Современные средства автоматизации процесса проектирования строго самосинхронных схем

Алексей БУМАГИН, к. т. н.
Алексей ГОНДАРЬ
Михаил КУЛЯС
Александр РУТКЕВИЧ
Владимир СТЕШЕНКО, к. т. н.
Мехди ТАЙЛЕБ
Григорий ШИШКИН

В статье рассматриваются вопросы построения и работы строго самосинхронных схем, приводятся требования к системам автоматизации проектирования таких схем, описываются два подхода к процессу автоматизации разработки строго самосинхронных схем — с использованием специализированной среды разработки с собственным языком описания схем, а также при помощи специализированных сценариев и существующих САПР для синхронных схем.

Строго самосинхронные схемы (ССС) имеют ряд существенных преимуществ перед тактируемыми схемами. ССС характеризуются низким уровнем электромагнитного излучения, низкой потребляемой мощностью и более высокой скоростью работы по сравнению с синхронными аналогами; работоспособность ССС сохраняется даже при снижении напряжения питания до уровня порога переключения транзисторов [1]. При разработке ССС отпадает необходимость проектирования разветвленного тактового дерева (что представляет определенную сложность для разработчиков синхронных схем при большом числе вентилей в кристалле).

Построение ССС от традиционного синхронного подхода отличается отсутствием тактового сигнала. Временная синхронизация в таких схемах осуществляется при помощи цепи обратной связи, как показано на рис. 1.

Входные данные объединяют в один поток информационные биты и управляющие символы при помощи парафазного кодирования (рис. 2). При таком кодировании для передачи одного бита сигнала (*A*) используется два проводника (*A.f, A.t*); при этом комбинация сигналов на проводниках данных 01 соответствует логической «1» (DATA1); 10 — логическому «0» (DATA0); 00 — разделителю (спейсер, NULL) между двумя посылками; комбинация 11 не используется и говорит об ошибке в работе схемы. Разделитель выставляется после каждой посылки данных (DATA0 или DATA1).

Самосинхронные регистры работают следующим образом: если на вход подтверждения подан логический «0», то через регистр может проходить спейсер, если же за ним будут следовать данные, то они не пройдут на выход регистра. При подаче на вход подтверждения логической «1» работа регистра меняется на противоположную: он будет пропускать данные, а спейсер нет.

Блок индикации предназначен для определения момента окончания переходного процесса. Его выход переключается в состояние логической «1», если на всех его входах находится состояние спейсера, обратное переключение происходит, только если на всех его входах есть данные.

Принцип работы таких схем следующий [2]: входные данные в парафазном коде поступают во входной регистр и далее на комбинационную схему. По мере окончания вычислений в комбинационной схеме данные проходят выходной регистр и поступают на блок индикации, который отслеживает момент окончания всех переходных процессов. Когда все переходные процессы закончены, индикатор вырабатывает сигнал подтверждения для входного регистра, который разрешает передачу спейсера. При поступлении спейсера на вход схемы он проходит первый регистр, комбинационную схему и ожидает на выходном регистре прихода сигнала подтверждения от последующей схемы. После его прихода спейсер проходит выходной регистр, срабатывает индикатор, и входной регистр готов принимать данные. Таким образом, скорость работы ССС зависит исключительно от времени распространения сигналов в схеме, которое, в свою очередь, зависит от самих данных и внешних условий (температуры, напряжения).

Для автоматизации процесса проектирования ССС необходимо иметь САПР, которые позволяют легко интегрироваться в общий процесс проектирования СБИС. Основными

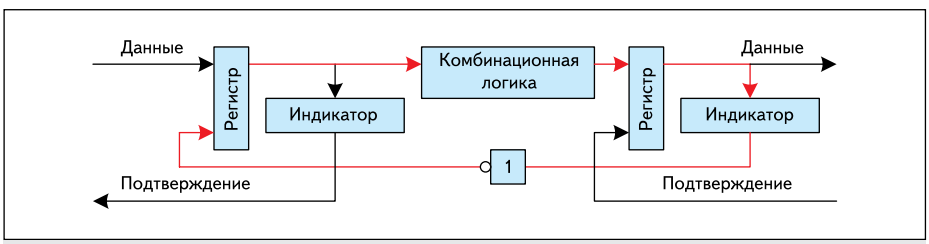

Рис. 1. Принцип построения ССС

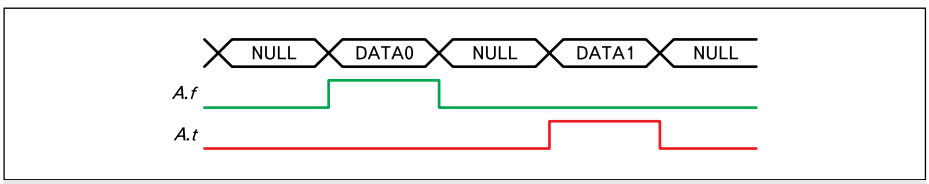

Рис. 2. Парафазное кодирование данных

требованиями к таким средствам автоматизации проектирования являются [3]:

- Возможность работать с проектами большого размера.
- Наличие широко развитых методик проектирования аналогично синхронным САПР.
- Возможность использования стандартных языков описания аппаратуры (HDL), таких как Verilog, VHDL и специализированных языков проектирования самосинхронных схем, совместно с современными средствами моделирования и верификации.
- Возможность использования стандартных средств для синтеза схем, размещения и трассировки соединений, а также программ временного анализа, контроля эквивалентности, экстракции паразитных параметров, автоматической генерации тестов и пр. на всех этапах разработки и производства микросхем.
- Возможность использования стандартной библиотеки КМОП-элементов фабрик, изготовителей микросхем, за исключением отдельных случаев, когда требуется максимальное быстродействие и низкое энергопотребление (тогда создается специализированная библиотека).

На данный момент существует два подхода к автоматизации проектирования ССС. В первом методе используют специализированные САПР со своими языками описания ССС. Примером таких программных средств может служить коммерческая среда разработки TiDE (tm) (the Timeless Design Environment) компании Handshake Solutions (www.handshakesolutions.com), в которой используется язык описания самосинхронных схем Haste, или свободно распространяемое средство синтеза самосинхронных схем Balsa [4], разработанное в университете Манчестера (www.manchester.ac.uk). Во втором подходе используют так называемую прямую трансляцию синхронных схем в ССС (SADT, Synchronous-Asynchronous Direct Translation) при помощи существующих САПР для синхронных схем и специализированных скриптов [5, 6]. Рассмотрим каждый из этих методов более подробно.

Использование современных САПР для строго самосинхронных схем

Компания Handshake Solutions создала собственную среду разработки строго самосинхронных схем TiDE с компилятором языка описания ССС Haste на основе проекта Tangram (Philips Research, 1986 год). TiDE позволяет проводить полный цикл разработки и оптимизации ССС, используя технологию Handshake Technology. Среда совместима со всеми стандартными САПР (для логической оптимизации, автоматической генерации тестовых векторов, размещения и разводки элементов, временной и формальной верификации, моделирования), а также есть возможность использования стандартных библиотечных элементов синхронных схем без разработки собственной библиотеки. TiDE поддерживает технологию создания тестопригодных схем — DFT (design for test) и создание прототипов на ПЛИС. Общий маршрут проектирования представлен на рис. 3.

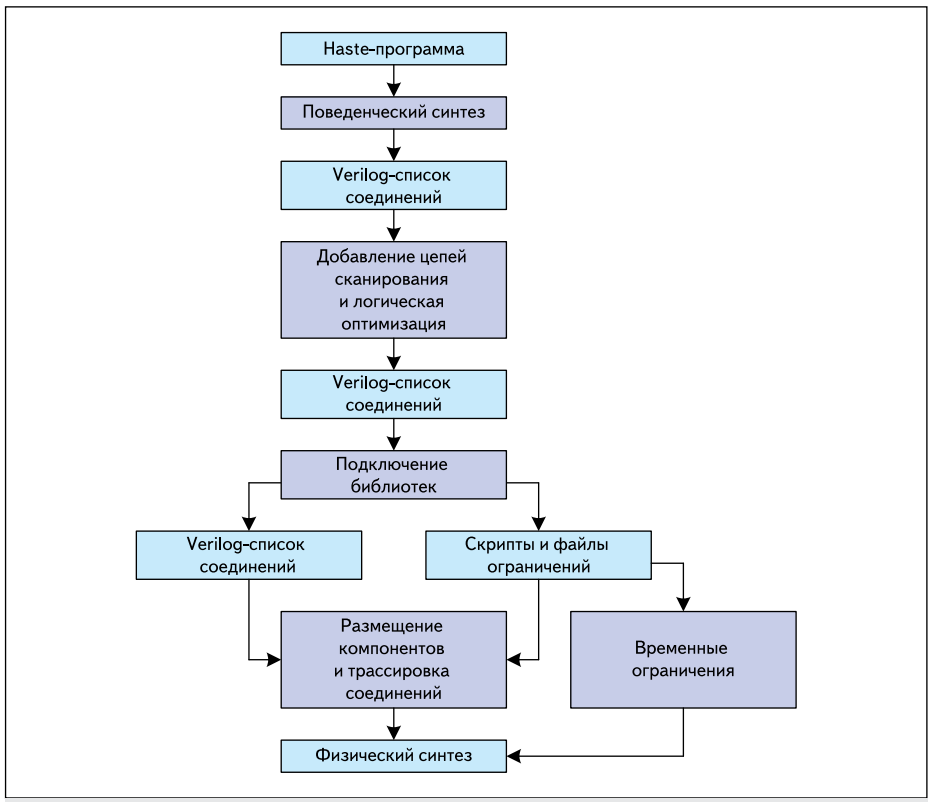

Рис. 3. Маршрут проектирования в САПР TiDE (tm)

Функциональное проектирование

Ввод проекта осуществляется при помощи языка описания ССС Haste, как показано на рис. 4. Далее программа на Haste компилируется и представляется в виде списка соединений (netlist) на языке Verilog. Это происходит в два этапа. Сначала htcomp

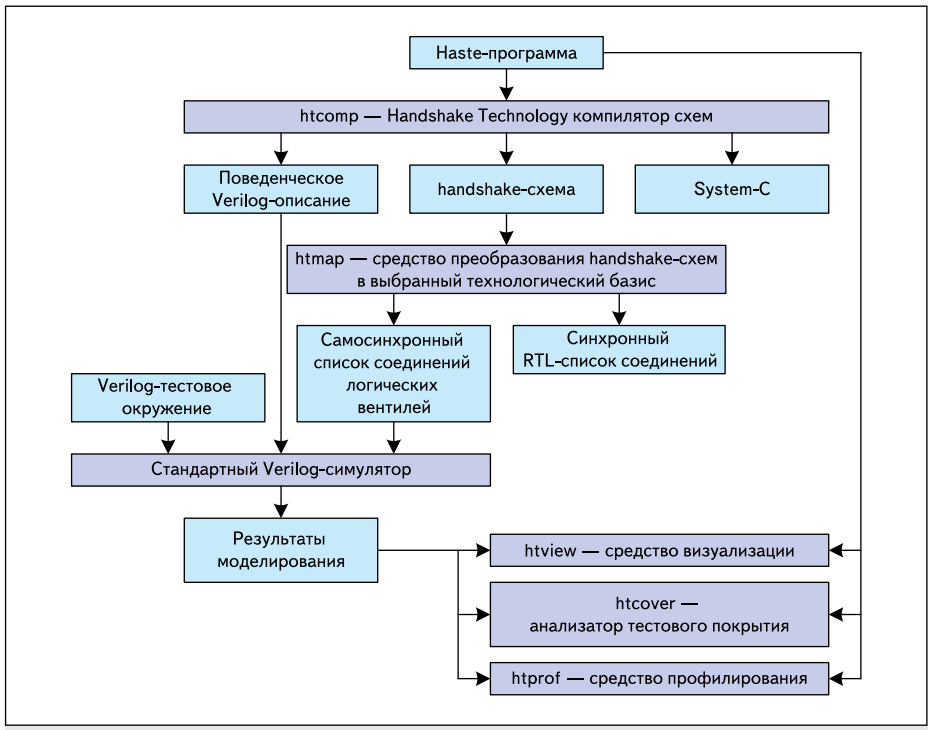

Рис. 4. Функциональное проектирование и моделирование

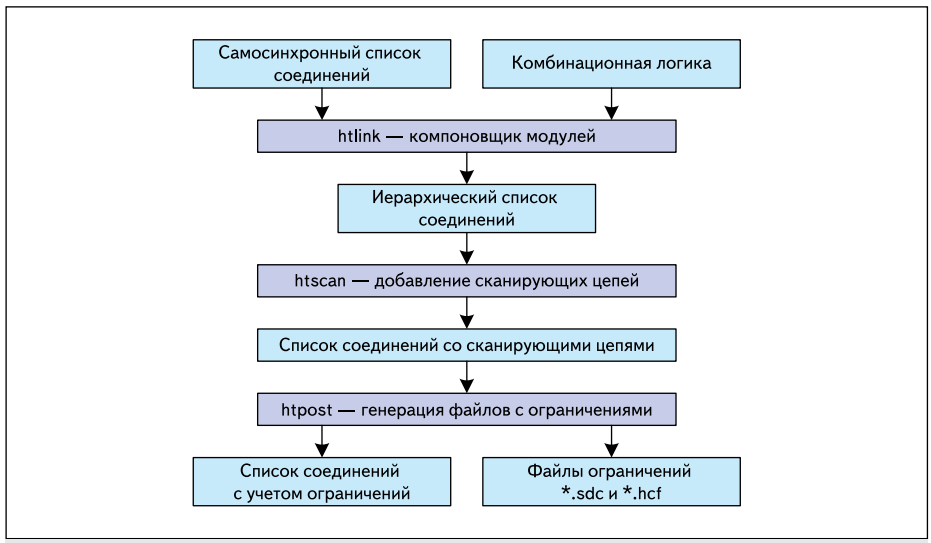

Рис. 5. Структурное проектирование

транслирует Haste-описание во внутренний формат описания CCC (handshake-схема), а затем при помощи htmap осуществляется преобразование в структурное описание схемы на языке Verilog, после чего возможна оптимизация проекта на начальном уровне.

Моделирование

TiDE поддерживает моделирование работы схемы на всех уровнях проектирования. Программа на Haste может быть откомпилирована в поведенческое Verilog-описание или в System-C модель для быстрого функционального моделирования при помощи существующих стандартных средств моделирования и генерации тестовых векторов. Кроме того, TiDE поддерживает моделирование до и после разводки библиотечных элементов в кристалле для более точного определения быстродействия работы схемы. Результаты моделирования могут быть визуализированы как средствами самой среды TiDE (htview), так и сторонними программами для анализа и визуализации результатов моделирования. Модуль htcover анализирует результаты моделирования на уровне логических вентилей и показывает статистические результаты покрытия кода тестовыми воздействиями, а также статистику использования операторов и историю значений переменных и выражений. Анализ производительности схемы проводится при помощи компонента htprof, который идентифицирует критические пути в схеме по Haste-описанию и вычисляет время их прохождения.

Создание прототипа на ПЛИС

Компонент htmap может по внутреннему описанию работы схемы (handshake-схема) создать список соединений в виде синхронной реализации на уровне регистровых передач (synchronous RTL netlist). Такая синхронная реализация схемы может быть использована для создания прототипа на ПЛИС, который позволяет эффективно проводить функциональную верификацию и отладку работы схемы на системном уровне.

Структурное проектирование

TiDE поддерживает модульное проектирование, таким образом, Haste-описания различных модулей могут быть откомпилированы отдельно и потом скомпонованы при помощи компонента htlink, как показано на рис. 5. Также htlink может подключать к проекту обычные комбинационные блоки (мультиплексор, например) и автоматически преобразовывать парафазные сигналы в двоичное представление.

Модуль htscan позволяет анализировать пути прохождения данных и управляющие цепи аналогично анализу в синхронных схемах. Модуль htremodel подготавливает файлы для программ автоматической генерации тестовых воздействий (ATPG files) после размещения элементов на кристалле (placement and routing), как показано на рис. 6. Такие программы производят анализ тестового покрытия и локализуют все найденные ошибки. В модуле htpost происходит генерация файлов с ограничениями в формате SDC (Synopsys Design Constraint) и внутреннем формате HCF (Handshake Constraint Format). Возможно преобразование форматов файла ограничений при помощи TCL-скриптов.

Проектирование на физическом уровне

В процессе проектирования на физическом уровне (layout) используются стандартные САПР, файлы ограничений и управляющие скрипты, как показано на рис. 6.

Верификация

Верификация работы схемы поддерживается на всех уровнях проектирования — от создания структурного Verilog-описания в модуле htmap до генерации списка соединений после размещения элементов в кристалле (post-layout netlist). Также имеется возможность проверки на эквивалентность схемы, реализованной на основе различных технологий производства микросхем.

Создание CCC при помощи существующих САПР для синхронных схем

Второй метод создания CCC использует NCL-логику (Null Convention Logic), разработанную специалистами компании Theseus Research [2], и подробно описан в [5, 6].

Суть метода показана на рис. 7 и заключается в следующем: разработчик проектирует схему в строго самосинхронном NCL-базисе с использованием пороговых элементов с гистерезисом. Для функций 4 и менее переменных таких элементов 27, и они составляют базис для проектирования. Каждый такой

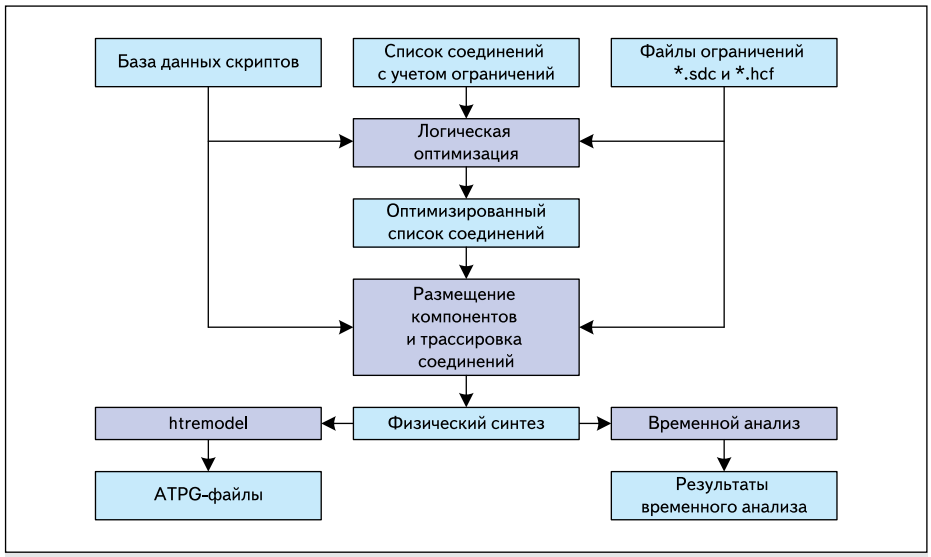

Рис. 6. Проектирование на физическом уровне

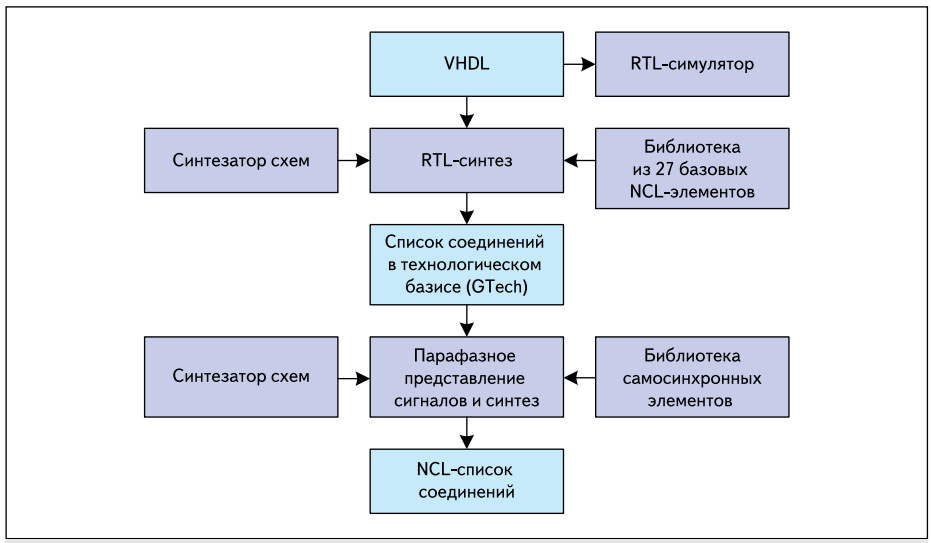

Рис. 7. Проектирование ССС при помощи существующих САПР для синхронных схем

элемент имеет функцию установки set, например, для двухвходового элемента TH12 функция set определяется выражением A+B, где A и B — сигналы на входах, a set — функция установки логической «1» на выходе элемента TH12. Создается библиотека из 27 базовых элементов на основании описания функции set. Например, описание элемента TH12 на VHDL имеет следующий вид:

```
-----------------------------------------
-- th12x0
-----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity th12x0 is
   port(a: in std_logic;
        b: in std_logic;
        z: out std_logic);
end th12x0;

architecture archth12x0 of th12x0 is
begin
   th12x0: process(a, b)
   begin
      if a = '0' and b = '0' then
         z <= '0' after 113.26 ps;
      elsif a = '1' or b = '1' then
         z <= '1' after 54.2 ps;
      else
         z <= a or b;
      end if;
   end process;
end arch th12x0;
```

Проектировщик выделяет комбинационную часть схемы, как показано на рис. 1, и описывает ее в таком базисе. Далее при помощи библиотеки из 27 базовых элементов и синтезатора схем (например, Design Compiler от Synopsys) создается RTL-описание в технологическом базисе (например, на двухвходовых элементах И-НЕ) и производится оптимизация схемы. Следующим этапом является представление схемы, описанной в технологическом базисе, в виде NCL-элементов с использованием парафазного кодирования сигналов и библиотеки NCL-элементов, описывающих базовые функции (И, ИЛИ, НЕ, ИСКЛЮЧАЮЩЕЕ-ИЛИ) (см. рис. 8 в КиТ № 10 `2009, стр. 105).

Окончательным этапом является синтез схемы с использованием парафазного представления сигналов. Итоговый список соединений, полученный таким образом, содержит только элементы TH22, TH34w22 и TH24w2, соединенные между собой.

Такой метод синтеза ССС позволяет разработчику использовать привычные ему САПР для синтеза, и нет необходимости изучать и покупать дополнительное программное обеспечение для проектирования строго самосинхронных схем.

В заключение отметим, что существующая методология проектирования ССС только начинает свое развитие, и от разработчика требуется кардинальное изменение мышления при переходе к самосинхронной схемотехнике. Но она дает огромный выигрыш в быстродействии и энергопотреблении схем по сравнению с синхронными аналогами. ■

Литература

1. Sparsø J., Furber S. Principles of Asynchronous Circuit Design: A Systems Perspective. Kluwer Academic Publishers, 2001.
2. Fant K. M. Logically Determined Design: Clockless System Design with NULL Convention Logic. John Wiley & Sons, 2005.
3. Taubin A., Cortadella J., Lavagno L., Kondratyev A., Peeters A. Design Automation of Real-Life Asynchronous Devices and Systems. Foundations and Trends(r) in Electronic Design Automation. Vol. 2, No. 1, September 2007.
4. Balsa manual 3.5: http://intranet.cs.man.ac.uk/apt/projects/tools/balsa/
5. Ligthart M., Fant K., Smith R., Taubin A., Kondratyev A. Asynchronous design using commercial HDL synthesis tools. Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems // IEEE Computer Society Press, April 2000.
6. Kondratyev A., Lwin K. Design of asynchronous circuits using synchronous CAD tools // IEEE Design & Test of Computers. Vol. 19. No. 4. 2002.