Иосиф Каршенбойм, Кирилл Паленов

«Встроенный» логический анализатор – инструмент разработчика «встроенных» систем.

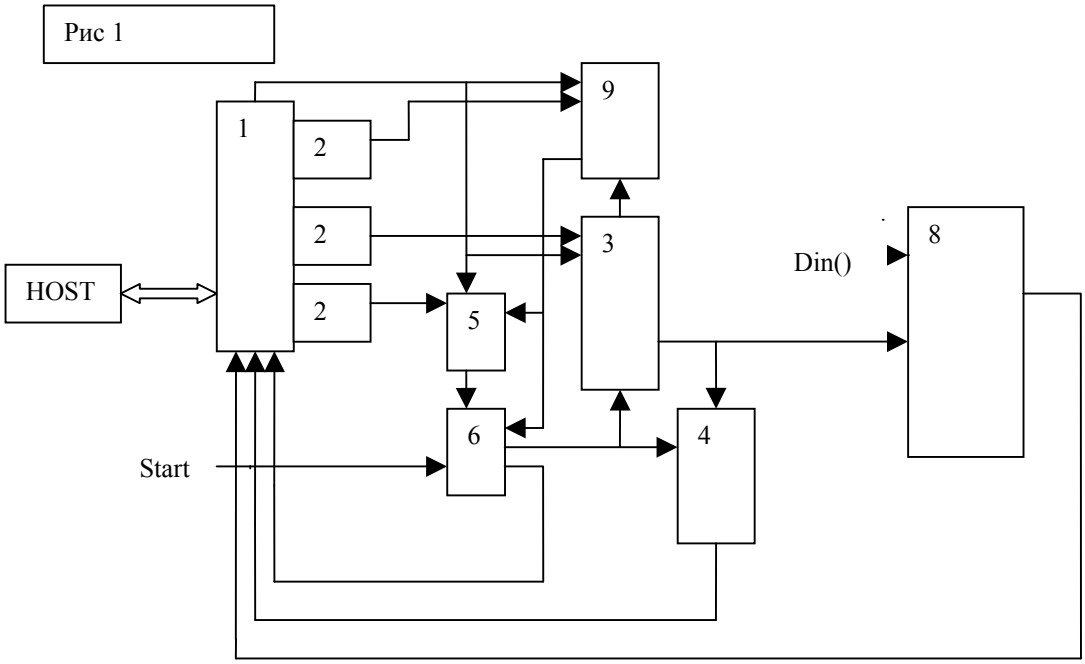
В настоящее время происходит резкое увеличение степени интеграции FPGA. Термин «система на кристалле» теперь уже никого не удивляет. Кристаллы большой емкости становятся более дешевы и более доступны для российских разработчиков. В данной статье автор хочет поделиться опытом отладки встроенных систем.

При общем увеличении ресурсов внутри кристалла – логических ячеек и памяти наблюдается значительное увеличение трудоемкости отладки и верификации всего проекта. Проекты имеют иерархическую структуру, в которых одни блоки являются составными частями для других блоков Множество цифровых автоматов, встроенные микроконтроллеры, контроллеры приемопередатчиков, такие например, как контроллер MAC-адреса для Ethernet и др. составляют «начинку» системы. Описания подобных блоков, состоят из большого числа файлов, каждый из которых может содержать до тысячи строк кода. Конечно, каждый файл и каждый блок проходят полную симуляцию при описании блока на HDL, и каждый блок отлаживается автономно, по мере возможности. Но при симуляции не всегда возможно воспроизвести все режимы работы блока. Например, симуляция 32-х канального HDLC-контроллера в пакете MAXPLUS2 при длине обрабатываемой последовательности, хотя бы из 10 байт приводит к диаграмме, состоящей из 32 * 10 * N тактов, где N – такты автомата контроллера, обрабатывающего один канал.
Языки описания аппаратуры являются объектно-ориентированными языками. Поэтому при подключении блока в верхнем файле видны только входы и выходы блока, но не видно его содержимое, и так далее до самого верхнего файла проекта. Если мы хотим увидеть как ведет себя сигнал в блоке, подкюченном в самом низу иерархии файлов, то этот провод приходится «тащить» через все файлы, исправляя и добавляя входы и выходы всех файлов от самого нижнего до самого верхнего, корректируя *.inc – файлы и непрерывно компилируя проект. Когда же провода от нижних файлов подведены к контрольным точкам, то неожиданно оказывается, что контрольных точек очень мало, все сигналы на них не вывести и не подключить к осциллографу или логическому анализатору.

Теперь несколько слов об осциллографах.
Не все помнят славные времена первых микропроцессоров серии КР580, когда мы пытались разглядеть бледно-зеленые сигналы в осциллографе С1-55. Сейчас, поработав, с Hp54645D, вспоминаешь об этом с грустью. Конечно, цифровой осциллограф с 2-мя аналоговыми и 16-ю цифровыми лучами даже сравнивать со старыми аналоговыми осциллографами невозможно. Но всегда есть свое «но». Хорош Hp54645D, но дорог и не всегда доступен, так как он один на несколько групп разработчиков. А ведь есть большое количество фирм-разработчиков, которые не могут себе позволить купить цифровой осциллограф такого класса.
Где-же выход? И всегда-ли нужен дорогой цифровой осциллограф?

Итак переходим к сути предлагаемой статьи.
Постараемся решить обе проблемы одновременно, не потеряв качества цифрового осциллографа и решив проблему с сигналами, которые не видны из верхнего файла. Решение вопроса таково: разработаем блок под названием “Логический Анализатор” (ЛА), припишем ему параметр “USED = YES/NO”, разрядность входов “DIN” и число отсчетов “PIXELS”.
Блок ЛА должен запоминать входные данные в буферную память и затем, эти данные могут быть считаны HOST-машиной. Для этого у нас все есть: в FPGA имеются встроенные блоки памяти, которые позволяют создавать массивы памяти требуемой разрядности, далее системы большой сложности обычно имеют какую-либо шину, связывающую FPGA с внешним миром, либо для этого могут быть задействованы те-же контрольные точки. Тип интерфейса и способ передачи информации существенной роли не играют.
Наиболее важным здесь является то, что есть возможность, установив параметр “USED = NO” не занимать ресурс в том блоке (файле), куда помещен ЛА. Встроим ЛА в каждый отлаживаемый нижний блок. Остается только снабдить каждый блок входами и выходами для ЛА и подвести все сигналы через все блоки к HOST-машине. Теперь, установив в нужном блоке параметр ЛА “USED = YES”, мы получим средство для контроля сигналов в любом нижнем блоке, и для этого не нужно исправлять весь проект. Мало того, таких ЛА можно задействовать в проекте несколько и считывать с них данные поочередно. Такое решение позволит отладить каждый блок в проекте на реальном «железе», отладить часть проекта или весь проект. У микросхем FPGA, большей частью, имеется возможность установливать на ту-же самую посадочную площадку микросхему «большей емкости», поэтому на этапе отладки есть возможность встроить ЛА, а, после отладки, в серийных изделиях использовать микрохемы «меньшей емкости». Кроме того устройства с микросхемами FPGA сами по себе могут служить и внешними ЛА.

Блок-схема простейшего ЛА приведена на рис.1



1- узел согласования с шиной HOST-машины
2- выделители одиночных импульсов
3- счетчик текущего адреса памяти
4- регистр, запоминающий адрес памяти при старте
5- триггер готовности
6- триггер старта
7- выходная шина данных
8- блок памяти
9- счетчик числа записей в память

ЛА работает следующим образом.
До начала работы ЛА производится установка счетчика 3, т.е. необходимо указать сколько раз будет производиться запись в память после сигнала старта.
При обращении от HOST-машины сбрасывается триггер готовности и ЛА переходит в состояние ожидания сигнала старта. При этом входная информация начинает записываться в память ЛА, что позволяет увидеть состояние сигналов на входах ЛА до момента старта. При приходе сигнала старта в регистр 4 записывается текущее состояние счетчика адреса памяти 3 и взодится триггер старта 6. Триггер старта 6 разрешает работу счетчика числа записей 9. По выполнении заданного числа записей триггер старта 6 – сбрасывается, триггер готовности 5 взводится и ЛА переходит в исходное состояние.
Далее HOST-машина читает счетчик адреса при старте 4, добавляет или вычитает смещение, задаваемое пользователем, и записывает полученное значение в счетчик текущего адреса памяти 3. При чтении данных HOST-машиной производится автоинкремент счетчика 3. Таким образом, производится чтение только части блока памяти или всего блока памяти.
Такое построение ЛА позволяет строить диаграммы с требуемым смещением относительно сигнала старта, т.е. сигнал старта может быть как в начале диаграммы, как у аналоговых осциллографов, так и в середине или в конце диаграммы как у цифровых ЛА.
Поскольку данный ЛА находится «целиком в руках пользователя», то это позволяет варьировать условия запуска ЛА по желанию пользователя, от любого сигнала в блоке пользователя.
Тактовая частота записи данных также может быть выбрана пользователем. Это может быть как тактовая частота, используемая в конкретном блоке пользователя, так и внешяя частота. Мало того, тактовая частота может переключаться в зависимости от состоятий сигналов в блоке пользователя.
Ла, построенный таким образом позволяют строить диаграммы сигналов в любом требуемом пользователем формате, при любой требуемой развертке и любом, задаваемом параметрически, числе отсчетов данных. То есть один и тот же блок внутренней памяти можно сначала использовать как ЛА, имеющий 16 лучей при 128 отсчетах и получить «панораму» событий, далее переключиться на 8 лучей при 265 отсчетах и так далее до 1 луча при 2048 отсчетах.

Приведем фрагмент файла – рапорта, полученного при компиляции ЛА, имеющего 8 лучей при 256 отсчетах.

** DEVICE SUMMARY **

Chip/ POF	Device	Input Pins	Output Pins	Bidir Pins	Memory Bits %	Memory Utilized	LCs	LCs % Utilized
oscill	EPF10K30ETC144-1	25	21	0	2048	8 %	105	6 %
User Pins:		25	21	0				

Всего 105 ячеек, большая часть из которых занята привязкой асинхронного интерфейса host-машины к внутреннему синхронному проекту на системной тактовой частоте 33МГц.

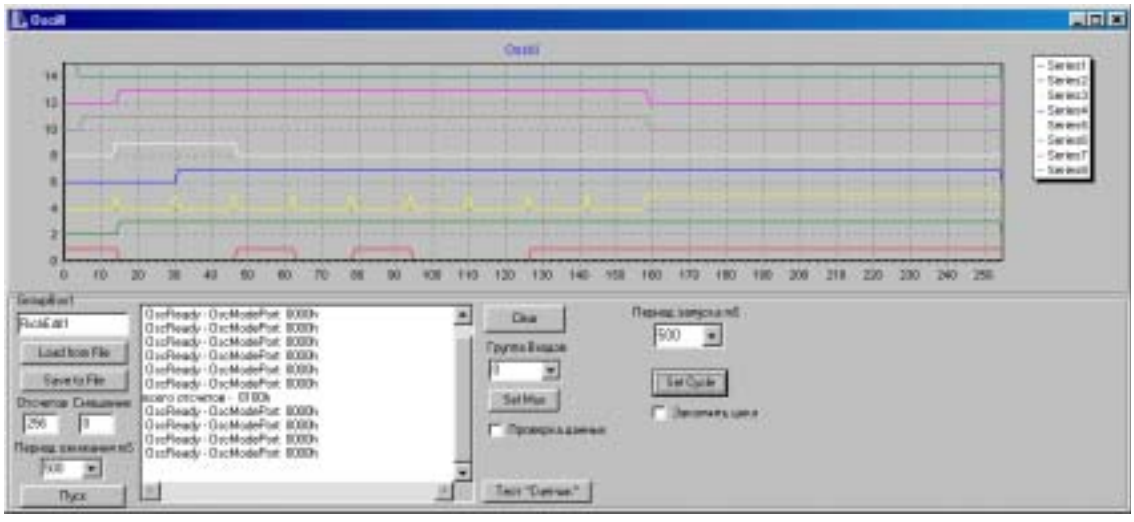
Теперь подключаем ЛА к отлаживаемому блоку MCS51 UART:
osc – логический анализатор,
uartbaud.baud_gen – выход генератора тактовой частоты для передачи данных,
uart – исследуемый блок.

Здесь приведен фрагмент AHDL-кода:

```
-- входы осцилл
        osc.din[7] = host_start;        -- сигнал старта от host-машины
        osc.din[6] = uart.send;         -- сигнал начала передачи стартового импульса
        osc.din[5] = uart.start_tx;     -- запомненный сигнал старта от host-машины
        osc.din[4] = uart.ninth;
        osc.din[3] = uart.data_tx;      -- строб передачи данных
        osc.din[2] = uart.stop_gen_tx;
        osc.din[1] = uart.send8;
        osc.din[0] = uart.txd;          -- данные на передачу, код 0xCA

        -- управление записью
        osc.start  = host_start;
        osc.din_ena = !uart.send # uartbaud.baud_gen;
```

На рис2 приведена диаграмма, снятая при отладке MCS51 UART



На приведенной диаграмме используется ЛА, занимающий только один блок внутренней памяти, что дает всего 256 отсчетов, но, как видно из фрагмента AHDL-кода, здесь применена нелинейная развертка по времени. От момента записи host-машиной сигнала разрешения работы запись в память производится на системной тактовой частоте – в данном случае 33МГц, что позволяет проследить всю работу триггеров при начале передачи данных в UART, а затем, после установки сигнала «uart.send»,

т.е. на этапе выдачи данных UART'ом частота записи понижается с системной до частоты генератора «baud_clk_gen».
Таким образом, при очень скромных затраченных ресурсах, можно получить сервис настоящего цифрового осциллографа.
Если пользователь имеет возможность затратить несколько больше ресурсов, то появляется возможность реализовать уже не простейший ЛА, а более функциональный ЛА, имеющий гораздо больше возможностей как по быстродействию, так и по возможности отображения записанной информации.
При необходимости проверить большее число сигналов на вход ЛА может быть подключен коммутатор сигналов и, таким образом, можно расширить возможности работы ЛА внутри отлаживаемого блока.

Программа управления логическим анализатором представляет собой удобный интерфейс взаимодействия с ЛА и расширяет его возможности и область применения. Основное ее назначение в отображении и сохранении данных, полученных из ЛА. Программа позволяет просматривать значение сигнала в любой момент времени, масштабировать графики, а также вводить на графики дополнительную информацию, такую как маркеры и подписи.

Кроме того, существует возможность объединения групп каналов в шину. Данные могут быть представлены как по отдельным каналам (каждый в виде отдельного бинарного графика), так и в виде waveform (единое представление всех каналов одновременно). При этом информация может быть представлена в цифровом виде либо в виде эквивалентного аналогового сигнала. Амплитуда результирующего графика определяется количеством каналов, объединенных в шину и их значением.

Возможно различное трактование отображения сигналов шин. Так, например, для шины I2C можно отображать адресную часть кадра или данные кадра в виде цифровых значений, при этом сигналы могут быть представлены в прямом или инверсном виде.

Помимо этого может быть реализована функция сохранения и загрузки данных в различных форматах: в бинарном (WFX-файлы), в виде пар отдельных отсчетов, записанных в текстовом виде (CSV- и PRN- файлы) или в графическом формате (PCX- PaintBrush). Это позволяет импортировать и экспортировать данные в другие программы, такие как MATLAB, MATCAD, Exel и другие, что, в свою очередь, позволит выполнить обработку полученных данных, а также загружать и отображать эталонные и ожидаемые данные.

Программная оболочка может так же представлять из себя простейший визуальный компонент из имеющихся библиотек, либо быть специализированной программой. Поскольку данный ЛА, с точки зрения интерфейса так же находится «целиком в руках пользователя», то и логика работы с шиной host-машины определяется пользователем. Именно это и определяет легость встраивания программной поддержки ЛА в тестовые, а при необходимости и в рабочие программы пользователя. Это позволяет контролировать работу ЛА непосредственно из программ пользователя, что значительно облегчает работу и требует меньших затрат ресурсов host-машины.

Иосиф Каршенбойм
Ik@lmail.loniis.spb.su

Кирилл Паленов
Kirill@rts.loniis.ru

Литература

1. И. Каршенбойм, Н. Семенов / Микропрограммые автоматы на базе специализированных ИС. – Chip News, №7, 2000.