УДК 004.415.2
DOI 10.21685/2072-3059-2016-2-3

Р. Н. Федюнин

ВРЕМЕННОЙ АНАЛИЗ И РЕАЛИЗАЦИЯ АППАРАТНО-ПРОГРАММНЫХ МОДУЛЕЙ АРИФМЕТИЧЕСКОГО ЛОГИЧЕСКОГО УСТРОЙСТВА

Аннотация.

Актуальность и цели. Объектом исследования являются аппаратно-программные модели вычислительных устройств на базе ПЛИС. Предмет исследования – методика временного анализа аппаратно-программных модулей вычислительных устройств и их синтез на базе ПЛИС. Цель– разработка способа быстрой оценки временных затрат модулей вычислительной системы, реализуемой на базе ПЛИС.

Материалы и методы. Представленная в статье методика объединяет математический и имитационный способы временной оценки алгоритмов, реализуемых в рамках аппаратно-программных модулей. Имитационный способ дает наглядную картину работы модуля и позволяет собрать библиотеку модулей для последующего комплексного анализа вычислительного устройства. Математический способ удобен тем, что не требует знания специализированного программного обеспечения и позволяет быстро сделать приблизительные расчеты отдельного модуля устройства.

Результаты. Разработаны две компьютерные модели пяти вычислительных устройств. Первая компьютерная модель построена на базе САПР CPNTools и математического аппарата сетей Петри. Вторая компьютерная модель – с использованием языка VHDL и САПР ALTERA. Оба подхода показали адекватность разработанной методики оценки временных затрат.

Выводы. Несмотря на наличие в средствах реализации СБИС систем программных модулей для расчета и визуализации временных затрат, описанная в статье методика на ранних этапах формирования архитектуры позволяет быстро оценить адекватность затрат на разработку вычислительного устройства и внести нужные разработчикам коррективы.

Ключевые слова: временной анализ, вычислительное устройство, вычислительная система, арифметико-логическое устройство, методика анализа, аппаратно-программный модуль, функциональный блок.

R. N. Fedyunin

TIME ANALYSIS AND IMPLEMENTATION OF HARDWARE AND SOFTWARE MODULES OF ALU

Abstract.

Background. The research object is hardware and software models of FPGA-based computing devices. The research subject is the methodology of time analysis of hardware-software modules of computing devices and synthesis thereof on the FPGA basis. The aim of the work is to develop a method of rapid time study of FPGA-based computing system modules.

Materials and methods. The methodology represented in the article unites mathematical and simulation methods of time estimates of algorithms, realized within hardware-software modules. The simulation method shows pictorial functioning of the module and allows to collect a library of modules for further complex analysis

of the computing device. The mathematical method is convenient as it requires no knowledge of specialized software and allows to roughly calculate an individual device module.

Results. The author developed two computer models of five computing devices. The first computer model is built on the basis of the CPNTools CAD and the Petri nets mathematical apparatus. The second model – using the VHDL language and the ALTERA CAD. Both approaches proved adequacy of the developed methodology of time estimates.

Conclusions. Despite a presence of systems of program modules for time calculation and visualization in VLSI realization means, the methodology described in the article allows to rapidly estimate computing device development time adequacy and to introduce required amendments at early stages of architecture formation.

Key words: timing analysis, computing device, computing system, arithmetic logic unit, methods of analysis, hardware and software module, functional block

Для иллюстрации рассматриваемой методики временного анализа в качестве примера возьмем схему RISC-ядра, разрабатываемого для эксплуатации в виде аппаратно-программных модулей вычислительного устройства (рис. 1) [1]. Каждый модуль RISC-ядра (рис. 1), в том числе и модуль арифметического логического устройства (АЛУ) [2], содержит блоки, выполняющие базовые операции, которые далее именуются функциональными [3, 4] (рис. 2).

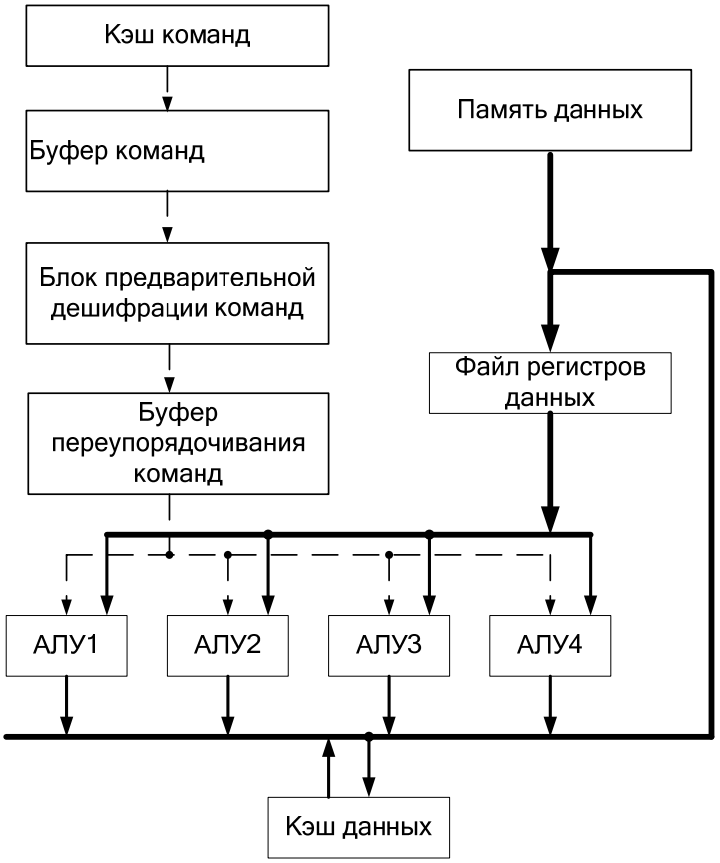

Рис. 1. Обобщенная структура операционной части RISC-ядра

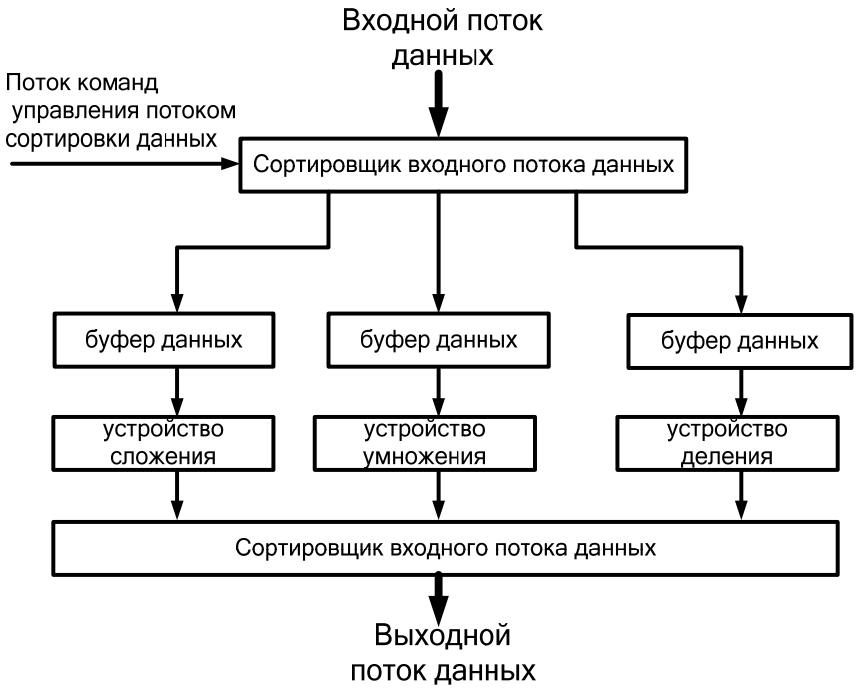

Рис. 2. Вариант реализации АЛУ RISC-ядра

Предложенный после анализа вариант вычислительного устройства будет реализован на базе ПЛИС, что позволит быстро и дешево ввести разработанное устройство в эксплуатацию.

В первую очередь перед исследователем встает задача разработки общей схемы вычислительного устройства (рис. 1), затем разработчик описывает схемы реализации аппаратно-программных модулей устройства (рис. 2) и на третьем шаге итеративно выбирает, анализирует и модернизирует алгоритмы реализации функциональных блоков (ФБ) модулей (рис. 3).

Выбранный алгоритм ФБ должен обладать простотой реализации и минимальными временными задержками относительно эталонного алгоритма. Эталоном разработчик обычно берет алгоритм аналогичных функциональных блоков, разработанных ранее. Затем разработчик итеративно действует по шагам методики (рис. 3): делает теоретическую оценку производительности предлагаемых алгоритмов арифметических и логических операций; производит имитационный анализ исследуемых алгоритмов с помощью специализированного программного обеспечения; выбирает алгоритм с наилучшими временными показателями для реализации функционального блока на базе ПЛИС. Алгоритм, показавший наилучшее время работы, реализуется на ПЛИС с помощью языка VHDL.

Для иллюстрации математического способа временного анализа (шаги 4–8 на рис. 3) рассмотрим процесс анализа, выбора и реализации алгоритма деления в рамках функционального блока деления АЛУ (рис. 2). В нашем случае выбор осуществляется среди алгоритмов целочисленного деления: деление с восстановлением остатка [5, 6], деление без восстановления остатка [5, 6], SRT-деление [7], способ деления Ньютона – Рафсона [7], алгоритм де-

ления Гольдшмита [7]. В качестве математического аппарата для определения временных затрат функциональных блоков выберем аппарат теории марковских процессов, подробно описанный [8] и эффективный на уровне аппаратно-программных модулей.

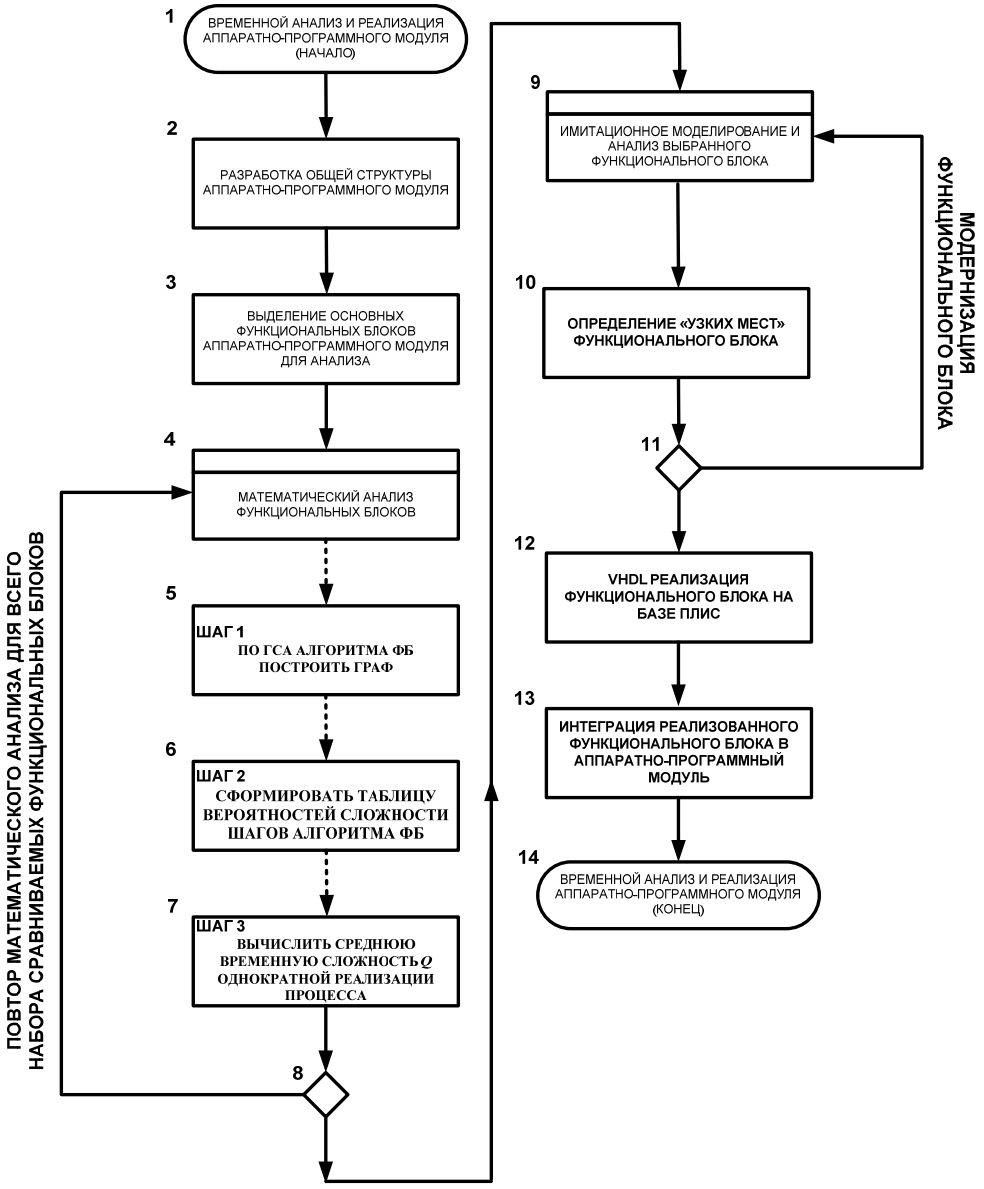

Рис. 3. Иллюстрация методики временного анализа и реализации аппаратно-программного модуля вычислительной системы

Процесс временного анализа (шаги 4–14 на рис. 3) показан для одного алгоритма – SRT-деление. Остальные способы деления анализируются аналогично.

На первом шаге теоретической оценки по граф-схеме SRT-алгоритма (ГСА) деления строится марковский граф (рис. 4) [8].

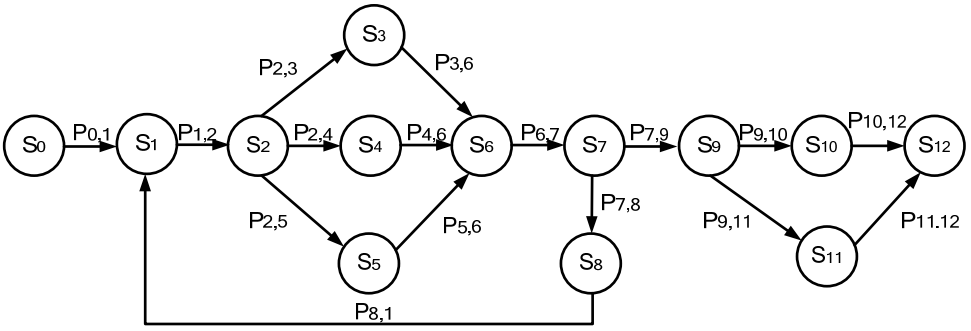

Рис. 4. Граф алгоритма SRT-деления

В данном графе состояния S_i определяют соответствующие состояния исследуемого алгоритма, а P_{ij} – вероятности переходов из одного состояния алгоритма S_i в другое S_j.

На втором шаге теоретической оценки формируется таблица вероятностной сложности шагов алгоритма деления (табл. 1).

Таблица 1

Вероятностная сложность шагов алгоритма SRT-деления

Состояние S_i	Значение состояния S_i	Вероятность P_{ij} перехода из состояния S_i в состояние S_j	Примечание
S_0	состояние, в котором находится вычислитель до запуска вычислительного процесса	$P_{0,1} = 1$	нет
S_1	ввод делимого X	$P_{1,2} = 1$	нет
S_2	ввод делителя Y	$P_{2,3} = 0,33$	нет
S_3	получить остаток $R = X - Y$	$P_{2,4} = 0,33$	нет
S_4	анализ: остаток больше нуля или нет	$P_{2,5} = 0,33$	нет
S_5	флаг Z установить в 1	$P_{3,6} = 1$	нет
S_6	сдвиг R вправо на один разряд	$P_{3,6} = 1$	нет
S_7	анализ: остаток больше нуля или нет	$P_{4,6} = 1$	нет
S_8	сложить остаток R с делителем Y: $R = R + Y$	$P_{5,6} = 1$	нет
S_9	декремент счетчика выполнения операции деления	$P_{6,7} = 1$	нет
S_{10}	анализ	$P_{7,8} = N - 1$	нет
S_{11}	конец операции деления	$P_{7,9} = 1$	нет
S_{12}	финальное состояние	$P_{9,10} = 0,5$	нет
		$P_{9,11} = 0,5$	нет

При однократной и результативной реализации процесса вычислений временная сложность алгоритма SRT-деления определяется по методике оценки сложности процессов с ветвлениями [8], для чего решается система уравнений [8] при $d_0 = 1$ и вероятностях P_{ij} (табл. 2).

Таблица 2

Временная сложность алгоритма SRT-деления

Система уравнений определения временной сложности алгоритма SRT-деления Q	Временная сложность алгоритма SRT-деления				
	при $N = 8$	при $N = 16$	при $N = 32$	при $N = 64$	при $N = 128$
$-1 \cdot d_0 = -1$	1	1	1	1	1
$P_{0,1} \cdot d_0 + P_{8,1} \cdot d_8 - d_1 = 0$	7,5	14	24	39	56,4
$P_{1,2} \cdot d_1 - d_2 = 0$	7,5	14	24	39	56,4
$P_{2,3} \cdot d_2 - d_3 = 0$	2,5	5	8	13	19
$P_{2,4} \cdot d_2 - d_4 = 0$	2,5	5	8	13	19
$P_{2,5} \cdot d_2 - d_5 = 0$	2,5	5	8	13	19
$P_{3,6} \cdot d_3 + P_{4,6} \cdot d_4 + P_{5,6} \cdot d_5 - d_6 = 0$	7,4	14	24	39	56
$P_{6,7} \cdot d_6 - d_7 = 0$	0,9	0,9	0,7	0,5	0,5
$P_{6,8} \cdot d_6 - d_8 = 0$	6,5	13	23	38	55
$P_{7,9} \cdot d_7 - d_9 = 0$	0,5	0,4	0,4	0,3	0,3
$P_{7,10} \cdot d_7 - d_{10} = 0$	0,5	0,4	0,4	0,3	0,3
$P_{9,11} \cdot d_9 + P_{10,11} \cdot d_{10} - d_6 = 0$	0,9	0,9	7,5	0,5	0,5
Временная сложность алгоритма SRT-деления $Q = \sum_{k=0}^{k-1} d_k$, в тактах машинного времени	**40**	**72**	**124**	**197**	**282**

На третьем шаге вычисляется средняя временная сложность Q однократной и результативной реализации процесса, которая определяется как сумма d_i: $Q = \sum_{k=0}^{k-1} d_k$ (табл. 2), а средняя временная сложность вычислений массива данных Q_{mas} определяется произведением средней временной сложности Q однократной и результативной реализации процесса на количество данных в массиве $Q_{mas} = N \cdot \sum_{k=0}^{k-1} d_k$, где N – количество данных в массиве.

Повторив описанные выше расчеты для исследуемых алгоритмов, разработчик выбирает лучший с точки зрения временных затрат (рис. 5 – иллюстрация массовой обработки нескольких десятков целых чисел). В данном случае меньшее время обработки показал алгоритм SRT-деления.

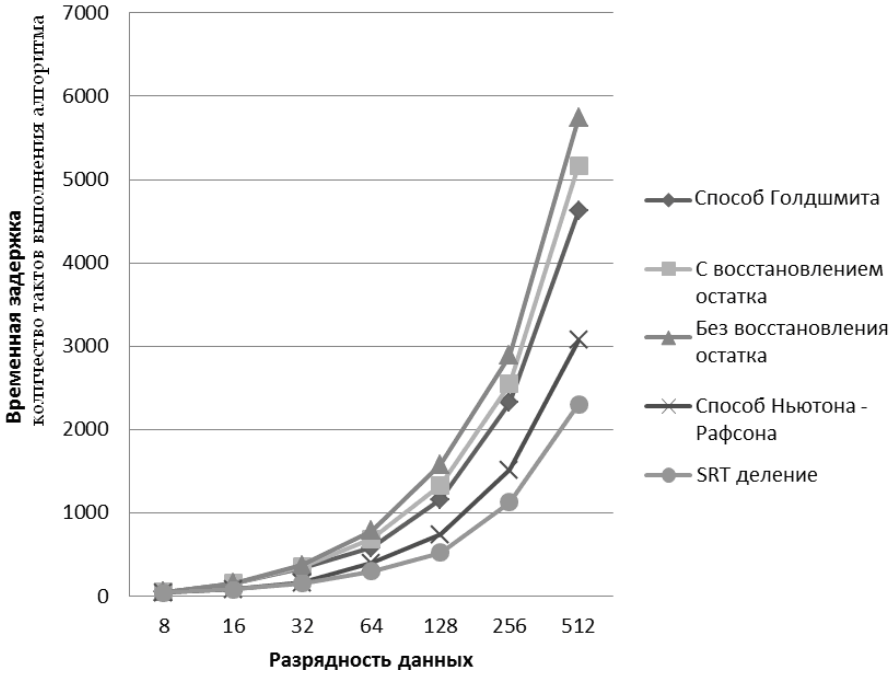

Рис. 5. Временные диаграммы временного анализа алгоритмов деления математическим способом

Для проверки адекватности математической модели реализуется имитационная модель алгоритмов деления в системе CPN Tools с использованием сетей Петри [9] (рис. 6) (шаги 9–10 на рис. 3). В отличие от иных систем и языка имитационного моделирования, данный подход позволяет описывать как дискретные, так и вероятностные модели, как синхронные, так и асинхронные вычислительные модули [9]. Вид имитационной модели алгоритма SRT-деления практически полностью повторяет граф на рис. 4. При реализации логики имитационной модели помимо графических элементов используется скриптовый язык описания логики. Так, в данной имитационной модели (рис. 6) элементы sub_signed, add_signed, setbit, lsr – скрипты, логика которых повторяет логику шагов алгоритма SRT-деления [7].

Приведем пример кода **add_signed** [7]:

```
     fun add_signedN (lhs : signedN, rhs : signedN) =
sl2signedN (add_sl([#1 lhs, #2 lhs, #3 lhs, #4 lhs, #5 lhs,
#6 lhs, #7 lhs, #8 lhs…#N lhs],
     [#1 rhs, #2 rhs, #3 rhs, #4 rhs, #5 rhs, #6 rhs, #7 rhs,
#8 rhs…#N rhs]));
     скрипт – сложения N разрядных чисел.
     Fun sub_signedN (lhs : signedN, rhs : signedN) = let
     val rhs_ok = (not_bit(#1 rhs), not_bit(#2 rhs),
not_bit(#3 rhs), not_bit(#4 rhs), not_bit(#5 rhs), not_bit(#6
rhs), not_bit(#7 rhs), not_bit(#8 rhs)… not_bit(#N rhs));
     in add_signedN(lhs, add_signedN(rhs_ok, (0, 0, 0, 0,… 0,
0, 0, 1)))
     end
```

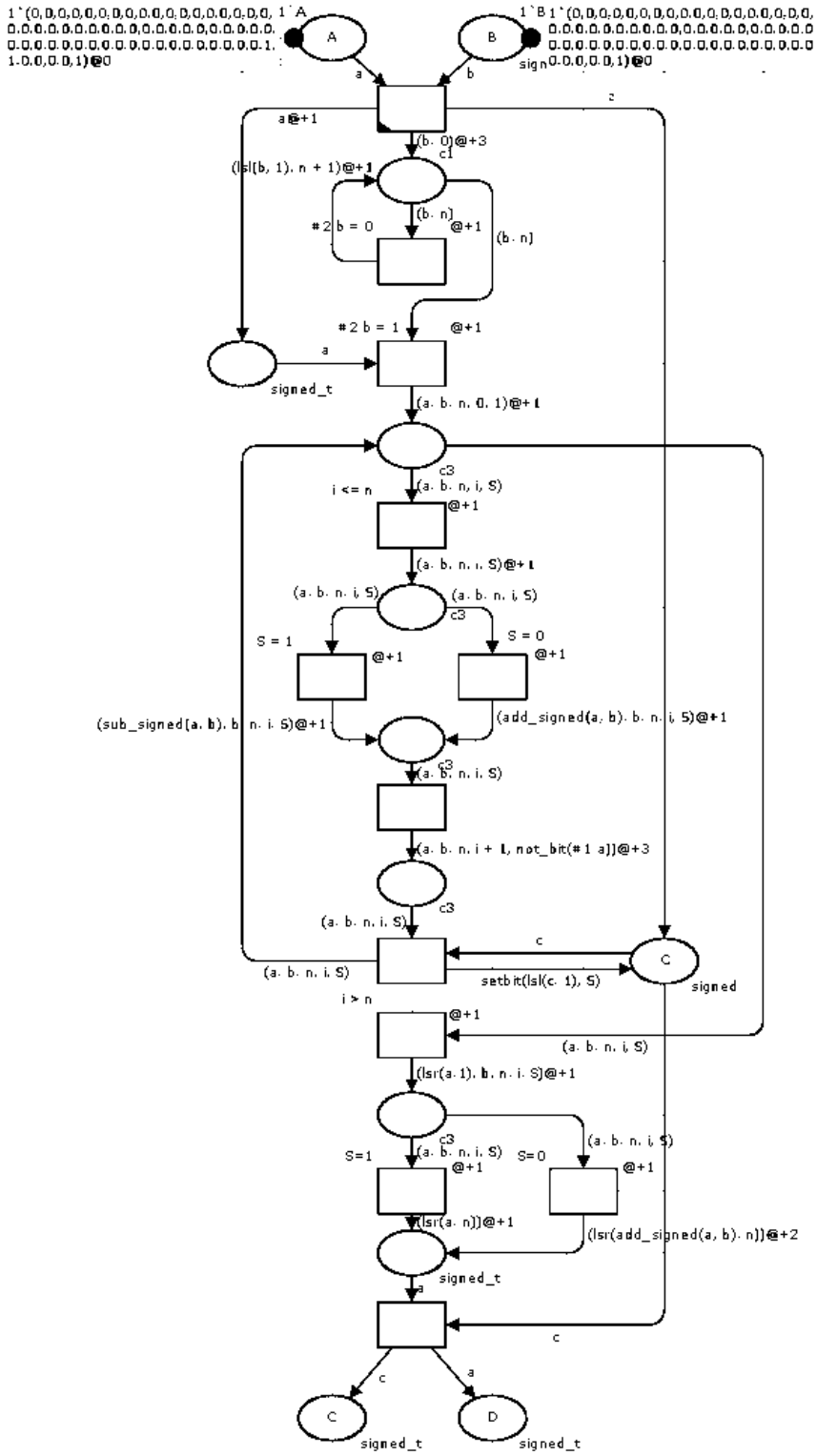

Рис. 6. Имитационная модель функционального блока, реализующего алгоритм SRT-деления в CPN Tools

Данная модель в интерактивном режиме показывает потоки информации по шагам в рамках функционального блока. Система статистики дает возможность увидеть как общую временную задержку, так и задержку по шагам [9]. В результате разработчик, если этого требует необходимость, может провести анализ и доработку модели, а затем испытать имитационную модель функционального блока (рис. 7) и всего модуля целиком.

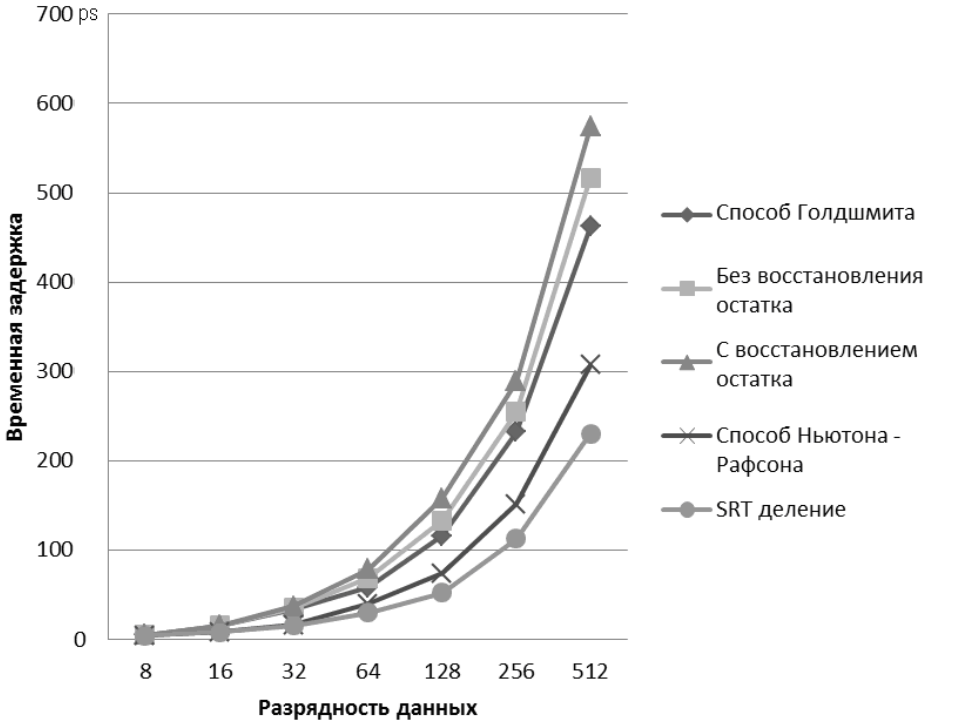

Рис. 7. Результаты имитационного моделирования алгоритмов деления

После теоретической и имитационной проверки испытанный алгоритм реализуется в виде самостоятельного устройства на базе ПЛИС фирмы ALTERA в виде иерархической связки условного графического обозначения (УГО) и логики на языке VHDL (рис. 8) [10].

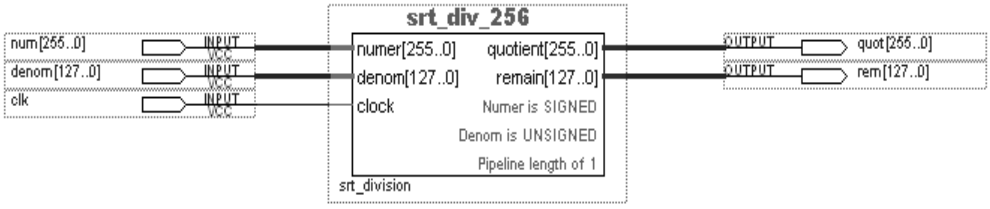

Рис. 8. Условное графическое обозначение устройства
блока SRT-деления 256-разрядных данных

Код устройства SRT-деления на языке VHDL (реализация логики алгоритма SRT-деления) (рис. 8):

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.all;
ENTITY square_root IS
  GENERIC(n: NATURAL := 256);
PORT (
  x: IN STD_LOGIC_VECTOR(n-1 DOWNTO 0);
  clk, reset, start: IN STD_LOGIC;
  r: INOUT STD_LOGIC_VECTOR(n-1 DOWNTO 0);
  done: OUT STD_LOGIC
);
END square_root;
ARCHITECTURE behavior OF square_root IS
  SIGNAL next_r, s, next_s: STD_LOGIC_VECTOR(n-1 DOWNTO 0);
  SIGNAL greater: STD_LOGIC;
  SIGNAL ce, load: STD_LOGIC;
  TYPE states IS RANGE 0 TO 2;
  SIGNAL current_state: states;
BEGIN
  next_r <= r + 1;

  next_s <= s + (next_r(n-2 DOWNTO 0)&'0') + 1;

  register_r: PROCESS(clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF LOAD = '1' THEN r <= (OTHERS => '0');
      ELSIF ce = '1' THEN r <= next_r;
      END IF;
    END IF;
  END PROCESS;
  register_s: PROCESS(clk)
  BEGIN
    IF clk'EVENT AND clk = '1' THEN
      IF LOAD = '1' THEN s <= CONV_STD_LOGIC_VECTOR(1,256);
      ELSIF ce = '1' THEN s <= next_s;
      END IF;
    END IF;
  END PROCESS;
  greater <= '1' WHEN s > x ELSE '0';
  control_unit_output: PROCESS(current_state, start, greater)
  BEGIN
    CASE current_state IS
      WHEN 0 => ce <= '0'; load <= '0'; done <= '1';
      WHEN 1 => ce <= '0';
                IF start = '1' THEN load <= '1'; done <= '0';
ELSE load <= '0'; done <= '1'; END IF;
      WHEN 2 => IF greater = '0' THEN ce <= '1'; ELSE ce <=
'0'; END IF;
                load <= '0'; done <= '0';
    END CASE;
  END PROCESS;
  control_unit_next_state: PROCESS(clk, reset)
  BEGIN
    IF reset = '1' THEN current_state <= 0;
    ELSIF clk'event AND clk= '1' THEN
```

```
      CASE current_state IS
       WHEN 0 => IF start = '0' THEN current_state <= 1; END
IF;
       WHEN 1 => IF start = '1' THEN current_state <= 2; END
IF;
       WHEN 2 => IF greater = '1' THEN current_state <= 0;
END IF;
      END CASE;
    END IF;
  END PROCESS;
END behavior;
```

Затем производится апробация готового к эксплуатации функционального блока средствами САПР Altera [10]. Если устройство на испытаниях показывает верные тестовые результаты (рис. 9), то оно сохраняется в рабочей библиотеке САПР Altera для последующего использования в рамках аппаратно-программного модуля. Иначе производится тестирование разработанного функционального блока [10].

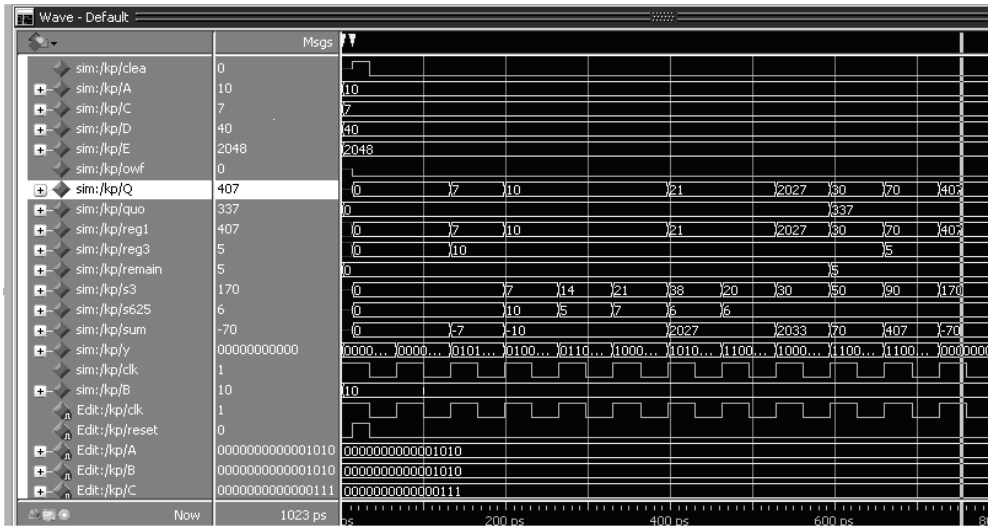

Рис. 9. Апробация работа функционального блока
SRT-деления средствами САПР Altera

После формирования библиотеки готовых функциональных блоков аппаратно-программного модуля, в данном случае модуль арифметико-логического устройства (рис. 10), осуществляется формирование и проверка работоспособности всей схемы АЛУ [10] (рис. 11, 12). Функциональные блоки АЛУ (рис. 10): lpm_add_sub, lpm_divide, inv_to_dop, shift – реализованы с использованием описанной выше методики.

В итоге получена и описана методика оценки временных затрат и реализации аппаратно-программных модулей на базе ПЛИС. Адекватность методики была проверена реализацией вычислительных устройств на базе различных типов ПЛИС с последующим анализом временных затрат в САПР фирмы Altera [10]. Практичность методики доказана неоднократным использованием в рамках грантов и федеральных целевых программ.

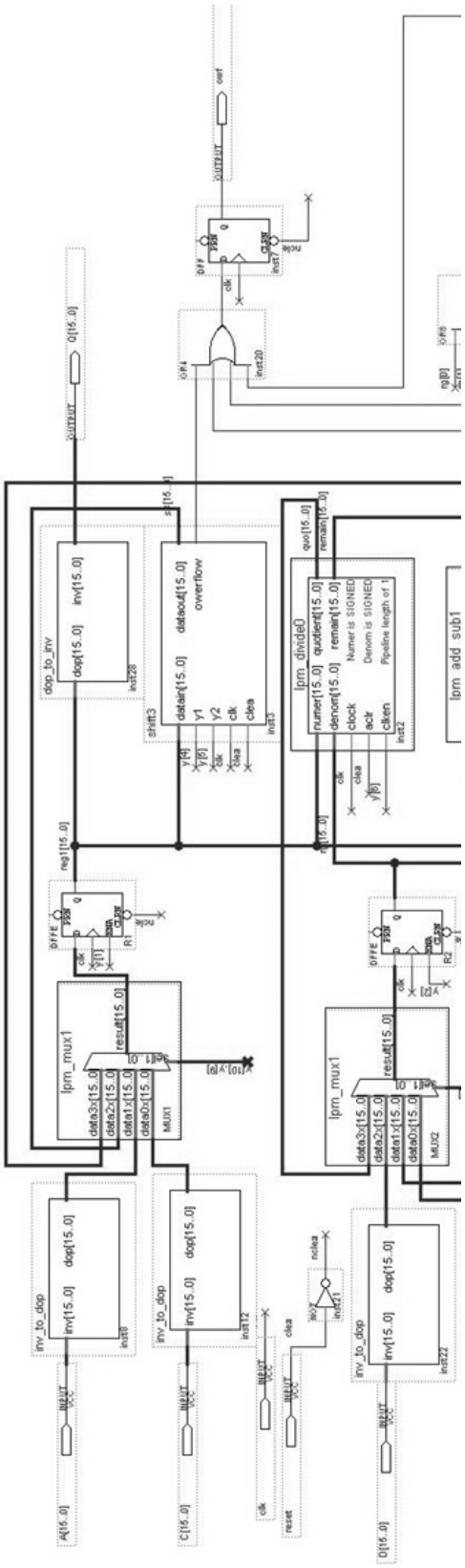

Рис. 10. Схема АЛУ на базе арифметико-логических функциональных блоков

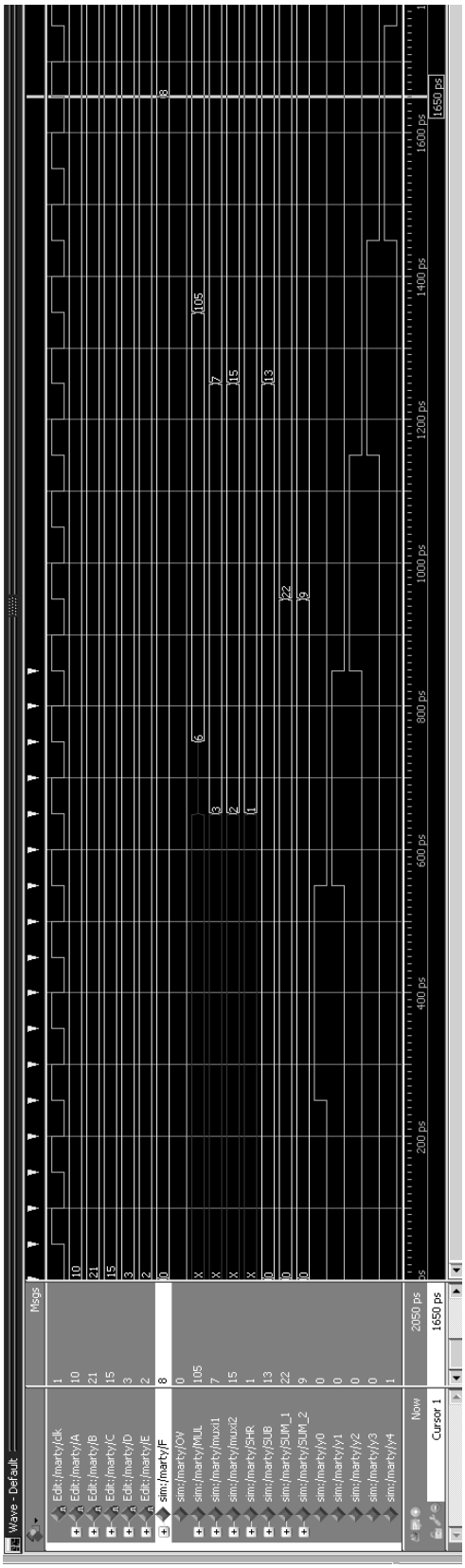

Рис. 11. Пример работы аппаратно-программного модуля АЛУ

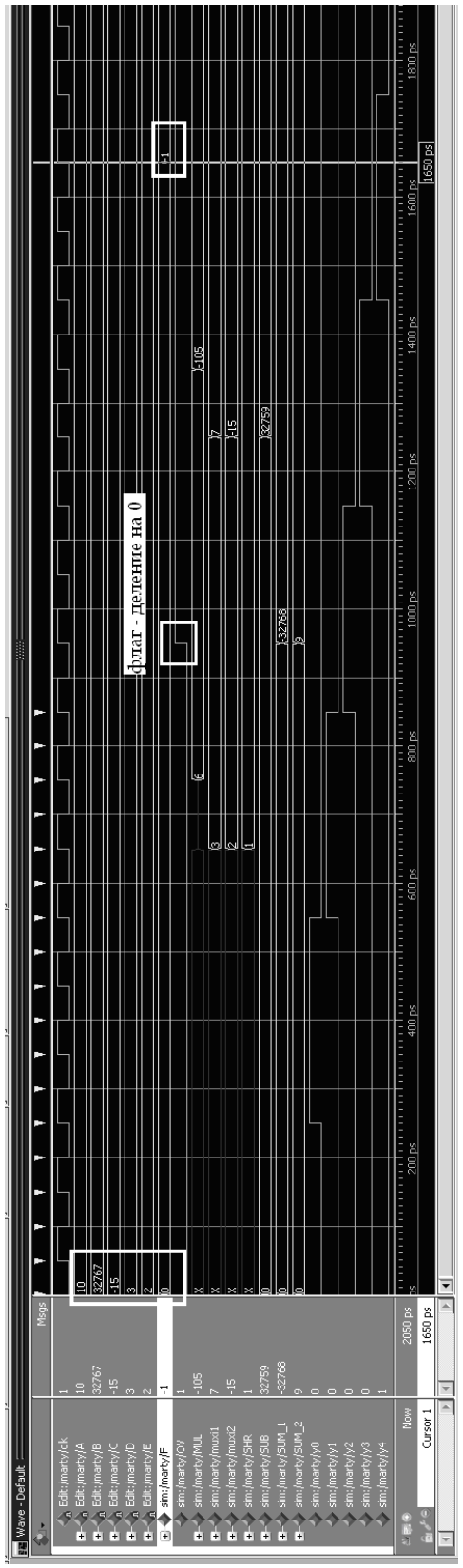

Рис. 12. Пример работы аппаратно-программного модуля АЛУ (исключительная ситуация – деление на ноль)

Несмотря на наличие в средствах реализации СБИС систем программных модулей для расчета и визуализации временных затрат, данная методика на ранних этапах формирования архитектуры позволяет быстро оценить адекватность затрат на разработку вычислительного устройства и внести нужные разработчикам коррективы. Автор неоднократно применял данную методику при обосновании патентных решений на базе вычислительных систем [2, 5].

Список литературы

1. **Федюнин, Р. Н.** Способ реализации аппаратного слоя VLIW архитектуры на базе систолических структур / Р. Н. Федюнин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2013. – № 2 (26). – С. 15–22.
2. Патент № 2284568 Российская Федерация. Ячейка однородной вычислительной среды / Федюнин Р. Н., Князьков В. С. – Опубл. 27.09.2006, Бюл. № 27.
3. Maltsev, A. Triangular systolic array with reduced latency for QR-decomposition of complex matrices, Circuits and Systems, 2006. ISCAS 2006 / A. Maltsev, V. Pestretsov, R. Maslennikov, A. Khoryaev // Proceedings. 2006 IEEE International Symposium, 2006. – P. 602–608.
4. **Hegde, G.** Implementation of Systolic Array Architecture for Full Search Block Matching Algorithm on FPGA / Ganapathi Hegde, Prasanna Cyril, P. Raj and P. R. Vaya // European Journal of Scientific Research. – 2009. – Vol. 33, № 4. – P. 606–616.
5. **Федюнин, Р. Н.** Конвейерно-модульные умножители в системе остаточных классов / Р. Н. Федюнин, В. С. Князьков // Вопросы радиоэлектроники. – 2007. – Т. 1, № 1. – С. 1–14.
6. **Федюнин, Р. Н.** Функциональные блоки АЛУ для конвейерно-параллельной обработки информации на базе однородных вычислительных структур / Р. Н. Федюнин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 2. – С. 32–42.
7. **Костин, А. Е.** Организация и обработка структур данных в вычислительных системах / А. Е. Костин, В. Ф. Шангин. – М. : Высшая школа, 1987. – 245 с.
8. **Федюнин, Р. Н.** Оценка пространственно-временной сложности и способы повышения скорости двоичных арифметических операций / Р. Н. Федюнин // Научное обозрение. – 2006. – № 3. – С. 100–111.
9. **Дворянский, Л. В.** Имитационное моделирование и верификация вложенных сетей Петри с использованием CPN Tools / Л. В. Дворянский, И. А. Ломазова // Моделирование и анализ информационных систем. – 2012. – Т. 19, № 5. – С. 115–130.
10. URL: http://www.altera.com/

References

1. Fedyunin R. N. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2013, no. 2 (26), pp. 15–22.
2. Patent № 2284568 Russian Federation. *Yacheyka odnorodnoy vychislitel'noy sredy* [A cell of a homogeneous computing environment]. Fedyunin R. N., Knyaz'kov V. S. Publ. 27.09.2006, Bull. № 27.
3. Maltsev A., Pestretsov V., Maslennikov R., Khoryaev A. *Proceedings. 2006 IEEE International Symposium*, 2006, pp. 602–608.
4. Hegde G., Prasanna Cyril, Raj P. and Vaya P. R. *European Journal of Scientific Research*. 2009, vol. 33, no. 4, pp. 606–616.

5. Fedyunin R. N., Knyaz'kov V. S. *Voprosy radioelektroniki* [Issues of radioelectronics]. 2007, vol. 1, no. 1, pp. 1–14.
6. Fedyunin R. N. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2007, no. 2, pp. 32–42.
7. Kostin A. E., Shangin V. F. *Organizatsiya i obrabotka struktur dannykh v vychislitel'nykh sistemakh* [Organization and processing of data structures in computing systems]. Moscow: Vysshaya shkola, 1987, 245 p.
8. Fedyunin R. N. *Nauchnoe obozrenie* [Scientific overview]. 2006, no. 3, pp. 100–111.
9. Dvoryanskiy L. V., Lomazova I. A. *Modelirovanie i analiz informatsionnykh system* [Information system modeling and analysis]. 2012, vol. 19, no. 5, pp. 115–130.
10. Available at: http://www.altera.com/

Федюнин Роман Николаевич
кандидат технических наук, доцент, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: frn_penza@mail.ru

Fedyunin Roman Nikolaevich
Candidate of engineering sciences, associate professor, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

УДК 004.415.2

Федюнин, Р. Н.

Временной анализ и реализация аппаратно-программных модулей арифметического логического устройства / Р. Н. Федюнин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 2 (38). – С. 33–48. DOI 10.21685/2072-3059-2016-2-3