Выводы

Разработан метод измерения частотных параметров СВЧ-сигнала в волноводе. По сравнению с существующими на сегодняшний день методами предложенный метод отличается применимостью в широком диапазоне длин волн, включая миллиметровые и субмиллиметровые, простотой технической реализации, возможностью автоматизации измерительного процесса, а также возможностью получения результатов измерения в реальном времени.

Научная новизна проведенных исследований заключается в нахождении оптимальных соотношений геометрических параметров датчиков и способов их расположений, позволивших получить максимальную чувствительность метода.

Практическая значимость состоит в упрощении процесса измерения частоты СВЧ-колебаний за счет его полной автоматизации.

Список литературы: 1. *Измерения* на миллиметровых и субмиллиметровых волнах: Методы и техника/ Р.А. Валитов, С.Ф. Дюбко, Б.И. Макаренко и др.; Под ред. Р.А. Валитова, Б.И. Макаренко. М.: Радио и связь, 1984. 256 с. **2.** *Волков. В.М.* Проектування засобів вимірювання прохідної потужності: Навч. посібник. Харків: ХТУРЕ, 2000. 160 с.

Поступила в редколлегию 29.08.2007

Захаров Игорь Петрович, д-р техн. наук, профессор кафедры метрологии и измерительной техники ХНУРЭ. Научные интересы: неопределенность измерений различных физических величин. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-1331.

Сергиенко Марина Петровна, канд. техн. наук, ассистент кафедры метрологии и измерительной техники ХНУРЭ. Научные интересы: динамические измерения. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 702-1331.

УДК 519.713:681.326

И.В.ХАХАНОВА

СИНТЕЗ МОДЕЛЕЙ УПРАВЛЯЮЩИХ АВТОМАТОВ ДЛЯ SOC-ФИЛЬТРОВ С КОНВЕЙЕРНОЙ АРХИТЕКТУРОЙ

Анализируются существующие модели цифровых автоматов и способы их реализации на микросхемах программируемой логики. Предлагаются две модели управляющих автоматов для устройств с конвейерной архитектурой, реализующие DSP задачи обработки изображений на основе SoC. Разрабатывается программный модуль автоматической генерации HDL-кода для упомянутых типов управляющих автоматов.

1. Введение

Сложность цифровых систем и сетей, имплементированных в кристаллы PLD, согласно закону Мура [1] удваивается каждые полтора года. Они представляют собой функционально и конструктивно законченные устройства, реализованные в чипе, и содержат микропроцессоры, блоки памяти, контроллеры, периферийные устройства, порты ввода-вывода информации. В такие архитектуры в последнее время довольно часто включаются блоки, решающие задачи цифровой обработки сигналов (DSP – digital signal processing). Для повышения быстродействия DSP-приложений применяется конвейерная архитектура, которая позволяет обрабатывать большие потоки входных данных при высокой частоте синхронизации. DSP-модуль можно представить в виде структуры, показанной на рис. 1, которая состоит из операционного автомата (ОА) в виде конвейера, управляющего автомата (УА) и блоков памяти различной размерности, используемой для хранения исходных данных, промежуточных результатов и выходной информации DSP-преобразования.

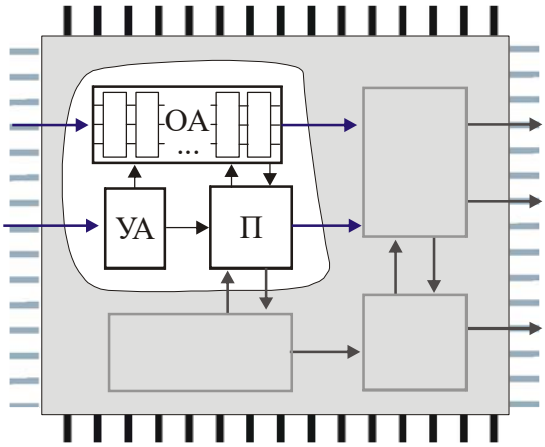


Рис. 1. DSP-модуль в системе на кристалле

Цель работы – автоматическая генерация управляющих автоматов для конвейерных вычислительных архитектур SoC фильтров, реализуемых в кристаллах PLD, что дает возможность существенно (x10, x100) повысить быстродействие обработки видеоинформации по сравнению с программным способом.

Для достижения поставленной цели в работе решаются следующие *задачи*.

1. Анализ моделей цифровых автоматов и способов их реализации в аппаратуре на основе PLD с помощью языков VHDL и Verilog.

2. Разработка моделей управляющих автоматов для SoC фильтров, реализующих стандарт JPEG 2000 с конвейерной обработкой данных.

3. Разработка шаблонов VHDL- и Verilog-моделей в целях автоматической генерации управляющих автоматов для операционных устройств, реализующих стандарт JPEG 2000 с конвейерной обработкой данных.

4. Программная реализация процедур автоматической генерации VHDL и Verilog-моделей управляющего автомата для конвейерных архитектур.

2. Модели цифровых автоматов. Абстрактный автомат

Абстрактный автомат [2] является математической моделью дискретного устройства и определяется вектором $S=(A,Z,W,\varphi,\psi,a_1)$, где векторы $A=(a_1,...,a_m,...a_M)$; $Z=(z_1,...,z_m,...z_M)$ и $W=(w_1,...,w_g,...,w_G)$ – множества состояний входных и выходных сигналов; $\varphi: A\times Z\to A$ – функция переходов, реализующая отображение $D_\varphi =\subseteq A\times Z$ в A, когда парам <состояние - входной сигнал> (a_m,z_f) функция φ ставит в соответствие состояния автомата $a_s=\varphi(a_m,z_1)$, $a_s\in A$; $\psi: A\times Z\to W$ – функция выходов, реализующая отображение $D_\lambda =\subseteq A\times Z$ в W, когда функция ψ парам <состояние - выходной сигнал> (a_m,z_f) ставит в соответствие выходные сигналы автомата $w_s=\psi(a_m,z_1)$; $a_1\in A$ – начальное состояние автомата.

Абстрактный автомат (рис.2) имеет один вход и один выход. Он функционирует в дискретном времени, принимающем целые неотрицательные значения t = 0, 1, 2, ... В каждый момент времени t автомат находится в некотором состоянии a(t) из множества состояний автомата, причем в начальный момент времени t=0 он всегда находится в начальном состоянии $a(0)=a_1$. В момент t, в состоянии a(t), автомат может принимать букву входного алфавита $z(t)\in Z$. Функцией выходов $w(t)=\psi(a(t),z(t))$ формируется соответствующая буква выходного алфавита, а функцией перехода $a(t+1)=\varphi(a(t),z(t))$ – следующее состояние. Идея абстрактного автомата состоит в реализации некоторого отображения множества слов входного алфавита Z в множество слов выходного алфавита W.

Рис 2. Абстрактный автомат

На практике наиболее распространены два класса автоматов – Мили и Мура (G.H.Mealy и E.F.Moore), функционирование которых задается уравнениями:

1) $a(t+1) = \varphi(a(t), z(t));\ w(t) = \psi(a(t), z(t)), t = 0,1,2,\ldots$.

2) $a(t+1) = \varphi(a(t), z(t));\ \ w(t) = \psi(a(t)), t = 0,1,2,\ldots$.

Автомат называется конечным, если конечны множества A, Z и W. Автомат называется полностью определенным, если $D_\varphi = D_\psi = A \times Z$. Для полностью определенного автомата области определения функций φ и ψ совпадают со множеством пар вида (a_m, z_f). У неполностью определенного или частичного автомата функции φ и ψ определены не для всех пар $(a_m, z_f) \in A \times Z$.

3. Структурный автомат

Здесь учитывается структура входных и выходных сигналов автомата, а также внутреннее устройство на уровне структурных схем. Основной задачей структурной теории автоматов является нахождение общих приемов построения структурных схем автоматов на основе композиции элементарных автоматов. При построении конечных автоматов функции переходов φ и выходов ψ реализуются с помощью комбинационных схем CL_φ и CL_ψ соответственно, а память автомата реализуется с помощью регистра RG, который хранит код внутреннего состояния в каждый момент автоматного времени. В зависимости от способов определения функций переходов и выходов в [3] представлена детальная классификация конечных автоматов (рис. 2). Классы A (рис. 3, а) и B (рис. 3, б) соответствуют общей форме автоматов Мили и Мура. Если каждый выходной набор w(t) автомата Мура совпадает с кодом его внутреннего состояния a(t), то данный автомат относится к классу C [2, 3] (рис.3, в) и его поведение описывается уравнениями: $a(t+1) = \varphi(a(t), z(t));$ $w(t) = a(t)$. Особенностью автомата C является отсутствие комбинационной схемы CL_ψ, что делает его выходы регистровыми. Если каждый выходной набор w(t) автомата Мили совпадает с кодом его следующего состояния a(t+1), то рождается автомат класса D [3,5] (рис. 3, г). Его особенностью также является отсутствие комбинационной схемы CL_ψ, однако значения входных функций формируются на входах памяти автомата RG.

Автомат D описывается уравнениями: $a(t+1) = \varphi(a(t), z(t));\ \ w(t) = a(t+1)$. Различие между автоматами классов C и D: в первой модели выходной набор w(t) автомата совпадает с кодом его текущего состояния a(t), во втором – с кодом следующего состояния a(t+1). Поскольку в автомате класса C выходной набор не зависит от значений входов, то он является частным случаем автомата Мура. Автомат D является частным случаем автомата Мили, поскольку выходные значения зависят от текущего состояния a(t) и входов z(t). Преимущество автоматов классов C и D перед автоматами A и B заключается в более простой схемной реализации. Однако не всякое устройство может быть построено с помощью автоматов классов C и D. Коды внутренних состояний конечного автомата могут также полностью определяться значениями входных переменных. Если каждый входной набор z(t) автомата Мили совпадает с кодом следующего состояния a(t+1), то такой автомат соответствует классу E (рис. 3, д), уравнения функционирования которого имеют вид: $a(t+1) = z(t);\ \ w(t) = \psi(z(t), a(t))$. Если каждый входной набор z(t) автомата Мура совпадает с кодом следующего состояния a(t+1), то такой автомат соответствует классу F (рис. 3, е), поведение которого может быть описано уравнениями: $a(t+1) = z(t);\ \ w(t) = \psi(a(t))$. Особенностью архитектур автоматов класса E и F является отсутствие комбинационной

схемы CL_φ. В автомате класса E комбинационная схема CL_ψ имеет связь со входом автомата, а в автомате класса F ее нет. Автоматы E и F не имеют цепей обратной связи. В автомате класса G (рис. 3, ж) каждый выходной набор w(t) автомата Мили совпадает с кодом его внутреннего состояния a(t), а каждый входной набор – с кодом состояния a(t+1). Автомату класса G соответствует обычный регистр. На практике редко удается непосредственно реализовывать автоматы классов C - F. В работе [3] выполнен анализ синтеза управляющих автоматов на микросхемах программируемой логики SPLD и CPLD.

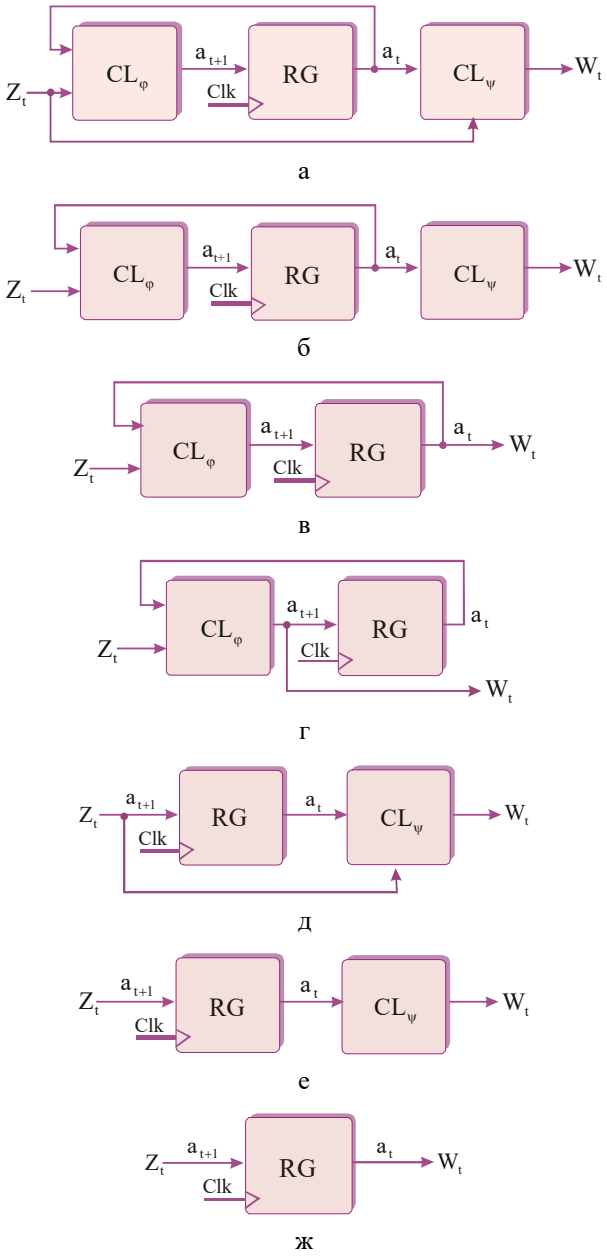


Рис. 3. Основные структурные схемы конечных автоматов: а – Мили класса A; б – Мура класса B; в – Мура класса C; г – Мили класса D; д – Мили класса E; е – Мура класса F; ж – Мура класса G

4. Сложность реализации автоматов

Пусть конечный автомат характеризуется числом L входных переменных множества $X=\{x_1, ..., x_L\}$, числом N выходных переменных множества $Y = \{y_1, ..., y_N\}$ и числом

разрядов R кода внутренних состояний, которое может быть определено в диапазоне от $\log_2 M$ до M, где M – число внутренних состояний автомата.

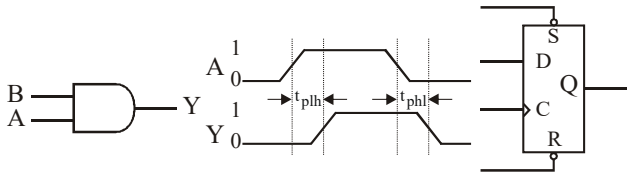
В работе [3] указывается, что на сложность реализации конечного автомата на современных PLD и на выбор модели влияет: тип входа или выхода – регистровый (t_r) или комбинационный (t_c); число входных ячеек n_B; число триггеров n_{FF}; число скрытых макроячеек n_{BMC} , когда буферизация входных сигналов осуществляется с помощью скрытых макроячеек; суммарное число внешних выводов n_{P+BMC} и скрытых макроячеек. Значение этих параметров остается важным для реализации автоматов на SPLD или CPLD. Однако в связи с современным развитием электроники и появлением систем и даже сетей на кристаллах уже нет таких жестких требований к уменьшению размерности устройства, уменьшению входных и выходных переменных, которые являются внутренними линиями микросхемы. Основным параметром, определяющим сложность реализации управляющего автомата, является его размерность, которая выражается в эквивалентных вентилях. Современный рынок электронных технологий предлагает многообразие архитектур и технологий изготовления микросхем PLD и ASIC, которые своими компонентами образуют функции устройства. Каждому модулю микросхемы ставится в соответствие некоторое количество эквивалентных вентилей в качестве метрики (системы) оценивания сложности проектов, реализованных в различных chipset платформах.

Быстродействие и аппаратная сложность являются взаимно противоречивыми понятиями при оптимизации устройства в процессе его реализации. Для повышения рабочей частоты вводятся дополнительные входные и выходные регистры, что увеличивает размерность схемы. Если необходимо получить устройство с меньшим количеством ресурсов, то для этого нужно пожертвовать быстродействием или потребляемой мощностью.

5. Оценка временных характеристик автомата

При вычислении временных характеристик устройства используются следующие параметры:

1) Propagation Delay – задержка распространения – время между поступлением входного сигнала и появлением выходной реакции. Интервал [tplh, tphl], изображенный на рис. 4, а, есть время между событием на входе элемента и изменением его выхода (0–1) и (1–0) соответственно. Величина измеряется между 50% точками уровней входного и выходного сигналов (рис. 4, б). Фрагмент индекса lh и hl обозначает изменение выходного сигнала. Для триггеров (рис. 4, в) определены следующие временные параметры: T_{CQ} – время, необходимое для изменения выхода Q в ответ на поступление активного фронта синхросигнала. При этом выходное значение Q зависит от входа данных D; T_{SQ}, T_{RQ} – задержка изменения выхода Q в ответ на поступление асинхронных сигналов установки (S) и сброса (R). Задержка между синхронным входом данных D и выходом Q не определена, поскольку изменение входа D не приводит к непосредственной модификации состояния триггера;



а б в

Рис. 4. Задержки комбинационного (а, б) и последовательностного (в) элементов

2) Setup Time – время установки T_{su} (рис. 5) – это интервал времени до появления фронта синхросигнала, в течение которого синхронный входной сигнал не может изменять свое значение;

3) Hold Time – время хранения T_{hd} (см. рис. 5) – это интервал времени, следующий за фронтом синхроимпульса, в течение которого синхронный входной сигнал не может изменять значение.

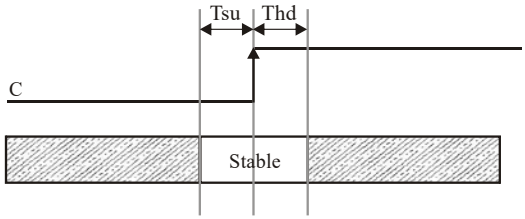


Рис. 5. Время установки и время хранения

Модель последовательностного устройства для вычисления временных параметров представлена на рис. 6. Здесь определены три пути следования сигналов:

1) Clock to Output – период времени между поступлением активного фронта синхросигнала и появлением соответствующей реакции на выходе:

$$T_{CO} = T_{c2q} + T_{comb\_Q2Omax}, \quad (1)$$

где T_{c2q} – задержка регистра RG; T_{comb_Q2Omax} – самый длинный путь от выхода регистра RG к любому выходу схемы $w_i(t)$;

2) Register to Register – период времени между поступлением активного фронта синхросигнала на вход CLK регистра RG и формированием значений на входах D:

$$T_{RR} = T_{c2q} + T_{comb\_Q2Dmax} + T_{su}, \quad (2)$$

где T_{comb_Q2D} – самый длинный путь от выхода триггера Qdff регистра RG до входа триггера Ddff регистра RG;

3) Pin to Pin – максимальный логический путь от входов автомата до его внешних выходов, не содержащий последовательностных элементов:

$$T_{PP} = T_{comb\_I2Omax}. \quad (3)$$

Максимальная рабочая частота любого последовательностного устройства определяется выражением:

$$F = \frac{1}{P_{min}}, \quad (4)$$

где F – максимальная рабочая частота; P_{min} – период времени, соответствующий самому длинному логическому пути схемы (структурная глубина), который равен:

$$P_{min} = \max(T_{CO}, T_{RR}, T_{PP}). \quad (5)$$

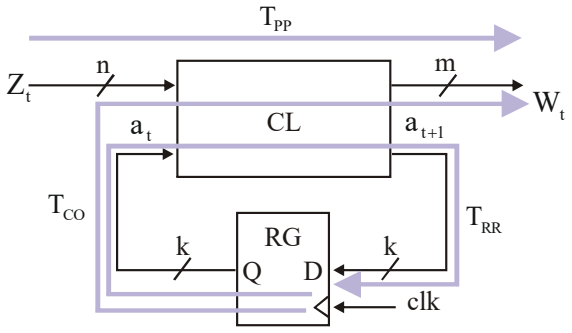


Рис. 6. Временная модель автомата

6. Сеть автоматов

Если цифровое устройство включает несколько автоматов управления, то их совместная работа может быть описана с помощью сети автоматов [2], которая задается вектором: $N = (Z, \{S_i\}, W, \{f_i\}, \{\psi_i\}, g)$, Z – входной алфавит, $\{S_i = (A_i, Z_i, \delta_i)\}$, $1 \le i \le n$ – множество компонентных автоматов (КА) сети, один из которых (полуавтомат) изображен на рис. 7:

$$Z_i = \begin{cases} Z_i^{'} \times Z_i^{''} & \leftarrow \quad Z_i^{'} \neq \varnothing; \\ Z_i^{''} & \leftarrow \quad Z_i^{'} = \varnothing, \end{cases}$$

$Z_i^{'}, Z_i^{''}$ – внутренний и внешний входные алфавиты S_i. Функция переходов S_i $\delta_i : A_i \times Z_i \to A_i$;

W – выходной алфавит сети; $\{f_i : (\underset{j}{\times} A_j) \to Z_i^{'}\}$, $1 \le i, \quad j \le n$ – множество функций соединения

КА сети; $\{\psi_i : Z \to Z_i^{'}\}, 1 \le i \le n$ – множество входных функций; $g : (\underset{i}{\times} A_i) \times Z \to W$ – выходная функция сети. Множества $\{S_i\}$ и $\{f_i\}$ являются базисом и структурой сети [2].

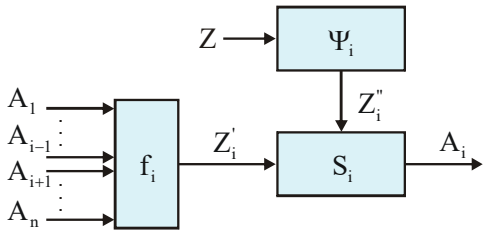


Рис. 7. Компонентный автомат сети

Сеть является общей моделью совместной работы совокупности из n автоматов, которая представлена на рис. 8.

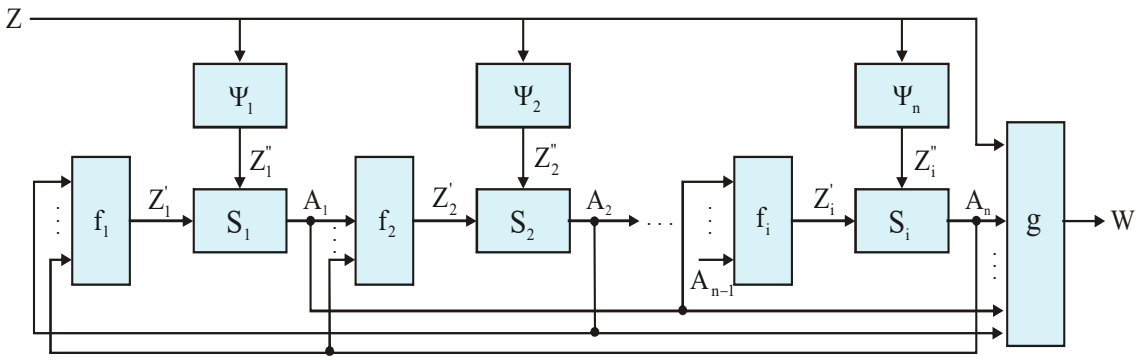


Рис. 8. Сеть из n компонентных автоматов

7. Создание HDL-моделей цифровых автоматов

В зависимости от задач, которые ставит перед собой проектировщик, и необходимой степени детализации управляющий автомат может быть реализован в виде HDL-моделей различных стилей и уровней описания. Однако наиболее популярным является системный (поведенческий) уровень представления моделей. Кодирование состояний и создание схемы возлагается на современные программы синтеза (генерирования), которые специально разрабатываются для конкретного аппаратного базиса. Такой подход позволяет проектировать переносимые на различные электронные базисы модели, что значительно упрощает работу проектировщика и позволяет сосредоточить усилия на разработке функциональности.

VHDL-модель автоматов Мура и Мили классов A (см. рис. 3, а) и B (см. рис. 3, б) может быть описана с помощью двух процессов (листинг 1): первый Process_1 реализует комбинационные схемы автомата CL_φ и CL_ψ, второй Process_2 – описывает регистр RG. В программной модели используется символьное описание состояний State_type, реализуе-

мое с помощью VHDL-типа перечисления. Двоичные коды состояний определяются программой синтеза, что предоставляет свободу выбора значений кодов в зависимости от конкретного аппаратного базиса. Таблица переходов (для классов автомата), соответствующая функции переходов $a(t+1) = \varphi(a(t), z(t))$, реализуется в Process_1 с помощью операторов case/if . Поскольку входами комбинационной части CL_φ являются входы автомата и его текущее состояние, то эти сигналы записываются в список чувствительности процесса. Стиль 1 предлагает задавать выходные функции $w(t) = \psi(z(t), a(t))$ автомата Мили CL_ψ с помощью операторов case/if, или применять оператор case для автомата Мура: $w(t) = \psi(a(t))$). Стиль 2 использует дополнительные условные параллельные операторы для реализации выходных функций. Process_2, описывающий регистр состояний, представляет собой программную модель реализуемого на D-триггерах регистра с асинхронным сбросом reset в начальное состояние. Для дополнительного управления автоматом в модели может быть использован сигнал разрешения синхронизации enable.

Листинг 1. Шаблон VHDL-модели автомата

```
library IEEE;
use  IEEE.std_logic_1164.all;
entity Имя_интерфейса is
 port ( Clk: in STD_LOGIC;
     Reset: in STD_LOGIC;
     Управляющие входы Z_i: in STD_LOGIC;
     Выходы W_i: out STD_LOGIC);
end entity Имя_интерфейса;
architecture Имя_арх of Имя_интерфейса is
-- Тип, использующий символьное кодирование состояний автомата
type State_type is (состояние1, состояние2, ...);
signal State, NextState: State_type;
begin

-- Процесс для вычисления значения следующего состояния и выходов
Process_1: process (State, Управляющие входы Z_i)
begin
 -- инициализация значений выходов
     W_i <= '0';
 case State is
     when состояние1 =>
          -- Присвоение значений выходам для автомата Мура, стиль 1
          if условие then
               NextState <= состояние i;
-- Присвоение значений выходам для автомата Мили, стиль 1
          else    NextState <= состояние j;
-- Присвоение значений выходам для автомата Мили, стиль 1
          end if;
     when ...
     when others => NextState <= состояние0;
-- Присвоение значений выходам для состояния -- по умолчанию. Установка автомата в на-
чальное
-- состояние
     end case;
end  process;

Process_2: process (Clk, reset)
begin
     if Reset='1' then
     -- Или Reset = '0',
     -- если активным является низкий уровень
     -- Начальное состояние
```

```
        State <= состояние0;
    elsif Clk'event and Clk = '1' then
        State <= NextState;
    end if;
end process;
-- Условные параллельные операторы
-- назначения, стиль 2
W_i <= '1' when условие else
        '0';
. . .
end ex1_arch;
```

Аналогичным образом создается Verilog-модель автомата (листинг 2). В Verilog нет типа перечисления, существующего в VHDL, поэтому для удобства работы с состояниями используются параметры, в которых задается конкретный двоичный код для каждого состояния. Однако эти значения являются достаточно условными, потому что программы синтеза сами выбирают коды для представления состояний автомата в зависимости от выбранного chipset – класса микросхем для реализации автомата. Подобно VHDL-модели, Verilog-структура автомата состоит из двух блоков always, реализующих комбинационную и последовательностные части автомата. Первый блок always соответствует комбинационным модулям автомата CL_{φ} и CL_{ψ} (см. рис. 2), второй блок always описывает регистр RG. Так же как и в модели VHDL, таблица переходов автомата обоих классов реализуется с помощью операторов case/if в always-блоке 1. Управляющие входы автомата и текущее состояние указываются в списке чувствительности блока. Подобно модели VHDL, описание выходных функций $w(t) = \psi(z(t), a(t))$ автомата Мили реализуется с помощью операторов case/if в первом блоке always, а описание выходных функций $w(t) = \psi(a(t))$ автомата Мура ($w(t) = \psi(a(t))$) – с помощью только оператора case. Стиль 2 предлагает использовать для реализации выходных функций условный оператор assign, который при синтезе реализуется комбинационной схемой. Регистр состояния RG описывается вторым блоком always. Для установки автомата в начальное состояние используется асинхронный управляющий сигнал сброса reset. Для дополнительного управления автоматом вводится сигнал разрешения синхронизации enable.

Листинг 2. Шаблон Verilog-модели автомата

```
module FSM1_synplify (clk, reset, enable, Управляющие входы Z_i, Выходы W_i);
    input clk, reset, enable;
    input Управляющие входы Z_i;
    output Выходы W_i;
/* Определение метки состояний, m – разрядность регистра состояний */
    parameter deflt = m'bxxx; /* Состояние по умолчанию */
    parameter состояние1 = m'b ...
    . . .
    reg Выходы W_i;
    reg [m:0] state, next_state;

/* Блок always для комбинационной схемы */
always @(state or enable or data_in) begin
/* Значения выходов по умолчанию*/
    W_i <= 1'b0;
    case (state)
        Состояние1 :
            if (Условие)
                begin NextState <= состояние i;
    /* Присвоение значений выходам для автомата Мили, стиль */
                end
             else
                begin NextState <= состояние i;
    /* Присвоение значений выходам для автомата Мили, стиль */
                end
```

```
    /* присвоение значений выходам для автомата Мура, стиль 1 */
        Состояние2 :
        . . .
        default : next_state <= deflt;
/* Присвоение значений выходам для состояния по умолчанию */
 /* Установка автомата в начальное состояние */
    endcase
end
/* Блок always для реализации регистра
состояний */
always @(posedge clk or negedge rst)
    if (!rst) state <= idle;
    else state <= next_state;
/* Условные параллельные операторы
назначения, стиль 2 */
assign W_i = условие ? 1'b1: 1'b0;
    . . .
endmodule
```

8. Системные иерархические модели автоматов

Современные устройства цифровой обработки сигналов представляют собой конвейер с управляющим автоматом (рис. 9), который подсчитывает число обработанных элементов и формирует управляющие сигналы в исключительных ситуациях: первый или последний элемент, первая или последняя строка изображения. Поскольку входные последовательности таких автоматов содержат большое количество элементов, то для их проектирования нельзя использовать классические методы синтеза структурных автоматов Мили или Мура, ввиду большого количества (100-1000) состояний. Использование комбинации {управляющий автомат Мили (Мура) – счетчик} не обеспечивает высокого быстродействия DSP-преобразователя. Как правило, управляющий блок для конвейерных вычислительных устройств строится на счетчиках, которые идентифицируют состояние управляющего автомата. Все выходные инициирующие сигналы генерируются на основе значений этих счетчиков.

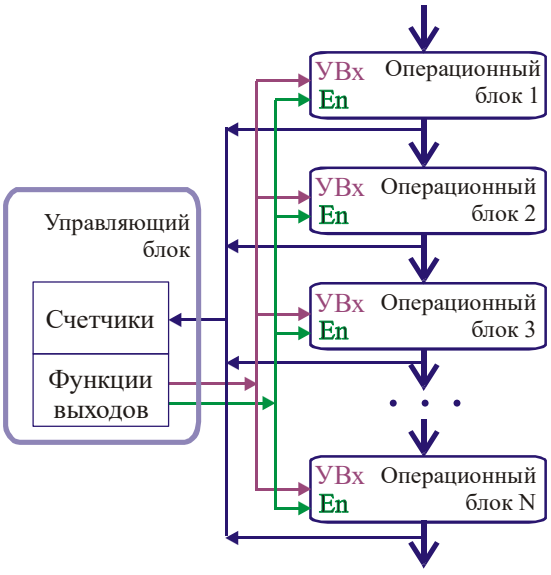


Рис. 9. Конвейерная структура DSP с макроавтоматом

Специфика управляющего модуля конвейерной архитектуры DSP-преобразователя , требует, чтобы предложенная в [2] классификация моделей автоматов, изображенных на рис. 3. была дополнена еще одним классом моделей управляющего автомата, который специализируется на конвейерный тип вычислений для DSP (рис. 10). Такая модель инте-

ресна тем, что она не имеет управляющих входных сигналов, за исключением входа синхронизации (Clk) и сброса автомата в начальное состояние (reset), и по сути является моделью автомата Мура, описываемой уравнениями:

$$a(t+1) = \varphi(a(t)); \quad w(t) = \psi(a(t)), t = 0,1,2,\ldots .$$

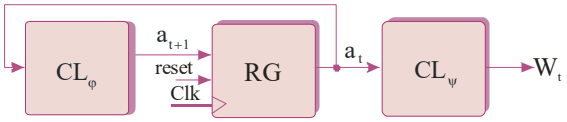


Рис. 10. Модель управляющего автомата конвейерного устройства

Блоки CL_φ и RG реализуют счетчик, заменяющий автомату таблицу переходов, а блок CL_ψ – выходные функции $w(t) = \psi(a(t))$.

Размерность схемы автомата обусловлена длиной регистра RG, а следовательно, рабочим диапазоном счетчика, который, в свою очередь, зависит от структуры входной информации. Таким образом, размерность управляющего блока зависит от двух параметров: R– разрядности счетчика и N_W – количества формируемых выходных функций. Быстродействие такого устройства, как следует из формул (1) и (2), определяется самым длинным путем схемы – структурной глубиной. В данной модели: 1) логический путь "Pin to Pin" начинается от асинхронного сброса reset и заканчивается выходами W: $T_{PP} = T_{RG\_reset} + T_{CL_\psi max}$; 2) путь "Register to Register" определяет задержку логического пути между регистрами счетчика: $T_{RR} = T_{RG} + T_{CL_\varphi max} + T_{su}$; 3) путь "Clock to Output": $T_{CO} = T_{RG} + T_{CL_\psi max}$.

Минимальный рабочий период автомата определяется следующим выражением:

$$P_{min} = \max(T_{CO}, T_{RR}, T_{PP}) =$$
$$= \max[(T_{RG} + T_{CL_\psi max}), (T_{RG} + T_{CL_\varphi max} + T_{su}),$$
$$(T_{RG\_reset} + T_{iCL_\psi})].$$

Как видно из формулы, минимальный рабочий период, а значит и максимальная рабочая частота, зависят от аппаратной сложности схем CL_φ и CL_ψ. Часто для повышения скорости работы устройства выходные функции W дополняются выходным регистром RG_W (рис. 11). Это позволяет сделать работу устройства более устойчивой к состязаниям и, благодаря делению логического пути, повысить максимальную рабочую частоту всего устройства, включая и управляющий автомат. При этом изменяются временные параметры системы. Такая модель не будет иметь путей "Clock to Output" и "Pin to Pin". Минимальный рабочий период устройства зависит только от самого длинного пути "Register to Register" и описывается формулой:

$$P_{min} = \max(T_{RR}) = \max[(T_{RG} + T_{CL_\psi max} + T_{su}),$$
$$T_{RG} + T_{CL_\varphi max} + T_{su})] =$$
$$= T_{RG} + T_{su} + \max(T_{CL_\psi max}, T_{CL_\varphi max}).$$

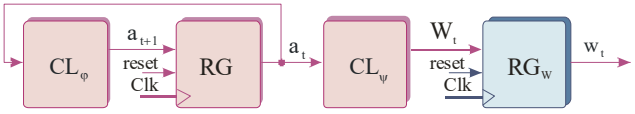


Рис. 11. Модель автомата с регистровыми выходами

При обработке двумерных данных, изображений, управляющий блок должен иметь два счетчика: 1) Сч 1 – для подсчета столбцов матрицы; 2) Сч 2 – для подсчета столбцов матрицы, если данные обрабатываются по строкам. Иначе – назначение счетчиков изменяется. Модель такого управляющего блока может быть представлена в виде сети автоматов (рис. 12).

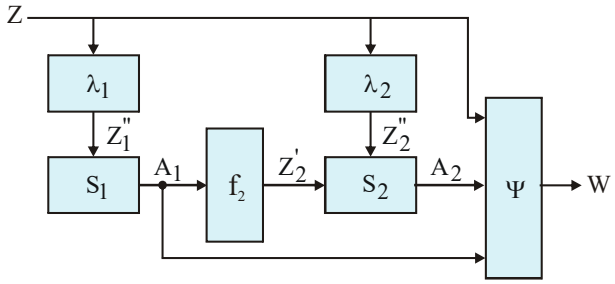


Рис. 12. Сеть n-компонентных автоматов

Сеть, содержащая два автомата, задается вектором:

$$N = (Z, \{S_1, S_2\}, W, \{f_1, f_2\}, \{\lambda_1, \lambda_2\}, \Psi).$$

Здесь параметры сети представлены в виде: 1) Входной алфавит Z={reset}. 2) Множество компонентных автоматов сети $S_i = \{S_1, S_2\}$:КА$_1$: $S_1 = (A_1, Z_1, \varphi_1)$,КА$_1$: $S_2 = (A_2, Z_2, \varphi_2)$; 3) Функции переходов для данных автоматов: $S_1\,\varphi_1 : A_1 \times Z_1 \to A_1$ $\delta_1 : A_1 \times \{\text{reset}\} \to A_1$; $S_2\,\varphi_2 : A_2 \times Z_2 \to A_2$ $\delta_2 : A_2 \times \{\text{rst}, \text{en}\} \to A_2$. 4) W – выходной алфавит сети. 5) Множество функций соединения компонентных автоматов сети представлено одной функцией: $f_1 = \varnothing$, $f_2 : A_1 \to Z_2^{'}$. 6) Множество входных функций : $\lambda_1 : Z_1 = Z_1^{''} = \{\text{reset}\}$; многокомпонентных автоматов сети $\lambda_2 : Z_2 = Z_2^{'} \times Z_2^{''} = \{\text{rst}, \text{en}\} \times \{\text{reset}\}$; 6) Выходная функция сети; $\psi : (A_1 \times A_2) \times Z \to W$.

Структурная модель сети, содержащей два автомата, представлена на рис. 13.

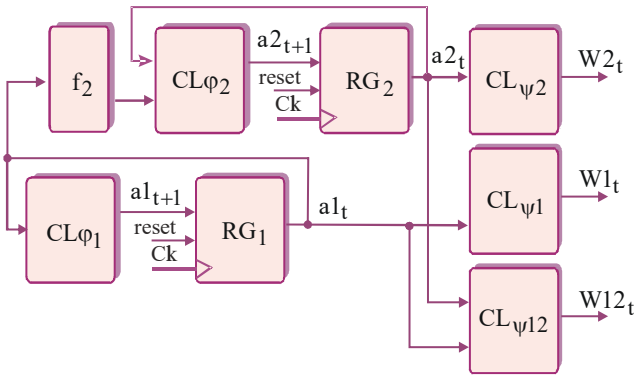


Рис. 13. Структурная модель сети автоматов

Блоки $CL\varphi_1$ и RG_1 реализуют первый счетчик Сч1, блоки $CL\varphi_2$ и RG_2 – второй счетчик Сч2. Блок f_2 является комбинационным и реализует функцию соединения двух компонентных автоматов Сч1 и Сч2. Совокупный автомат содержит три вида выходных функций: ψ_1 – зависит от состояния автомата Сч1; ψ_2 – от состояния автомата Сч2; ψ_{12} – от состояний автоматов Сч1 и Сч2:

$$w_1(t) = \psi_1(a1(t)); \; w_2(t) = \psi_2(a2(t)); \; w_{12}(t) = \psi_{12}(a1(t), a2(t)).$$

Функциям ψ_1, ψ_2 и ψ_{12} соответствуют комбинационные блоки $CL\psi_1, CL\psi_2$ и $CL\psi_{12}$.

Также как и для автомата с одним счетчиком, минимальный рабочий период будет равен максимальному пути:

$$P_{min} = \max(T_{CO}, T_{RR}, T_{PP}).$$

1) Комбинационный (логический) путь "Pin to Pin" проходит от асинхронного сброса reset до выходов W:

$$T_{PP} = \max[(T_{RG1\_reset} + T_{CL_{\psi 1}\max}), (T_{RG2\_reset} + T_{CL_{\psi 1}\max}), (T_{RG\_reset_{max}} + T_{CL_{\psi 12}\max})].$$

2) Путь "Register to Register" определяет задержку комбинационного пути между регистрами счетчика:

$$T_{RR} = \max[(T_{RG1} + T_{CL_{\varphi 1 \max}} + T_{su}), (T_{RG2} + T_{CL_{\varphi 2 \max}} + T_{su}), (T_{RG1} + T_{f2_{max}} + T_{su})].$$

3) Путь "Clock to Output" определяется компонентами:

$$T_{CO} = \max[(T_{RG1} + T_{CL_{\psi 1}\max}), (T_{RG2} + T_{CL_{\psi 2}\max}), (T_{RG_{max}} + T_{CL_{\psi 12}\max})].$$

При реализации устройства на микросхемах программируемой логики PLD или ASIC все триггеры однотипны и поэтому имеют одинаковое время установки Tsu и задержки переключения T_{RG}. Тогда представленные выше формулы трансформируются к виду:

$$T_{PP} = \max[(T_{RG1\_reset} + T_{CL_{\psi 1}\max}), (T_{RG2\_reset} + T_{CL_{\psi 1}\max}), (T_{RG\_reset_{max}} + T_{CL_{\psi 12}\max})] =$$

$$= T_{RG\_reset} + \max(T_{CL_{\psi 1}\max}, T_{CL_{\psi 2}\max}, T_{CL_{\psi 12}\max}).$$

$$T_{CO} = T_{RG} + \max(T_{CL_{\psi 1}\max}, T_{CL_{\psi 2}\max}, T_{CL_{\psi 12}\max}).$$

Если триггеры регистров RG1 и RG2 переключаются по одному фронту, то имеет место формула:

$$T_{RR} = T_{RG} + T_{su} + \max(T_{CL_{\varphi 1 \max}}, T_{CL_{\varphi 2 \max}}, T_{f2_{max}}).$$

Если триггеры регистров RG1 и RG2 переключаются по разным фронтам, тогда действительно выражение:

$$T_{RR} = T_{su} + \max[(T_{RG} + T_{CL_{\varphi 1 \max}}), (T_{RG} + T_{CL_{\varphi 2 \max}}), 2(T_{RG} + T_{f2_{max}})].$$

Для описания алгоритма функционирования предложенных системных иерархических моделей управления конвейерными вычислениями DSP-преобразования традиционные формы синтеза на основе графа или таблицы переходов не являются эффективными. Исходной информацией для системных иерархических моделей может быть структура, состоящая из следующих параметров:

Модель 1: 1) Cnt_{min} и Cnt_{max} минимальная и максимальная границы счета; 2) Cnt_{S0} – начальное состояние; 3) N_F – количество выходных функций; 4) тип выходов: комбинацион-

ный или регистровый; 5)множество выходных функций $\Psi = \{\psi_1, \psi_2, ..., \psi_{N_F}\}$. Модель 2: 1) $Cnt1_{min}$ и $Cnt1_{max}$ – минимальная и максимальная границы счета счетчика 1,

$Cnt2_{min}$ и $Cnt2_{max}$ – минимальная и максимальная границы счета счетчика 2; 2) $Cnt1_{S0}$ и $Cnt2_{S0}$ – начальное состояние счетчиков 1 и 2; 3) N_F – количество выходных функций; 4) тип выходов: комбинационный или регистровый; 5) множество выходных функций $\Psi = \{\psi_1, \psi_2, ..., \psi_{N_F}\}$; 6) f2 – функции соединения компонентных автоматов сети.

На рис. 14 схематично представлен способ создания HDL-кода для модели 1. Блоки CL_φ и RG, соответствующие счетчику, могут быть реализованы одним процессом для VHDL-модели или одним блоком always для Verilog. Для реализации выходных функций используются параллельные условные операторы (VHDL) или условный оператор assign (Verilog). Шаблоны VHDL- и Verilog-кода для модели 1 представлены листингами 3 и 4 соответственно.

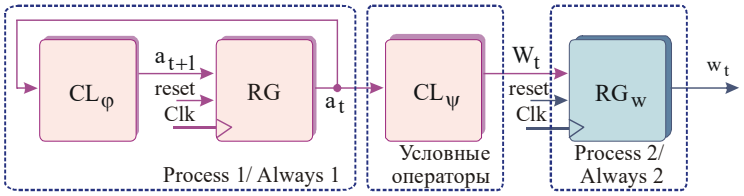


Рис. 14. Модель управляющего автомата с регистровыми выходами

Листинг 3. VHDL-шаблон для реализации модели 1

```
library IEEE;
use  IEEE.std_logic_1164.all;
-- Пакеты, поддерживающие арифметические операции над данными std_logic_vector
use  ieee.std_logic_unsigned.all;
entity Имя_интерфейса is
 port (En, Clk, reset: in STD_LOGIC;
          Выходы Wi или W_i_reg: out STD_LOGIC);
end entity Имя_интерфейса;
architecture Имя_арх of Имя_интерфейса is
     -- Разрядность счетчика
constant m: natural:= ...;
     -- Нижняя граница счетчика
constant Cnt_min: std_logic_vector(m-1 downto 0):=...;
         -- Верхняя граница счета
constant Cnt_max: std_logic_vector(m-1 downto 0):=...;
         -- Состояние сбросв
constant Cnt_S0: std_logic_vector(m-1 downto 0):=...;
     signal count1: std_logic_vector(m-1 downto 0);
     --Определение сигналов Wi,
     -- Если выходы устройства регистровые
     signal Wi : std_logic;
begin
-- Модель счетчика
Process_1 : process(reset, En, clk)
          begin
               if reset='1' then
     -- Сброс счетчика в начальное состояние
                    count1 <= Cnt_S0;
     -- Описание проверки фронта синхросигнала
               elsif clk='1'and clk'event then
-- Или задний фронт elsif clk='0'and clk'event then
                    if en='1' then
     -- Проверка верхней границы счета
                         if count1 = Cnt_max then
     -- Присвоение нижней границы счета
```

```
                    count1 <= Cnt_min;
                else
                    count1 <= count1 + '1';
                end if;
            end if;
        end if;
    end process;
-- Реализация выходных функций с помощью
-- условных параллельных операторов
W_i <= '1' when условие else
    '0';
. . .
-- Процесс для реализации регистровых выходов
Process_2: process(reset, clk)
    begin
        if reset='1' then
    -- Сброс в нулевое состояние
            W_i_reg <= (others =>'0');
            . . .
-- Описание проверки фронта синхросигнала
        elsif clk='1'and clk'event then
-- Или задний фронт elsif clk='0'and clk'event then
            W_i_reg <= W_i;
            . . .
        end if;
    end process;
end Имя_арх;
```

Листинг 4. Verilog-шаблон для реализации модели 1

```
module Имя_интерфейса (En, Clk, reset, Выходы Wi или W_i_reg)
    input En, Clk, reset;
    output Выходы Wi;
    // Если выходы комбинационные
    // или, если выходы регистровые
    //   output Выходы W_i_reg;    reg Выходы W_i_reg; wire Выходы Wi;
        // Параметры конфигурации устройства
    parameter m = ...;          // разрядность счетчика
    parameter Cnt_min =...; // нижняя граница
    parameter Cnt_max =...; // верхняя граница
    parameter Cnt_S0 =...;   // состояние сбросов
        reg[m-1:0]    count1;
begin
// Являющийся моделью счетчика блок always 1
always @(posedge reset or posedge En or posedge clk)
    if (reset)
    // Сброс счетчика в начальное состояние
        count1 <= Cnt_S0;
    else if (en)
    // Проверка верхней границы счета
            if (count1 == Cnt_max)
    // Присвоение нижней границы счета
                count1 <= Cnt_min;
            else
                count1 <= count1 + 1'b1;
// Реализация выходных функций с помощью условных операторов
assign W_i = условие ? 1'b1: 1'b0;
. . .
// Блок always 2, для регистровых выходов
always @(posedge CLk or negedge Reset)
    begin
        if (Reset) W_i_reg <= 'b0;
```

```
        else W_i_reg <= W_i;
        . . .
    end
endmodule
```

Принцип построения HDL-кода для иерархической модели 2 представлен на рис. 15.

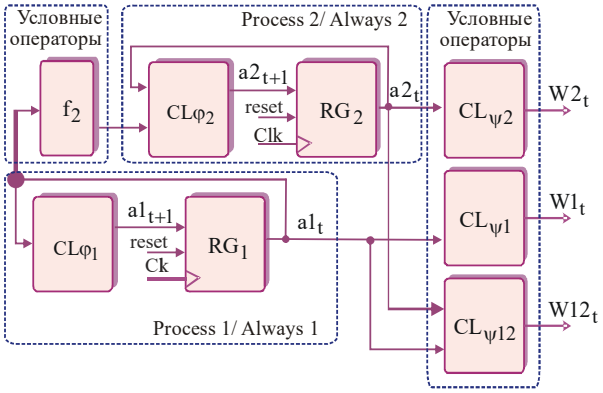


Рис. 15. Структурная модель сети автоматов

Каждому счетчику, реализуемому блоками CL_φ и RG, ставится в соответствие процесс для VHDL-модели или блок always для Verilog. Таким образом, счетчик 1 кодируется оператотом Process 1 или Always 1, счетчик 2 – оператором Process 2 или Always 2. Для реализации выходных функций используются параллельные условные операторы (VHDL) или операторы assign (Verilog). Условные операторы также применяются для реализации функций соединения компонентных автоматов сети f_2. Шаблоны VHDL- и Verilog-кода для модели 1 представлены листингами 5 и 6 соответственно. Условия в операторах для функций $W1_t$ зависят только от состояния счетчика 1, а для функций $W2_t$ – от состояния счетчика 2, условия функций $W2_t$ – от значений обоих счетчиков. Для реализации регистровых выходных функций HDL-модель может быть дополнена соответственным оператором Process или Always, как это сделано в модели 1.

Листинг 5. VHDL-шаблон для реализации модели 2

```
library IEEE;
use  IEEE.std_logic_1164.all;
-- Пакеты, поддерживающие арифметические
-- операции над данными std_logic_vector
use  ieee.std_logic_unsigned.all;
entity Имя_интерфейса is
 port (En, Clk, reset: in STD_LOGIC;
          Выходы W1i, W2i, W12i: out STD_LOGIC);
end entity Имя_интерфейса;
architecture Имя_арх of Имя_интерфейса is
     -- Разрядность счетчика 1
constant m1: natural:= ...;
     -- Разрядность счетчика 2
constant m2: natural:= ...;
     -- Нижняя граница счетчика
constant Cnt1_min: std_logic_vector(m-1 downto 0):=...;
constant Cnt2_min: std_logic_vector(m-1 downto 0):=...;          -- Верхняя граница счета
constant Cnt1_max: std_logic_vector(m-1 downto 0):=...;
constant Cnt2_max: std_logic_vector(m-1 downto 0):=...;
          -- Состояние сброса
constant Cnt1_S0: std_logic_vector(m-1 downto 0):=...;
constant Cnt2_S0: std_logic_vector(m-1 downto 0):=...;           signal  count1:
```

```
std_logic_vector(m1-1 downto 0);
    signal count2: std_logic_vector(m2-1 downto 0);
    -- Сигналы, формируемые функцией f2
    signal en2, rst2: std_logic;
begin
-- Процесс счетчика 1
Process_1 : process(reset, En, clk)
    begin
        if reset='1' then
        -- Сброс счетчика в начальное состояние
            count1 <= Cnt1_S0;
    -- Описание проверки фронта синхросигнала
        elsif clk='1'and clk'event then
-- или задний фронт elsif clk='0'and clk'event then
                if en='1' then
    -- Проверка верхней границы счета
                    if count1 = Cnt1_max then
    -- Присвоение нижней границы счета
                        count1 <= Cnt1_min;
                    else
                        count1 <= count1 + '1';
                    end if;
                end if;
            end if;
        end  process;
-- Процесс счетчика 2
Process_2 : process(reset, En, clk, rst2, en2)
    begin
        if (reset='1')or(rst2='1') then
        -- Сброс счетчика в начальное состояние
            count2 <= Cnt_S0;
    -- Описание проверки фронта синхросигнала
            elsif clk='1'and clk'event then
 -- Или задний фронт elsif clk='0'and clk'event then
                if (en='1') and (en2='1') then
    -- Проверка верхней границы счета
                    if count2 = Cnt2_max then
    -- Присвоение нижней границы счета
                        count2 <= Cnt2_min;
                    else
                        count2 <= count2 + '1';
                    end if;
                end if;
            end if;
        end  process;
-- Функции соединения компонентных автоматов сети f2
    rst2 <= '1' when условие else
        '0';
    en2    <= '1' when условие else
        '0';
-- Реализация выходных функций условными
--  параллельными  операторами
    W1_i <= '1' when условие else
            '0';
    . . .
    W2_i <= '1' when условие else
        '0';
    . . .
    W12_i <= '1' when условие else
        '0';
    . . .
```

```
end Имя_арх;
```

Листинг 6. Verilog-шаблон для реализации модели 2

```
module Имя_интерфейса (En, Clk, reset, Выходы W1i, W2i, W12i)
    input En, Clk, reset;
    output Выходы W1i, W2i, W12i;
    // Параметры конфигурации счетчика 1
    parameter Cnt1_min =...;    // нижняя граница
    parameter Cnt1_max =...;    // верхняя граница
    parameter Cnt1_S0 =...;     // состояние сбросв
    parameter m1 = ...;       // разрядность счетчика
    reg[m1-1:0] count1;
        // Параметры конфигурации счетчика 1
    parameter Cnt2_min =...;    // нижняя граница
    parameter Cnt2_max =...;    // верхняя граница
    parameter Cnt2_S0 =...;     // состояние сбросов
    parameter m2 = ...;       // разрядность счетчика
    reg[m2-1:0] count2;
    // Сигналы, формируемые функцией f2
    wire en2, rst2;
// Счетчик 1, блок always 1
always @(posedge reset or posedge En or posedge Clk)
    if (reset)
        // Сброс счетчика в начальное состояние
        count1 <= Cnt1_S0;
    else if (En)
        // Проверка верхней границы счета
            if (count1 == Cnt1_max)
                count1 <= Cnt1_min;
            else
                // Присвоение нижней границы счета
                count1 <= count1 + 1'b1;
    // Счетчик 2, блок always 2
always @(posedge reset or posedge En or posedge Clk or posedge en2 or posedge rst2)
    if (reset||rst2)
        // Сброс счетчика в начальное состояние
        count2 <= Cnt2_S0;
    else if (En)
        // Проверка верхней границы счета
            if (count1 == Cnt2_max)
                count1 <= Cnt2_min;
            else
        // Присвоение нижней границы счета
                count2 <= count2 + 1'b1;
 // Функции соединения компонентных автоматов сети f2
    assign rst2 = условие ? 1'b1: 1'b0;
    assign  en2   = условие ? 1'b1: 1'b0;
// Реализация выхондых функций с помощью
// условных операторов
    assign W1_i = условие ? 1'b1: 1'b0;
    . . .
    assign W2_i = условие ? 1'b1: 1'b0;
    . . .
    assign W12_i = условие ? 1'b1: 1'b0;
        . . .
endmodule
```

9. Синтез HDL-модели автомата для DSP-фильтра

Управляющий блок фильтра для обработки изображений размерностью 256×256 пикселов имеет следующие параметры: 1) минимальная и максимальная границы счета счетчика 1: $Cnt1_{min}$=1 и $Cnt1_{max}$ =256; Минимальная и максимальная границы счета счетчика 2: $Cnt2_{min}$ =1 и $Cnt2_{max}$ =258. 2) Начальное состояние счетчиков 1 и 2: $Cnt1_{S0} = 0$ и $Cnt2_{S0} = 0$. 3) Количество выходных функций N_F =3. 4) Тип выходов – комбинационный. 5) Множество выходных функций:

$$\Psi = \{\psi_1, \psi_2, \psi_3\} = \{\text{First}, \text{Last}, \text{Enable}\};$$

$$\text{First} = \begin{cases} 1, & \text{count2} = 2; \\ 0, & \text{else.} \end{cases} \quad \text{Last} = \begin{cases} 1, & \text{count2} = \text{Cnt2}_{\min} - 1; \\ 0, & \text{else.} \end{cases}$$

$$\text{Enable} = \begin{cases} 1 & (\text{count2} > 1) \,\&\, (\text{count2} <= \text{Cnt2}_{\max} - 1); \\ 1 & (\text{count2} = \text{Cnt2}_{\max}) \text{ и } (\text{count1} = 1); \\ 0 & \text{else.} \end{cases}$$

$$W1_i = \varnothing, \; W2_i = \{\text{First}, \text{Last}\}, \; W12_i = \{\text{Enable}\}.$$

Здесь count1 – счетчик 1, count2 – счетчик 2. 6) Функции соединения компонентных автоматов сети f2: $f_2 = \{en2\}$.

VHDL и Verilog-модели управляющего блока представлены листингами 7 и 8.

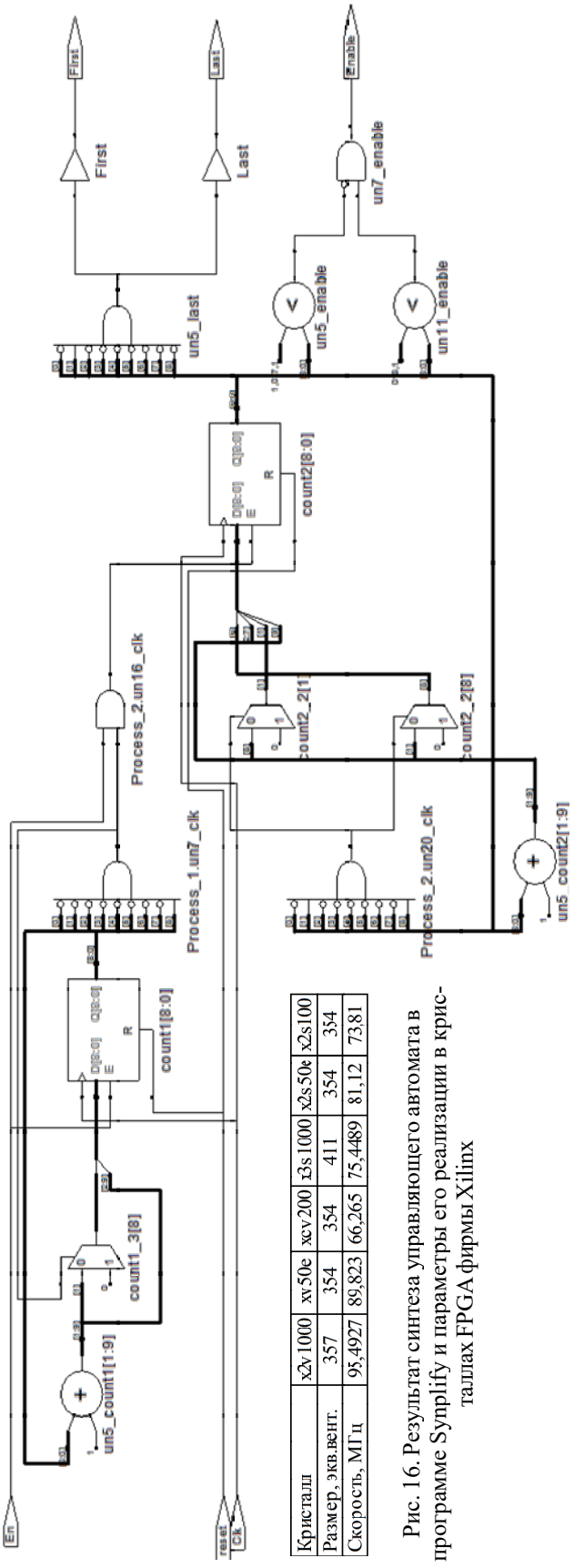
Листинг 7. VHDL-модель управляющего блока

```
library IEEE;
use  IEEE.std_logic_1164.all;
use  ieee.std_logic_unsigned.all;
use  ieee.std_logic_arith.all;
entity Control1 is
 port (En, Clk, reset: in STD_LOGIC;
          First, Last, Enable: out STD_LOGIC);
end entity Control1;
architecture Control1 of Control1 is
-- разрядность счетчика 1
constant m1: natural:= 9;
-- разрядность счетчика 2
constant m2: natural:= 9;
     -- нижняя граница счетчика
constant Cnt1_min: std_logic_vector(m1-1 downto 0):=conv_std_logic_vector(1, m1);
constant Cnt2_min: std_logic_vector(m2-1 downto 0):=conv_std_logic_vector(1, m2);
     -- верхняя граница счета
constant Cnt1_max: std_logic_vector(m1-1 downto 0):=conv_std_logic_vector(256, m1);
constant Cnt2_max: std_logic_vector(m2-1 downto 0):=conv_std_logic_vector(258, m2);
     -- состояние сбросв
constant Cnt1_S0: std_logic_vector(m1-1 downto 0):=conv_std_logic_vector(0, m1);
constant Cnt2_S0: std_logic_vector(m2-1 downto 0):=conv_std_logic_vector(0, m2);
signal count1: std_logic_vector(m1-1 downto 0);
signal count2: std_logic_vector(m2-1 downto 0);
     -- сигналы, формируемые функцией f2
signal en2: std_logic;
begin
-- Процесс счетчика 1
Process_1 : process(reset, En, clk)
          begin
               if reset='1' then
     -- сброс счетчика в начальное состояние
                    count1 <= Cnt1_S0;
     -- описание проверки фронта синхросигнала
               elsif clk='1'and clk'event then
-- или задний фронт elsif clk='0'and clk'event then
                    if en='1' then
```

```
        -- проверка верхней границы счета
                    if count1 = Cnt1_max then
                        count1 <= Cnt1_min;
                    else
        -- присвоение нижней границы
счета
                        count1 <= count1 + '1';
                    end if;
                end if;
            end if;
        end process;
-- Процесс счетчика 2
Process_2 : process(reset, En, clk, en2)
        begin
            if (reset='1') then
    -- сброс счетчика в начальное состоя-
ние
                count2 <= Cnt2_S0;
    -- описание проверки фронта синхро-
сигнала
            elsif clk='1'and clk'event then
-- или задний фронт elsif clk='0'and
clk'event then
                if (en='1') and (en2='1') then
                -- проверка верхней границы
счета
                    if count2 = Cnt2_max then
                -- присвоение нижней
границы счета
                        count2 <= Cnt2_min;
                    else
                        count2 <= count2 + '1';
                    end if;
                end if;
            end if;
        end process;
-- Функции соединения компонентных
автоматов сети f2
    en2    <= '1' when count1 = Cnt1_max
else
        '0';
-- реализация выходных функций условны-
ми
--  параллельными  операторами
    First <= '1' when count2=2 else '0';
    Last <= '1' when count2=Cnt2_min+'1'
else '0';
    Enable <= '1' when ((count2>1) and
(count2<=Cnt2_max-1) ) or ((count2 =
Cnt1_max) and (count1=1)) else   '0';
end Control1;
```

Листинг 8. Verilog-модель управляющего блока

```
module Control1 (En, Clk, reset, First, Last,
Enable);
    input En, Clk, reset;
    output First, Last, Enable;
    // параметры конфигурации счетчика 1
    parameter Cnt1_min = 'd1;  // нижняя
```



Кристалл	x2v1000	xv50e	xcv200	i3s1000	x2s50e	x2s100
Размер, экв.вент.	357	354	354	411	354	354
Скорость, МГц	95,4927	89,823	66,265	75,4489	81,12	73,81

Рис. 16. Результат синтеза управляющего автомата в программе Synplify и параметры его реализации в кристаллах FPGA фирмы Xilinx

```
граница
    parameter Cnt1_max = 'd256;// верхняя граница
    parameter Cnt1_S0 = 'd0;   // состояние сбросв
    parameter m1 = 8;          // разрядность счетчика
    reg[8:0] count1;
    // параметры конфигурации счетчика 1
    parameter Cnt2_min = 'd1;  // нижняя граница
    parameter Cnt2_max = 'd258;// верхняя граница
    parameter Cnt2_S0 = 'd0;   // состояние сбросов
    parameter m2 = 8;    // разрядность счетчика
    reg[8:0] count2;
    // сигналы, формируемые функцией f2
    wire en2;
    //wire en_in;//, rst_in;
// Счетчик 1, блок always 1
always @(posedge reset or posedge Clk)
if (reset)    // сброс счетчика в начальное состояние
        count1 <= Cnt1_S0;
else
    begin
        if (En)
        // проверка верхней границы счета
        begin
            if (count1 == Cnt1_max)
                count1 <= Cnt1_min;
            else
            // присвоение нижней границы счета
                count1 <= count1 + 1'b1;
     end
    end
// Счетчик 2, блок always 2
always @(posedge reset or posedge Clk)
    if (reset)
        // сброс счетчика в начальное состояние
        count2 <= Cnt2_S0;
    else
    begin
        if (en2)
        // проверка верхней границы счета
        begin
            if (count2 == Cnt2_max)
                count2 <= Cnt2_min;
            else
                // присвоение нижней границы счета
                count2 <= count2 + 1'b1;
        end
    end
// Функции соединения компонентных автоматов сети f2
assign en2= (En &&(count1 == Cnt1_max))?1'b1: 1'b0;
// реализация выходных функций с помощью
// условных операторов
assign First = (count2 == 2) ? 1'b1: 1'b0;
assign Last = (count2 == Cnt2_min+1) ? 1'b1: 1'b0;
assign Enable = ((count2>1)&&(count2<=Cnt2_max-1))|| ((count2==Cnt2_max)&&(count1==1))?
1'b1: 1'b0;
endmodule
```

Структурная схема управляющего автомата для конвейерного DSP-преобразователя, сгенерированная программой синтеза Synplify, представлена на рис. 16. При синтезе использовались микросхемы программируемой логики FPGA фирмы Xilinx [12].

10. Генератор HDL-кода моделей автоматов

На основе предложенной системной иерархической модели управляющего автомата разработана программа автоматической генерации VHDL- и Verilog-кодов блока управления для вычислителя с конвейерной архитектурой. При создании программного продукта использовался скриптовый язык пакета Matlab [13].

Интерфейс программы генерирования управляющих автоматов для конвейрных архитектур представлен на рис. 17. Выбор языка для генерирования модели и ее тип определяются входными параметрами, реализованными с помощью переключателей: HDL selection – выбор языка для генерирования кода; Number of Counters – модель с одним или двумя счетчиками. Для каждого счетчика задается нижняя Min и верхняя Max границы счета, значение параметра инициализации Init. Для модели с одним счетчиком параметры панели Counter 2 игнорируются. Панель F2 function предназначена для создания функций соединения компонентных автоматов. Программа позволяет задавать две функции En2 и Rst2. Первая подключается к входу разрешения синхронизации, а вторая – к сбросу устройства в начальное состояние. Для того чтобы задать выходные функции, необходимо указать их количество. Затем для каждой функции следует ввести ее имя (Name) и условие формирования единичного значения (Conditions of 1 value). Условия должны быть записаны с использование синтаксиса HDL-языка, который выбран для генерирования модели. Текущее значение счетчика 1 обозначается переменной u1, счетчика 2 – u2.

Для разработки приложения использовался инструмент GUIDE(MATLAB® Graphical User Interface development environment) программного пакета MATLAB. Полный код программы генерирования содержит 1110 строк М-скрипта. Генератор работает в программной среде MATLAB, операционная система Windows.

11. Выводы

Предложены две модифицированные модели управляющих автоматов, предназначенные для использования в устройствах с конвейерной архитектурой. Приведено их математическое и структурное описание. Разработаны HDL-шаблоны реализации разработанных моделей управления для DSP-преобразователей.

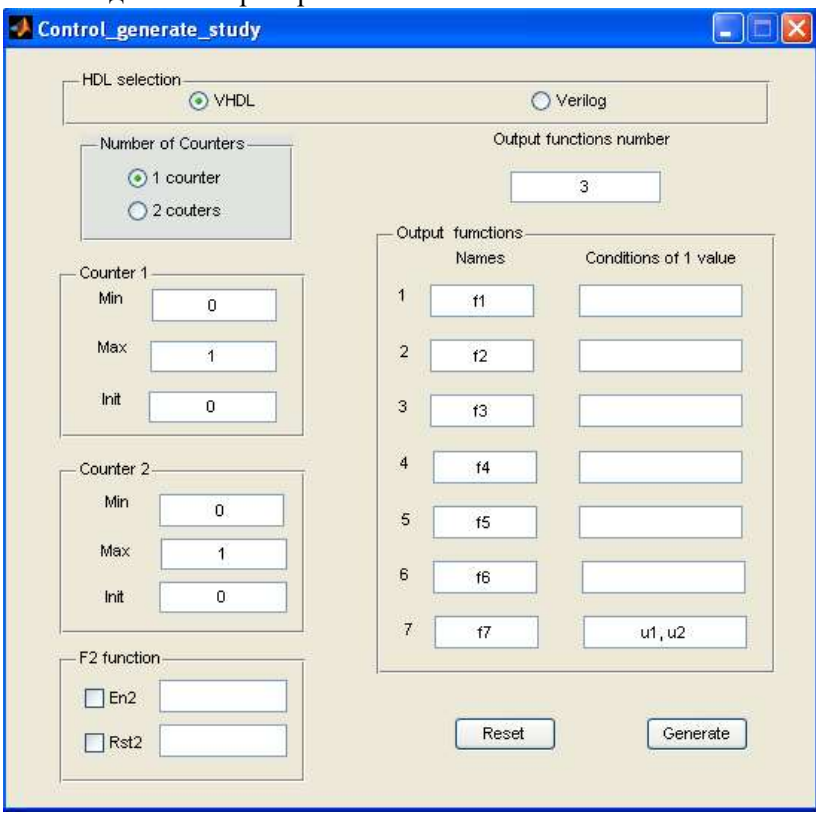


Рис. 17. Интерфейс программы генерирования HDL-модели

Созданные модели предназначены для использования в DSP-устройствах с конвейерной архитектурой. В отличие от существующих классических моделей предложенные структуры не используют таблицы переходов для исходного описания поведения автомата.

Предложена технология системного проектирования, которая позволяет автоматизировать процесс синтеза цифровых устройств на кристаллах с конвейерной архитектурой. Создан программный продукт, который выполняет автоматический синтез VHDL и Verilog-кода предложенных моделей управляющих автоматов. Генераторы моделей автоматов упрощают процесс проектирования DSP-преобразователей, позволяют эффективно создавать и верифицировать системные HDL-модели управляющего блока.

Список литературы: 1. *Philip E. Ross.* 5 Commandments. IEEE Spectrum. December. 2003. P. 30-35. **2.** *Баранов С.И.* Синтез микропрограммных автоматов. Л.: Энергия, 1979. 232 с. **3.** *Соловьев В.В.* Проектирование цифровых автоматов на основе программируемой логики интегральных схем. М.: Горячая линия. Телеком. 2001. 636 с. **4.** *Solowjew W., Chyzy M.* Synteza automatow skonczonych na ukladah PAL . Electronika. XXXVII. 1996. No 10. P.23 - 27. **5.** *Solovjev V., Mazalewski J., Chyzy M.* Models of robotics control systems on programmable logic devices. Proc. of the 4th Int. Symposium on Methods and Models an Automation and Robotics (MMAR'97). August 1997. Miedzyzdroje. Poland. Vol. 3. P. 1019 – 1024. **6.** *Кеэваллик А.Э.* Теорема декомпозиции конечных автоматов. Автоматика и вычислит. техника. 1974. № 1. С. 77-81. **7.** *Hartmanis J., Sterns R.* Algebraic Structure Theory of Sequential Machines. New York, Prentice-Hall. 1966. 464 p. **8.** *Хаханов В.И., Хаханова И.В.* VHDL + Verilog = Синтез за минуты. Харьков: СМИТ. 2007. 264 с. **9.** *Семенец В.В., Хаханова И.В., Хаханов В.И.* Проектирование цифровых систем с использованием языка VHDL. Харьков: ХНУРЭ. 2003. 492 с. **10.** *Charles H. Roth, Jr.* Digital Systems Design UsingVHDL. Boston. PWS Publishing Company. 1998. 470 p.**11.** *Ashenden, Peter J.* The designer's guide to VHDL. San Francisco. Calis. California. Morgan Kaufmann Publishers, Inc. 1996. 688 p. **12.** Xilinx.com. **13.** www.mathworks.com

Поступила в редколлегию 23.08.2007

Хаханова Ирина Витальевна, докторантка кафедры АПВТ ХНУРЭ. Научные интересы: Проектирование цифровых систем на кристаллах. Увлечения: английский язык, музыка. Адрес: Украина, 61166, Харьков, пр. Ленина, 14, тел. 70-21-326. E-mail: hahanova@mail.ru