящика» необходим доступ к исходному коду библиотечных компонентов САПР, который является коммерческой тайной. Поэтому тестирование проводилось методом «черного ящика», чтобы удостовериться в функциональном соответствии компонентов и известных алгоритмов их работы.

Одним из элементов методики тестирования является тестирование компонентов с использованием симуляции в САПР (Wave Form Editor). С помощью этого редактора было проведено тестирование таких мегафункции Quartus II, как: LPM_OR, LPM_AND, LPM_XOR, ALTPLL, LPM_DECODE, LPM_CONSTANT, LPM_MUX, LPM_DIVIDE, PARALLEL_ADD, LPM_ABS, LPM_COMPARE, LPM_COUNTER, LPM_MULT, LPM_ADD_SUB.

Для тестирования компонентов шины Avalon, в состав которых входят память программ и ОЗУ, интерфейсы UART и SPI, таймеры, DMA, ядро Nios находилось в отладочном режиме, а для отобра-жения

© С.Б. Остроумов, Ю.Н. Прохорова, А.А. Андрашов, А.Д. Герасименко

тестевой информации использовался циф-ровой осциллограф LeCroy WaveSurfer WS343.

Тестирование компонентов Quartus II

Опишем процесс тестирования нескольких мегафункций в Wave Form Editor.

Зачастую в проекте используется несколько тактирующих частот. Для достижения этого используется **функциональный блок ALTPLL,** который представляет собой фазовый умножитель частоты (рис. 1)**.** Входным сигналом (INPUT) этого блока является тактирующий вход in_clk в 27 МГц. На выходе блока ALTPLL формируются три необходимые заданные тактовые частоты (OUTPUT): out_clk0 – 48 МГц; out_clk1 – 32 МГц; out_clk2 – 27 МГц.

В таблице 1 приведены тестовые проверки функционального блока ALTPLL.

Таблица 1

Тестирование функционального блока ALTPLL

Входной сигнал in_clk	Выходные сигналы		
	out_clk0	out_clk1	out_clk2
27 МГц	48 МГц	32 МГц	27 МГц

Эпюры тестирования функционального блока ALTPLL изображены на рис. 2.

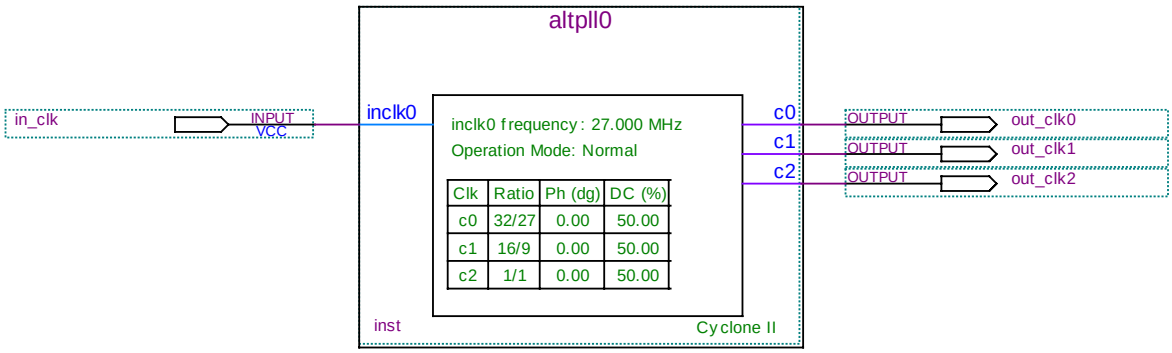

Рис. 1. Функциональный блок ALTPLL

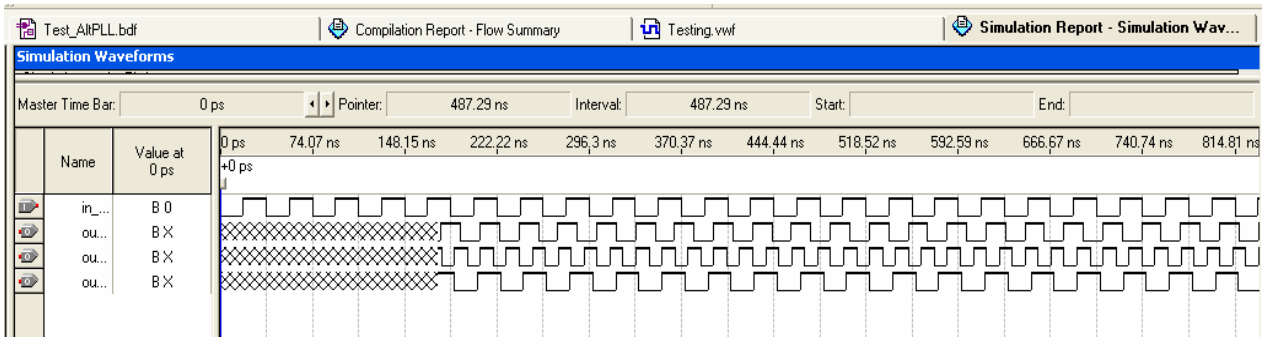

Рис. 2. Эпюры тестирования функционального блока ALTPLL

Тестирование функциональных блоков ADD_SUB, LPM_MULT и LPM_DIV проводилось с помощью процессорного ядра Nios II (LPM_TEST, рис. 3), которое использовалось для выдачи чисел и вывода результата на экран компьютера. Каждый из блоков имеет входы dataa и datab, на которые подавались десять чисел подряд с помощью программы, приведенной на рис. 3. На выходе result тестируемой мегафункции формировалось значение, которое поступало на вход Nios II. В качестве примера, рассмотрим тестирование блока LPM_MULT по указанной методике.

Мегафункция **LPM_MULT** представляет собой блок умножения двух чисел dataa и datab (рис. 4). Результат операции умножения формируется на выходном сигнале result. В таблице 2 приведены

тестовые проверки функционального блока LPM_MULT и результаты выполнения программы.

```
#include <system.h>
#include <stdio.h>
#include <altera_avalon_pio_regs.h>

int main(){
  int i,j;

  for(i=0; i<10; i++){
   IOWR_ALTERA_AVALON_PIO_DATA(NUMBER_1_BASE, i);
   for (j=0; j<10; j++)
   {
     IOWR_ALTERA_AVALON_PIO_DATA(NUMBER_2_BASE, j);
     printf("%d ",IORD_ALTERA_AVALON_PIO_DATA(RESULT_BASE));
   }
   printf("\n");
  }
  return 0;
}
```

Рис. 3. Программа, выполняющая тестирование путем выдачи десяти чисел подряд и выводящая результат функциональных блоков

Аналогичным образом было проведено тестирование мегафункций ADD_SUB и LPM_DIV с учетом особенностей их функционирования.

Таблица 2

Тестирование функционального блока LPM_MULT

	Входные сигналы dataa, datab	**Выходной сигнал result**
0	0 1 2 3 4 5 6 7 8 9	0 0 0 0 0 0 0 0 0 0
1	0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9
2	0 1 2 3 4 5 6 7 8 9	0 2 4 6 8 10 12 14 16 18
3	0 1 2 3 4 5 6 7 8 9	0 3 6 9 12 15 18 21 24 27
4	0 1 2 3 4 5 6 7 8 9	0 4 8 12 16 20 24 28 32 36
5	0 1 2 3 4 5 6 7 8 9	0 5 10 15 20 25 30 35 40 45
6	0 1 2 3 4 5 6 7 8 9	0 6 12 18 24 30 36 42 48 54
7	0 1 2 3 4 5 6 7 8 9	0 7 14 21 28 35 42 49 56 63
8	0 1 2 3 4 5 6 7 8 9	0 8 16 24 32 40 48 56 64 72
9	0 1 2 3 4 5 6 7 8 9	0 9 18 27 36 45 54 63 72 81

Исходя из анализа данных таблицы, можно сделать вывод, что мегафункция умножения LPM_MULT формирует правильные результаты.

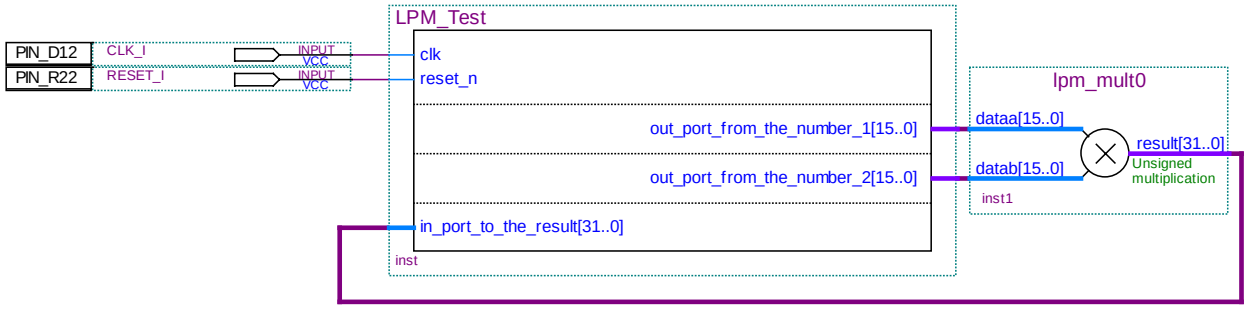

Рис. 4. Функциональный блок LPM_MULT

Тестирование компонентов Nios

Для реализации вычислений в ПТК используется процессорное IP-ядро (soft-processor) Nios с шиной Avalon [4]. При этом к этой шине подключены различные интерфейсы и дополнительные IP-ядра, которые также имеет смысл протестировать. Важным блоком - IP-ядром является таймер, задающий рабочий цикл программы. Для тестирования таймера сигнал переполнения, был выведен на определенный вывод микросхемы, к которому подключался осциллограф LeCroy WaveSurfer (рис. 5). В данном случае интервальный таймер является неуправляемым и генерирует сигнал по переполнению один раз в 10 мс (X2 = 10.00448 ms).

Аналогичным образом были протестированы интерфейсы UART и SPI.

Тестирование ОЗУ микросхемы проводилось в два этапа. Первый – побитная запись «0» и «1». Второй – запись в адреса их номеров.

Что касается вычислительных возможностей ядра Nios, тестирование осуществлялось путем выполнения операций сложения и умножения двух чисел. Так как операции вычитания и деления выполняются аналогично и результат операций умножения и сложения верный, дополнительного тестирования первых не проводилось.

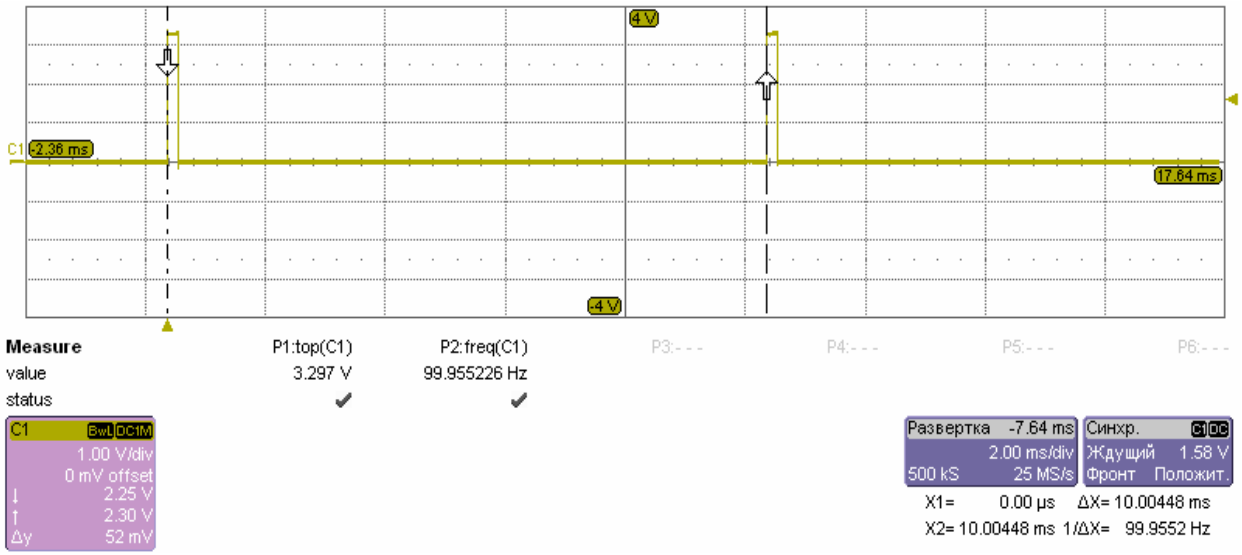

Рис. 5. Осциллограмма тестирования интервального таймера

Тестирование ПО Nios методом статического анализа

Функционирование IP-ядро Nios невозможно без наличия программного кода, реализующего логику технологического процесса того или иного объекта контроля и управления. Этот факт делает тестирование, исключительно аппаратных элементов системы, недостаточным для объективной оценки качества и надежности ПТК критического применения.

Наряду с тестированием аппаратных компонент, проводился статический анализ программного кода, внедряемого в Nios. В качестве примера был выбран один из блоков ПТК аварийной и предупредительной защиты реактора, разработки НПП «Радий».

При проведении экспертизы использовались инструментальные средства для статического анализа программного кода, а именно, инструментальное средство «Together» [6].

Программное обеспечение микросхем ПЛИС реализовано на языке С в среде разработки GNUPro Toolkit.

Статический анализ программного кода выполнялся по следующим направлениям:

– проверка программного кода на соответствие правилам программирования;

– расчет и анализ значений метрик.

Для проверки программного кода на соответствие правилам программирования были заданы 10 правил программирования, отклонение от которых считаются недопустимыми, поскольку это создает потенциальную проблему в ходе функционирования программного кода.

В ходе метрического анализа ПО было выполнено:

– автоматизированное определение атрибутов программного кода, необходимых для расчета метрик сложности;

– автоматизированный расчет метрик сложности;

В ходе проведения экспертизы были получены следующие результаты:

– несоответствий правилам кодирования выявлено не было.

Результаты метрического анализа ПО представлены в табл.3.

Анализ таблицы 3 показал, что значения всех метрик для ПО находятся в пределах допустимых диапазонов.

Таблица 3

Тестовые проверки функционального блока LPM_MULT

Наименование метрики	Краткое описание	Диапазон допустимых значений	Значение метрики
Количество строк кода	–	[0 – 2500]	243
Количество атрибутов	–	[0-30]	26
Количество операций		[0-50]	35
Количество классов	–	[0-5]	4
Количество конструкторов	–	[0-5]	4
Количество членов класса	Определяется как сумма количества атрибутов и количества операций	[0-80]	38
Количество выходов	Учитываются связанные типы данных, такие как атрибуты, возвращаемые формальные параметры, типы, метки и локальные переменные. Простые типы и супертипы при этом не учитываются	[0-15]	12
Цикломатическая сложность	CC = (число связей в графе потока управления) – (число узлов в графе потока управления) + 2 * (число разъединенных частей в графе потока управления)	[0 – 50]	21

Выводы

В работе предложены элементы методики тестирования библиотечных и аппаратных компонентов, которые используются при разработке критических ПТК, а также элементы статического анализа программного кода ядра Nios, написанного на языке С.

При тестировании различного рода элементов возникает вопрос полноты тестирования. Исходя из того, что алгоритм работы мегафункций не зависит от размерности входов и выходов, их тестирование при минимальной размерности является достаточным. Проанализировав полученные данные можно утверждать, что мегафункции САПР и IP-ядра Nios соответствуют известным алгоритмам их работы. Метрический анализ программного кода показал, что значения всех метрик ПО находятся в пределах допустимых диапазонов. Это позволяет говорить о возможности использования данных компонентов и программного кода при разработке критических приложений, таких как ПТК ИУС АЭС.

Направлением дальнейших исследований является тестирование комплекса компонентов, объединенных в инфраструктуру IP-ядер.

Литература

1. Бахмач Е.С., Виноградская С.В., Розен Ю.В., Сиора А.А., Токарев В.И., Юрцевич А.М. Программно-технические комплексы аварийной и преду-предительной защиты атомных реакторов: обеспечение и оценка безопасности // Ядерная и радиационная безопасность. – вып.1, том 8. – 2005 г. – С. – 21-51.

2. НП 306.5.02/3.035-2000. Требования по ядерной и радиационной безопасности к информационным и управляющим системам, важ-ным для безопасности.

3. IEC 61513 Ed.1: 2001. Nuclear power plants – Instrumentation and Control for systems important to safety – General requirements for systems.

4. FPGA, CPLID and Structured ASIC: Altera, the Leader in Programmable Logic [Электр. Ресурс] – Режим доступа: http://www.altera.com/

5. Гленфорд Майерс. Искусство тестирования программ. – М., 1982 г. – С. – 20-34.

6. User Guide for Together ControlCenter™. Borland Software Corporation 2004. – 922 p.

Поступила в редакцию 22.01.2008

Рецензент: д-р техн. наук, проф. В.С. Харченко, Национальный аэрокосмический университет им. Н.Е. Жуковського «ХАИ», Харьков.