УДК 004.274

В.Н. Опанасенко, А.Н. Лисовый

ОСОБЕННОСТИ ЯЗЫКА VHDL ДЛЯ ПРОГРАММИРОВАНИЯ КРИСТАЛЛОВ ПЛИС

Рассмотрены особенности языка описания аппаратных средств для программирования структуры преобразователей чисел с плавающей точкой в числа с фиксированной точкой и обратно. Алгоритмы представлены путем поведенческого описания на языке VHDL. Рассмотрены различные конструкции для организации функции сдвига битов регистра и функции доступа к отдельным битам регистра, которые являются базовыми в преобразователях. Верификация преобразователей реализована методом моделирования в системе ModelSim Xilinx Edition - MXE II.

Введение

Решение широкого круга задач требует использования представления чисел в формате с плавающей точкой (ФПТ) вместо формата с фиксированной точкой (ФФТ). Так как основным форматом представления чисел в современных компьютерах является ФФТ, то полный набор операций для чисел ФПТ содержит, кроме арифметических, операции по преобразованию форматов. Функциональные IP-блоки ФПТ находят широкое применение при построении математических сопроцессоров, DSP-процессоров, встроенных арифметических сопроцессоров. В связи с этим рядом фирм разрабатываются собственные IP-блоки для обработки операндов ФПТ (согласно стандарту IEEE-754 [1]) для кристаллов программируемых логических интегральных схем (ПЛИС) FPGA (Field Programmable Gate Array) [2], которые позволят программировать не только алгоритм обработки данных, но также структуру устройства, реализующего заданный алгоритм. Такие IP-блоки можно легко настраивать в соответствии с требованиями нового проекта, и они, как правило, независимы от технологии изготовления ПЛИС. Наиболее важным свойством готового технического решения является его гарантированное воспроизведение в новом проекте в соответствии со спецификацией, определенной разработчиком этого решения и уточненной разработчиком проекта. Следует отметить, что описание модели с помощью HDL-технологии (Hardware Description Language – язык описания аппаратных средств) позволяет не только сделать ее перенастраиваемой и независимой от технологии, но и выполнять ее моделирование и синтез с использованием инструментальных средств различных фирм [3]. Благодаря сетевым технологиям проектировщик в соответствии со сформулированными техническими требованиями по сети Internet может получить оптимизированное логическое ядро и включить его в свой проект.

Проектирование и реализация цифровых устройств на современной элементной базе может быть выполнено на ПЛИС. Одним из подходов к проектированию устройства является использование HDL-технологии, которая представляет собой комплекс инструментальных средств САПР и методологию проектирования, ориентированных на описание проекта с помощью языка HDL (VHDL или Verilog). Основными предпосылками для внедрения в практику HDL-технологии являются: внедрение промышленных стандартов, обеспечивающих эффективное и оптимальное проектирование средств электроники и вычислительной техники; увеличение логической емкости элементной базы (например, кристаллы Virtex-4 содержат до 10 млн логических вентилей или 125 тыс. логических ячеек); развитие инструментальных средств САПР и в первую очередь средств автоматического синтеза и верификации. Такие возможности HDL-технологии, как иерархическое проектирование, переносимость библиотек, платформонезависимость, позволяют использовать имеющиеся IP-блоки в качестве макроэлементов для разработки новых технических решений в виде IP-блоков (soft cores).

Если до недавнего времени основным способом описания проектов являлся графический синтез схемы устройства, выполняемый схематическим редактором, то в последнее время предпочтение отдается текстовому описанию, подобному тексту программы. Этим достигается компактность и более строгая формализация описания в соответствии со стандартом используемого языка.

Описание в виде текста на HDL-языке регламентируется соответствующими стандартами. Следует отметить, что синтаксически правильное описание на HDL-языке не всегда может быть синтезировано, т.е. представлено в виде аппаратной модели, полученной после компиляции из HDL-кода. Так, например, не могут быть синтезированы сигналы и переменные типа с плавающей точкой, операторы ***Wait*** с параметром задержки и т.д. Перечень несинтезируемых элементов во многом зависит от конкретного средства синтеза и типа выбранного кристалла. Некоторые конструкции могут синтезироваться в различных проектах по разному, к примеру, конструкция ***For*** может синтезироваться как последовательная комбинационная схема (итерации выполняются последовательно с помощью одного синхросигнала) или как схема, каждая итерация цикла будет выполняться по новому синхросигналу.

Постановка задачи

Рассмотрим описание структур, реализующих преобразование чисел одинарной точности ФПТ в формат ФФТ и обратно посредством языка VHDL. Операнд одинарной точности ФПТ, согласно стандарту IEEE-754, представлен 32 битами: 1 бит для знака, 8 бит для порядка и 23 бита для дроби мантиссы. Данный формат предполагает наличие «скрытого» старшего бита, что позволяет представить истинное значение мантиссы на самом деле 24 битами.

Операнд ФПТ представляется выражением

$$A = (-1)^S \times 2^{E-127} \times 1.M ,$$

где S – знак числа; E – показатель порядка (смещенный на значение b=127); $1.M$ – нормализованная мантисса.

Формат операнда и диапазон допустимых значений представлен на рис. 1.

Операнд ФФТ представлен 32 битами (1 бит для знака и 31 бит для модуля целого числа) и определяется выражением

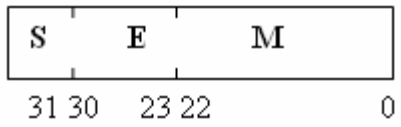
$$A = (-1)^S \times N ,$$

где: S – знак числа; N – значение целого числа.

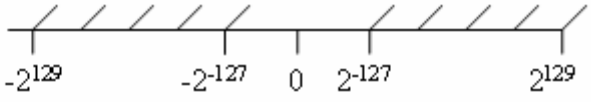
Формат операнда и диапазон допустимых значений представлен на рис. 2.

Диапазон допустимых значений операнда ФФТ меньше диапазона допустимых значений операнда ФПТ.

Преобразователь чисел ФПТ в ФФТ должен генерировать флаг ошибки при выходе за диапазон допустимых значений операнда ФФТ, а при преобразовании числа

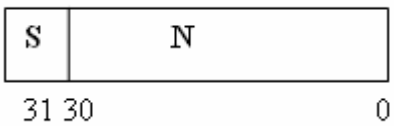


Формат операнда с плавающей точкой

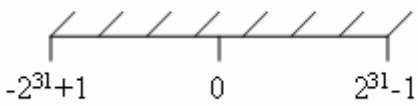


Диапазон допустимых значений операнда

Рис. 1



Формат операнда с фиксированной точкой



Диапазон допустимых значений операнда

Рис. 2

ФФТ в ФПТ может наблюдаться некоторая погрешность, так как мантисса числа ФПТ ограничена 24 битами, а модуль числа ФФТ содержит 31 бит.

Описание преобразователей на языке VHDL

Числа ФФТ и ФПТ, рассматриваются как 32-разрядные бит–векторы. В качестве входного операнда используется бит–вектор ***a***, в качестве выходного – ***b***. Вводятся дополнительные переменные: ***a1, a_inv1, a_inv, ind, i, b_x, count, M_i, M_u,*** которые относятся к разным типам данных и имеют определенные назначение.

Преобразование чисел ФФТ в ФПТ. Рассмотрим реализацию преобразователя ФП1, который основан на базовых конструкциях языка VHDL.

Процесс описания преобразователя состоит из 5 этапов.

1. С помощью конструкции ***if*** формируется дополнительная переменная ***a1***, в которой число представлено в прямом коде.
2. С помощью конструкции ***For*** определяется индекс бита, в котором первый раз встречается единица при просмотре ***a1*** слева направо (от старших значащих бит к младшим).
3. С помощью трех конструкций ***if*** формируется мантисса числа ФПТ.
4. Определяется значение порядка числа ФПТ.
5. Значения знакового бита числа ФФТ записываются в знаковый бит числа ФПТ.

Как было описано выше, не все коды, полученные в результате описания, могут быть синтезированы в системе САПР ПЛИС. Например, конструкции, в которых в качестве индексов при доступе к бит–вектору, использовались переменные, не по-

```
b(31 downto 0)<="00000000000000000000000000000000";
if a(30 downto 0)="00000000000000000000000000000000" then
        a1(30 downto 0):=a(30 downto 0);
else
        if a(31)='0' then
                a1(30 downto 0):=a(30 downto 0);
        else
                a_inv1:=conv_integer(a(30 downto 0))-1;
                a_inv(30 downto 0):=conv_std_logic_vector(a_inv1,31);
                a1(30 downto 0):=not(a_inv(30 downto 0));
        end if;
end if;

ind:=31;
for i in 30 downto 0 loop
        if a1(i)='1' then
                ind:=i;
                if ind<24 and ind>0 then
                        b(22 downto 23-ind)<=a1(ind-1 downto 0);
                end if;
                if ind>23 then
                        b(22 downto 0)<=a1(ind-1 downto ind-23);
                end if;

                b(30 downto 23)<=conv_std_logic_vector(127+ind,8);
                exit;
        end if;
end loop;

b(31)<=a(31);
rdy<='0';
```

зволяют проходить этап синтеза в ранних версиях САПР Xilinx ISE Foundation.

Преобразователь ФФТ в ФПТ может быть модифицирован, этап, в котором формируется мантисса числа ФПТ, может быть вынесен из конструкции ***For***, данный подход реализован в преобразователе ФП2.

Одним из способов организации устройства может быть использование конструкции ***Case*** (машины состояний). Принцип конструкции состоит в том, что в каждом состоянии выполняются заданные операции и определяется переход к следующему состоянию. Операции в каждом состоянии выполняются по новому синхросигналу. Данный подход используется в том случае, когда требуется уменьшить длительность синхросигнала, что приводит к увеличению аппаратных затрат. В конструкции ***Case*** легко организовать любые циклы, в том числе с переменным количеством итераций, что не всегда возможно при использовании конструкции ***For***, которая позволяет использовать ограниченное количество итераций.

На основе указанных свойств рассмотрим реализацию преобразователя ФП3. Преобразователь представляет машину состояний, имеющую семь состояний, связанных между собой условными переходами. Переход в первое состояние происходит с помощью оператора присвоения "***state <=S1***;" Последующие переходы осуществляются после выполнения операции данного состояния. В последнем состоянии (седьмом) выполняются операции и устройство переходит в ожидание нового сигнала загрузки "***load='1'***".

Преобразование чисел ФПТ в ФФТ. Рассмотрим реализацию преобразователя ПФ2. Он генерирует флаг ошибки "***error_x<='1'***" при выходе за диапазон допустимых значений операнда ФФТ.

Процесс описания преобразователя состоит из 7 этапов.

1. С помощью двух конструкций ***if*** отбираются все числа ФПТ, значение порядка которых не входит в допустимый диапазон $127 \le E \le 157$. Числа ФПТ, значение порядка которых больше 157, выходят за диапазон допустимых значений числа ФФТ, а числа, степень порядка которых меньше 127, при преобразовании в ФФТ представляются нулевым значением.

2. Значения знакового бита числа ФПТ записывается в знаковый бит числа ФФТ.

3. С помощью операции присвоения в дополнительную переменную ***b_x*** записывается значение мантиссы числа ФПТ вместе со скрытым битом.

4. Значение ***ind*** – число, которое определяет количество сдвигов вправо (в сторону младших значащих бит) дополнительной переменной.

5. Выполняется сдвиг вправо с помощью ***SHR***-функции, которая входит в пакет STD_LOGIC_UNSIGNED.

6. Анализируется знак дополнительной переменной и осуществляется ее перевод в дополнительный код.

7. Значение дополнительной переменной присваивается выходному регистру в качестве результата.

Альтернативой функции сдвига ***SHR*** может быть анализ переменной ***ind*** и перезапись части бит–вектора входного операнда ***a*** в бит–вектор выходного операнда ***b***. Данный подход, реализованный в преобразователе ПФ1, характеризуется использованием переменных в качестве индексов бит–вектора и представляется только последними версиями САПР Xilinx ISE Foundation.

Преобразователь чисел ФПТ в ФФТ может быть представлен как машина, имеющая четыре состояния. Этот подход, реализован в преобразователе ПФ3.

Вход в первое состояние осуществляется по сигналу загрузки "***load='1'***". Данное состояние реализует анализ входного операнда в ФПТ: если операнд может быть представлен в формате с ФФТ, то осуществляется переход в состояние 2, в противном случае – переход в состояние 3. В состоянии 2 непосредственно реализуется преобразование операндов.

Аппаратные (в пересчете на количество слайсов, которые определяют логическую сложность кристалла ПЛИС) и временные оценки для предложенных реализаций модулей преобразователей форматов представлены в табл. 1.

Для варианта ФП3 время преобразования (T_{min} = 20 нс) имеет место только в

```
if load='1' then
        rdy<='1';
        i:=0;
        state <= S1;
end if;
case state is
        when S1 =>
                if a(30 downto 0)="0000000000000000000000000000000" then
                        state <= S7;
                else
                        b(30 downto 0)<="0000000000000000000000000000000";
                        if a(31)='0' then  a1(30 downto 0):=a(30 downto 0);
                        else a_inv1:=conv_integer(a(30 downto 0))-1;
                        a_inv(30 downto 0):=conv_std_logic_vector(a_inv1,31);
                        a1(30 downto 0):=not(a_inv(30 downto 0));
                        end if;
                        state <= S2;
                end if;
        when S2 =>
                for ind in 30 downto 0 loop
                        if a1(ind)='1' then
                                i:=ind;
                                state <= S3;
                                exit;
                        end if;
                end loop;
        when S3 =>
                if i=0 then
                        b(0)<='1';
                        state <= S6;
                else    state <= S4;
                end if;
        when S4 =>
                if i<24 then
                        b(22 downto 23-i)<=a1(i-1 downto 0);
                        state <= S6;
                else
                        state <= S5;
                end if;
        when S5 =>
                b(22 downto 0)<=a1(i-1 downto i-23);
                state <= S6;
        when S6 =>
                b(30 downto 23)<=conv_std_logic_vector(127+i,8);
                b(31)<=a(31);
                rdy<='0';
        when S7 =>
                b(31 downto 0)<=a(31 downto 0);
                rdy<='0';
        when others =>
                null;
end case;
```

```
rdy<='1';
error_x<='0';
b_x(31 downto 0):="00000000000000000000000000000000";
    if a(30 downto 23)>="01111111" then
        if a(30 downto 23)>"10011101" then
            error_x<='1';
        else
            b_x(31):=a(31);
            b_x(30 downto 7):='1'&a(22 downto 0);
            ind:=157-conv_integer(a(30 downto 23));
            count:=conv_unsigned(ind,5);
            M_i:=conv_integer(b_x(30 downto 0));
            M_u:=conv_unsigned(M_i,31);
            M_u:= SHR(M_u,count);
            b_x(30 downto 0):=conv_std_logic_vector(M_u,31);

              if b_x(31)='1' then
                  b_inv(30 downto 0):=not(b_x(30 downto 0));
                  b_inv1:=conv_integer(b_inv(30 downto 0))+1;
            b_x(30 downto 0):=conv_std_logic_vector(b_inv1,31);
            end if;

        end if;
    else
        error_x<='1';
    end if;

b(31 downto 0)<=b_x(31 downto 0);
rdy<='0';
```

том случае, если значение исходного операнда представляется нулевым значением, т. е. вероятность этого случая есть $(1/2^{31})$, во всех остальных случаях время преобразования определяется величиной T_{max} = 60 нс.

Для варианта ПФ3 время преобразования (T_{min} = 16 нс) имеет место только в том случае, если значение степени порядка не входит в диапазон представления результирующего операнда ФФТ, т. е. вероятность этого случая есть $(1/(2^8 - 2^5 + 1))$, во всех остальных случаях время преобразования определяется величиной T_{max} = 24 нс.

Верификация преобразователей

Верификация предложенных структур преобразователей осуществлена методом моделирования в системе ModelSim Xilinx Edition - MXE II. В табл. 2 представлены результаты верификации процесса преобразования массива входных операндов ФФТ в ФПТ, а также, для большей наглядности, десятичные значения этих чисел.

В табл. 3 приведены результаты верификации процесса преобразования массива входных операндов ФПТ в ФФТ, а также десятичные значения этих чисел и значения флага ‘Ошибка’ (Er). Этот флаг принимает значение “1” (когда результат выходит за диапазон представления допустимых значений операнда в ФФТ) и значение “0” (результат входит в диапазон представления допустимых значений операнда в ФФТ).

Задача выбора оптимальной структуры

В результате синтеза имеем несколько структурных реализаций алгоритма преобразования форматов ФП1, ФП2, ФП3 для преобразования ФФТ в ФПТ и ПФ1, ПФ2, ПФ3 для обратного преобразования форматов. Решение задачи выбора оптимальной структурной ПЛИС–реализации алгоритма не может быть определено стро-

```
if load='1' then
    rdy<='1';
    error_x<='0';
    b_x(31 downto 0):="00000000000000000000000000000000";
    state <= S1;
end if;
case state is
when S1 =>
    if a(30 downto 23)>="01111111" and a(30 downto 23)<="10011101" then
        state <= S2;
    else    state <= S3;
    end if;
when S2 =>
    ind:=157-conv_integer(a(30 downto 23));
    count:=conv_unsigned(ind,5);
    M_i:=conv_integer('1'&a(22 downto 0)&"0000000");
    M_u:=conv_unsigned(M_i,31);
    M_u:= SHR(M_u,count);
    b_x(30 downto 0):=conv_std_logic_vector(M_u,31);
    if a(31)='1' then
        b_inv(30 downto 0):=not(b_x(30 downto 0));
            b_inv1:=conv_integer(b_inv(30 downto 0))+1;
        b_x(30 downto 0):=conv_std_logic_vector(b_inv1,31);
    end if;
    state <= S4;
when S3 =>
    error_x<='1';
    b(31 downto 0)<=b_x(31 downto 0);
when S4 =>
    b(31)<=a(31);
    b(30 downto 0)<=b_x(30 downto 0);
    rdy<='0';
when others =>
    null;
```

гими аналитическими целевыми функциями. Кроме того, необходимо учитывать наличие довольно мощного ряда кристаллов ПЛИС, каждая из имеющихся серий представлена множеством кристаллов, отличающихся быстродействием, потребляемой мощностью, логической емкостью, типом корпуса, количеством выводов и другими важными параметрами.

Целевая функция в аналитическом виде может быть найдена одним из приближенных методов. Наиболее распространен-

Таблица 1. **Аппаратные и временные оценки модулей преобразователей форматов**

Тип модуля	Количество слайсов	Период синхросигналов Clk (нс)	Время реализации операции преобразования форматов (нс)	
			T_{min}	T_{max}
ФП1	396	19	19	19
ФП2	290	23	23	23
ФП3	390	14	2*10=20	6*10=60
ПФ1	248	16	16	16
ПФ2	109	14	14	14
ПФ3	153	14	2*8=16	3*8=24

Таблица.2. **Результаты верификации преобразования ФФТ в ФПТ**

Значения входных операндов ФФТ в дополнительном коде	Значения входных операндов (десятичные)	Значения выходных операндов ФПТ	Значения выходных операндов (десятичные)
00000000000000000000000000000000	0	00000000000000000000000000000000	0
10000000000000000000000000000100	-2147483644	11001110111111111111111111111111	-2147483520
00000000000000000000000000010000	16	01000001100000000000000000000000	16
00000000000000000000000001000000	64	01000010100000000000000000000000	64
10000000000000000000000100000000	-2147483392	11001110111111111111111111111110	-2147483392
00000000000000000000010000000001	1025	01000100100000000010000000000000	1025
00000000000000000001000000000100	4100	01000101100000000010000000000000	4100
00000000000000000100000000010000	16400	01000110100000000010000000000000	16400
10000000000000010000000001000000	-2147418048	11001110111111111111110111111111	-2147417984
10000000000001000000000100000000	-2147221248	11001110111111111111011111111110	-2147221248
00000000001000000000010000000001	1049601	01001001100000000010000000001000	1049601
00000000100000000001000000000100	4198404	01001010100000000010000000001000	4198404
10000001000000000100000000010000	-2130690032	11001110111111101111111101111111	-2130689920
00000100000000010000000001000000	67174464	01001100100000000010000000001000	67174464
10010000000001000000000100000000	-1878785792	11001110110111111111011111111110	-1878785792

Таблица.3. **Результаты верификации преобразования ФПТ в ФФТ**

Значения входных операндов ФПТ	Значения входных операндов (десятичные)	Значения выходных операндов ФФТ в дополнительном коде	Er	Значения выходных операндов (десятичные)
00000000000000000000000000000000	0	00000000000000000000000000000000	1	0
01000000110000000000000000000000	6	00000000000000000000000000000110	0	6
11000001110000000000000000000000	-24	11111111111111111111111111101000	0	-24
01000010010000000000000000000000	48	00000000000000000000000000110000	0	48
01001010010000000000000000000000	3145728	00000000001100000000000000000000	0	3145728
11000110010000000000000000000000	-12288	11111111111111111101000000000000	0	-12288
01001011010000000000000000000000	12582912	00000000110000000000000000000000	0	12582912
01000111010000000000000000000000	49152	00000000000000001100000000000000	0	49152
11000111110000000000000000000000	-98304	11111111111111101000000000000000	0	-98304
01001000010000000000000000000000	196608	00000000000000011000000000000000	0	196608
01001000110000000000000000000000	393216	00000000000000110000000000000000	0	393216
11001011110000000000000000000000	-251165824	11111110100000000000000000000000	0	-251165824
11001110110000000000000000000000	-1610612736	10100000000000000000000000000000	0	-1610612736
01001111110000000000000000000000	6442450944	00000000000000000000000000000000	1	0

ными являются методы линейной или нелинейной интерполяции и экстраполяции по нескольким опорным точкам. Эти точки (структурные реализации алгоритма) получаются путем предварительного формирования вариантов реализаций алгоритма или взяты из набора, имеющихся в составе САПР, готовых CORE - ядер.

Из множества этих точек, где каждой l – й точке соответствует ПЛИС–реализация с параметрами $<T_l, Q_l>$, формируется множество Парето мощностью M на плоскости $T-Q$ с учетом соотношений

$$T_1 \le T_2 \le \ldots \le T_l \le \ldots \le T_m;$$

$$Q_1 \ge Q_2 \ge \ldots \ge Q_l \ge \ldots \ge Q_m.$$

В общем виде задача выбора оптимального варианта реализации алгоритма сводится к минимизации функционала

$$L_l = \alpha T_l + \beta Q_l \Rightarrow \min \quad (1)$$

с учетом ограничений

$$\begin{cases} T_l \le T_0; \\ Q_l \le Q_0, \end{cases} \quad (2)$$

где α, β – весовые коэффициенты, которые могут быть определены, например, методом экспертных оценок; T_0, Q_0 – заданные предельные значения параметров T_l и Q_l.

Если заданным ограничениям (2) удовлетворяет единственная точка множества Парето, то реализация, соответствующая этой точке, является результирующей. Если ограничениям (2) удовлетворяют несколько таких точек, то необходимо минимизировать функционал (1).

Выводы

Рассмотренные особенности конструкций языка VHDL позволяют сделать вывод о том, что при поведенческом описании структур для преобразования форматов не все конструкции могут быть использованы для ввода проекта в САПР ПЛИС, так как этап синтеза в этих системах не поддерживает полный набор конструкций. Использование инструментальных средств FPGA Express и программ размещения и трассировки системы Xilinx Foundation Series позволило осуществить синтез и реализацию преобразователей в кристалле XCV600E–BG432–8, аппаратные и временные оценки которых приведены в табл.1. Полученное множество реализаций позволяет выбрать оптимальный вариант структурной реализации преобразования форматов по критериям аппаратных и временных затрат. По результатам анализа табл. 1 можно сделать следующие выводы. Для преобразования ФФТ в ФПТ вариант ФП1 является самым быстродействующим, при этом требует для реализации наибольшего количества слайсов. Вариант ФП2 является менее быстродействующим по сравнению с ФП1 и аппаратных затрат требуется меньше. Вариант ФП3 для Т(макс) по сравнению с вариантом ФП1 требует меньшего количества слайсов и является более медленным, но по сравнению с вариантом ФП2 реализуется большими аппаратными затратами и является еще медленнее, поэтому эта реализация может быть исключена из дальнейшего рассмотрения. Вариант ФП3 для Т(мин) по сравнению с вариантом ФП1 требует меньше слайсов и обладает меньшим быстродействием, но по сравнению с вариантом ФП2 требует больших аппаратных затрат и меньшего времени преобразования.

Для преобразования ФПТ в ФФТ вариант ПФ2 по сравнению с вариантами ПФ1 и ПФ3 обладает самым высоким быстродействием, при этом для его реализации используется наименьшее количество слайсов. Поэтому варианты ПФ1 и ПФ3 могут быть исключены из дальнейшего рассмотрения. Это связано с использованием функции сдвига вправо SHR, которая входит в пакет STD_LOGIC_UNSIGNED.

Синтезированные преобразователи верифицированы методом моделирования в системе ModelSim Xilinx Edition - MXE II с помощью стенда для проверки, что подтверждает правильность их функционирования.

1. *Hollash S.* IEEE Standard 754 Floating Point Numbers / Available. - htpp:// IEEE / IEEE Standard 754 Floating-Point.htm.
2. *Xilinx Data* Book 2004 / Available. - http: // support.xilinx.com /partinfo /databook.htm.
3. *VHDL'93*. IEEE Standard VHDL Language Reference Manual. IEEE Std 1076-1993. – 289 p.

Получено 21.09.05

Об авторах

Опанасенко Владимир Николаевич
канд. техн. наук, ст. науч. сотрудник

Место работы автора
Институт кибернетики им. В.М. Глушкова
Киев, просп. Акад. Глушкова, 40
Тел.(сл.). (044) 526 2598
Тел. (дом.). (044) 530 7091
E-mail: vlopanas@ukr.net

Лисовый Александр Николаевич
студент
Место работы автора
Национальный авиационный университет
Киев, просп. Комарова, 1
Тел.(сл.). (044) 526 2598
Тел. (дом.). (044) 402 9341
E-mail: vlopanas@ukr.net