HDL-реализация контроллера флэш-памяти

Алексей Гребенников (г. Актау, Казахстан)

Статья содержит описание контроллера флэш-памяти на языке Verilog. Рассмотрены принципы действия блоков контроллера со схемами конечных автоматов основных модулей. Приведена схема системы на кристалле, обеспечивающей обмен данными через шину Wishbone.

Введение

Отладочная плата SP605 для ПЛИС Xilinx семейства Spartan 6 содержит большое количество периферийных устройств, одним из которых является флэш-память типа CF (Compact Flash). Эта память подключается к ПЛИС не напрямую, а через контроллер System ACE (System Advanced Configuration Environment). При таком подключении флэш-память может конфигурировать ПЛИС или служить внешним носителем данных. Если память работает как внешний носитель, требуется дополнительный контроллер в составе ПЛИС (далее – контроллер CF), который, как правило, является частью системы на кристалле и обеспечивает чтение/запись данных из памяти для других устройств системы. При этом все низкоуровневые операции выполняются контроллером System ACE, а контроллер CF генерирует только высокоуровневые команды, например, команды чтения/записи определённого числа блоков данных.

Данная статья содержит описание контроллера CF на языке Verilog. Все исходные файлы проекта находятся в архиве *CF_cntr_verilog.zip*, который можно скачать с интернет-страницы журнала. Проект построен в программной среде PlanAhead 12.3; моделирование системы проводилось в среде ModelSim.

Схема подключения памяти к ПЛИС изображена на рисунке 1. Контроллер System ACE конфигурирует ПЛИС через интерфейс JTAG. Конфигурационный файл может быть считан из различных источников, в зависимости от установки переключателей на плате, – внешнего порта JTAG, встроенных параллельной и последовательной флэш-памяти и внешней карты памяти типа CF. При использовании памяти в качестве внешнего носителя обмен данными происходит через порт MPU. Этот порт представляет собой набор регистров в адресном пространстве контроллера System ACE; подробное описание регистров приведено в [1]. Контроллер CF является частью системы на кристалле и обеспечивает обмен данными с внутренними потребителями через шину Wishbone.

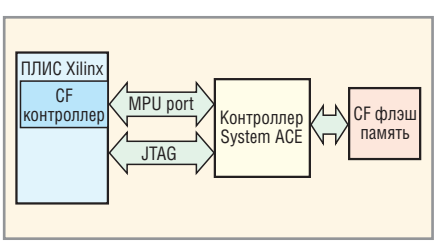

Рис. 1. Схема подключения памяти к ПЛИС

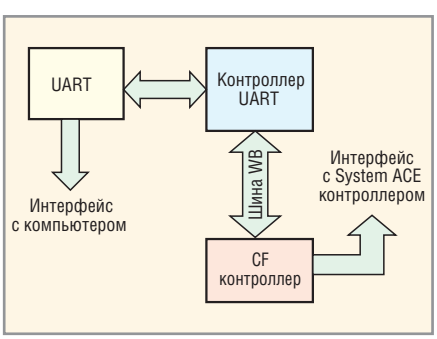

Рис. 2. Блок-схема системы на кристалле

Контроллер CF

Контроллер CF является частью системы на кристалле, изображённой на рисунке 2. Модуль верхнего уровня системы находится в файле *sopc.v*. Для обмена данными с компьютером используется асинхронный приёмопередатчик (UART) [2]. Эта простейшая система не имеет специального контроллера шины Wishbone, – обмен данными между контроллерами UART и CF происходит напрямую. В системе на кристалле модуль верхнего уровня контроллера CF представлен в файле *sdcard_cntr.v*. Этот модуль непосредственно генерирует управляющие сигналы для контроллера System ACE и содержит два дополнительных модуля – контроллер шины Wishbone *wb_cnt_ace.v* и конечный автомат *sdcard_fsm.v*.

Как упоминалось выше, обмен данными между контроллерами CF и System ACE происходит через порт MPU, который определяется следующими сигналами:

- output [6:0] mpa_o – адресные линии, задающие адрес регистра;
- inout [7:0] mpd_io – линии данных. Шина данных может быть 8- или 16-битной.

Плата SP605 имеет восьмибитную двунаправленную шину данных, поэтому контроллер сконфигурирован для этой разрядности:

- output mpce_n_o – сигнал Chip Enable, активация контроллера;
- output mpwe_n_o – сигнал Write Enable, разрешение записи;
- output mpoe_n_o – сигнал Output Enable, чтение данных;
- input mpbrdy_i – готовность буфера данных для операций чтения/записи.

В данной версии контроллера прерывания не используются.

При восьмиразрядной шине данных, по адресам 0h – 3Fh расположены различные конфигурационные и контрольные регистры контроллера System ACE. В адресном пространстве 40h – 5Fh расположен 32-байтный буфер данных.

Для корректной работы необходима синхронизация рабочих частот контроллеров CF и System ACE. На плате SP605, согласно схеме [3], рабочая частота (33 МГц) контроллера System ACE задаётся генератором U29. Этот же сигнал доступен в ПЛИС и является рабочей частотой системы на кристалле.

Рассмотрим более подробно работу отдельных блоков контроллера CF.

Контроллер шины Wishbone

Модуль контроллера шины Wishbone представлен в файле *wb_cntr_ace.v*. Этот модуль обеспечивает приём команд, данных (от контроллера UART), а также передачу данных, считанных из флэш-памяти. Модуль содержит в своём составе конечный автомат, изображённый на рисунке 3. В режиме ожидания конечный автомат находится в состоянии *WBSD_IDLE*. Для

передачи данных внешним устройствам автомат переходит в состояние *WBSD_BWRITE*. В этом случае контроллер становится мастером шины Wishbone. После завершения передачи пакета данных конечный автомат возвращается в состояние *WBSD_IDLE* и ожидает дальнейших команд. Для декодирования команд от контроллера UART конечный автомат переходит в состояние *WBSD_DECODE*. Команды декодируются по смещению адреса относительно базового адреса шины Wishbone для контроллера CF. Базовый адрес задаётся константой *WBSD_BASE* в файле *sd_parms.v*.

Конечный автомат различает следующие команды в зависимости от смещения базового адреса:

- *WBSD_BASE* + *0* – чтение одного восьмибитного регистра. Адрес регистра для чтения задаётся младшими 7 битами слова данных на шине Wishbone. Прочитанный байт затем передаётся через шину Wishbone как младшие 8 бит 32-битного слова. Эта команда используется для чтения контрольных и конфигурационных регистров контроллера System ACE;
- *WBSD_BASE* + *1* – запись одного восьмибитного регистра. Адрес регистра для записи задаётся младшими 7 битами слова данных на шине Wishbone, и байт для записи определяется битами 15-8;
- *WBSD_BASE* + *2* – чтение группы регистров. Эта команда считывает четыре последовательных регистра и передаёт их через шину Wishbone как одно 32-битное слово. Адрес первого регистра задаётся младшими 7 битами слова данных шины Wishbone; адреса регистров 2-4 получаются путём прибавления единицы к первоначальному адресу. Например, если в команде задан адрес 7'h10, будут считаны регистры по адресам 7'h10, 7'h11, 7'h12, 7'h13;
- *WBSD_BASE*+ *3* – запись группы регистров. Команда записывает четыре регистра в адресном пространстве контроллера System ACE. Первое слово данных на шине Wishbone задаёт начальный адрес регистра, и следующее 32-битное слово содержит четыре байта для записи;
- *WBSD_BASE* + *4* – чтение буфера данных. Команда задаёт число буферов данных, которые необходимо прочитать. Каждый буфер содержит 32 байта данных;
- *WBSD_BASE* + *5* – запись буфера данных. Команда задаёт число буферов данных, которые необходимо записать. Один буфер, так же как и для команды чтения буфера, состоит из 32 байтов данных, которые передаются через шину Wishbone вслед за командой.

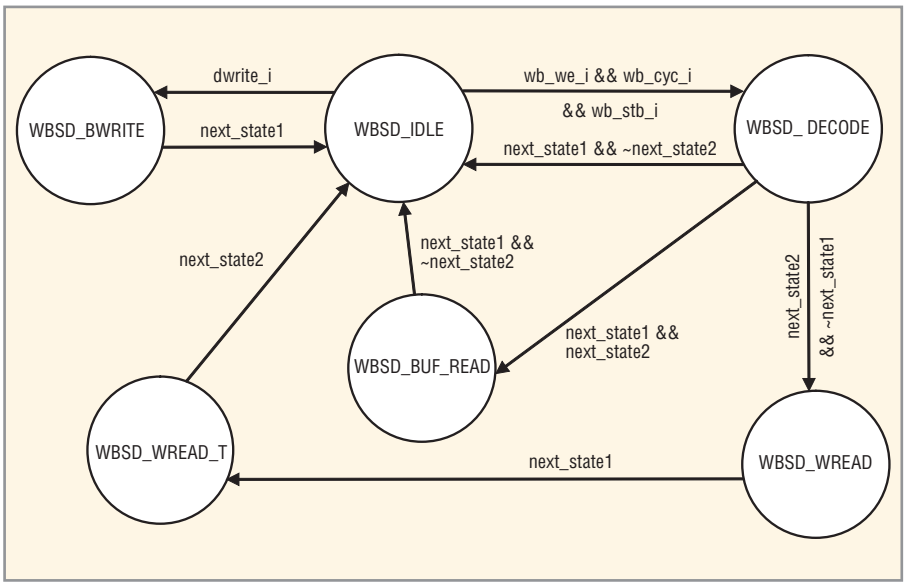

Рис. 3. Конечный автомат контроллера шины Wishbone

Полученные команды контроллер шины Wishbone передаёт конечному автомату контроллера CF. Рассмотрим более подробно работу этого конечного автомата.

Конечный автомат контроллера CF

Модуль конечного автомата CF контроллера находится в файле *sdcard_fsm.v*. Блок-схема автомата изображена на рисунке 4. В начальный момент времени конечный автомат ожидает команды от контроллера шины Wishbone, находясь в состоянии *SD_IDLE*. Обработка команд начинается после установки регистра *cmd_i* в соответствующее значение, при этом контрольные сигналы для контроллера System ACE формируются модулем *sdrcard_cntr.v* на основании состояний конечного автомата контроллера CF.

Рассмотрим более подробно каждую команду.

cmd_i == *CMD_SRR*. команда чтения одного регистра. После получения этой команды автомат переходит в состояние *SD_SRR* и формирует управляющие сигналы для контроллера System ACE [1]. В этом состоянии автомат читает один регистр контроллера ACE согласно параметрам, заданным в команде. После завершения операции чтения автомат переходит в состояние *SD_SRR_WB*. В этом состоянии прочитанный регистр передаётся через шину Wishbone контроллеру UART, при этом старшие 24 бита заполняются нулями.

cmd_i == *CMD_SRW* команда записи одного регистра. Автомат переходит в состояние *SD_SRW*, формирует управляющие сигналы для контроллера System ACE и затем возвращается в состояние *SD_IDLE*.

cmd_i == *CMD_MRR* автомат читает четыре байта данных. Временная диаграмма управляющих сигналов для этого состояния схожа с диаграммой для состояния *SD_SRR*, за исключением того, что считываются четыре байта вместо одного. После чтения четырёх регистров автомат переходит в состояние *SD_MRR_WB* и передаёт прочитанные данные через шину Wishbone. Все 32 бита слова в этом случае содержат полезные данные.

cmd_i == *CMD_MRW* запись четырёх регистров. Временная диаграмма этого состояния схожа с диаграммой для состояния *SD_SRW* с той лишь разницей, что записываются четыре регистра вместо одного.

cmd_i == *CMD_RDB* получив эту команду, автомат начинает чтение одного буфера данных контроллера System ACE. Буфер данных представляет собой 32-байтную память типа FIFO, расположенную по адресам 7'h40 – 7'h5F, следовательно, возможно чтение любого байта в пределах буфера. Однако алгоритм чтения буфера использует

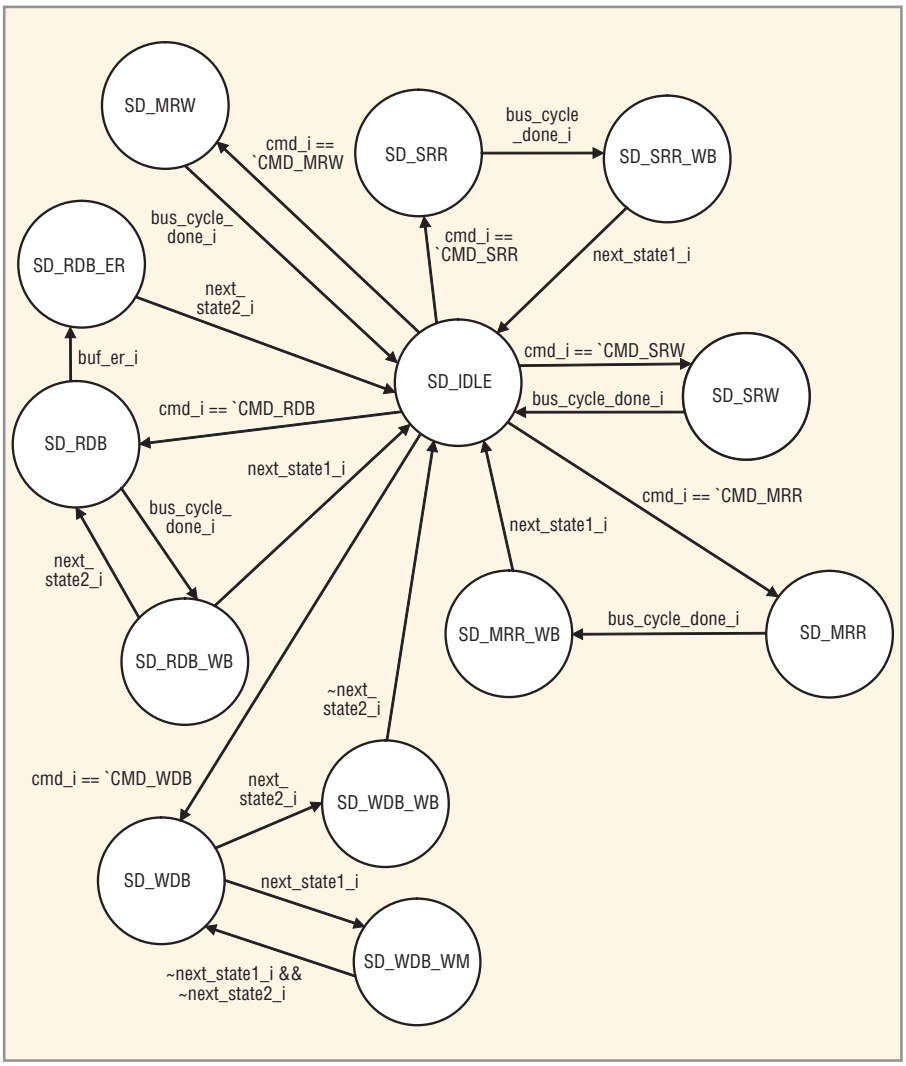

Рис. 4. Конечный автомат контроллера CF

только адреса 7'h40 и 7'h41. Первым считывается байт по адресу 7'h40, затем после считывания байта по адресу 7'h41 происходит автоматический сдвиг данных FIFO-памяти буфера, поэтому следующий байт вновь читается по адресу 4'h40. Этот цикл продолжается до тех пор, пока не будет прочитано 32 байта – один полный буфер. Заметим, что в отличие от команд чтения индивидуальных регистров, которые всегда доступны, при чтении буфера данных используется дополнительный сигнал готовности буфера данных *mpbrdy_i*. Если во время чтения буфера данных происходит ошибка, автомат переходит в состояние *SD_RDB_ER*, в котором он передаёт контроллеру UART одно слово с кодом ошибки и затем возвращается в состояние *SD_IDLE*. В случае успешного чтения буфера автомат переходит в состояние *SD_RDB_WB*, в котором происходит передача считанного буфера через шину Wishbone. При помощи одной команды *CMD_RDB* возможно чтение нескольких буферов данных. Таким образом, после передачи буфера данных внешним потребителям в состоянии *SDB_RDB_WB* автомат анализирует общее количество буферов для чтения. Если требуется считать больше буферов, конечный автомат вновь переходит в состояние *SD_RDB* и читает следующий буфер. Если считанный буфер был последним, контроллер переходит из состояния *SD_RDB_WB* в состояние *SD_IDLE*, – чтение всех буферов завершено.

cmd_i == CMD_WDB команда записи буфера данных в адресное пространство контроллера System ACE. Так же как и для операции чтения, буфер записи в этом случае представляет собой память типа FIFO, расположенную по адресам 7'h40 – 7'h5F. При этом запись происходит только по адресам 7'h40 и 7'h41. После записи байта по адресу 7'h41 происходит автоматический сдвиг памяти FIFO, и следующий байт вновь записывается по адресу 7'h40. В состоянии *SD_WDB* автомат считывает одно 32-битное слово на шине Wishbone и затем переходит в состояние *SD_WDB_WM*, в котором происходит запись этого слова в память FIFO контроллера System ACE. Адресные линии во время этого цикла будут принимать значения 7'h40 – 7'h41 – 7'h40 – 7'h41, как это видно по результатам симуляции (см. рис. 5). После записи одного слова автомат вновь переходит в состояние *SD_WDB* и считывает следующее слово с шины Wishbone. Цикл продолжается до тех пор, пока не будет записан хотя бы один буфер данных – 32 байта. Но возможна передача и большего количества буферов за одну команду. После завершения цикла записи буферов автомат переходит в состояние *SD_WDB_WB*. В этом состоянии контроллеру UART передаётся слово, содержащее статус операции записи. В случае успешного завершения операции передаётся слово *C_PATTERN = 32'h00000002*. В случае ошибки передаётся слово *TO_PATTERN = 32'h00000001*. Так же, как и в случае чтения буфера, необходимо, чтобы память FIFO контроллера System ACE была в состоянии

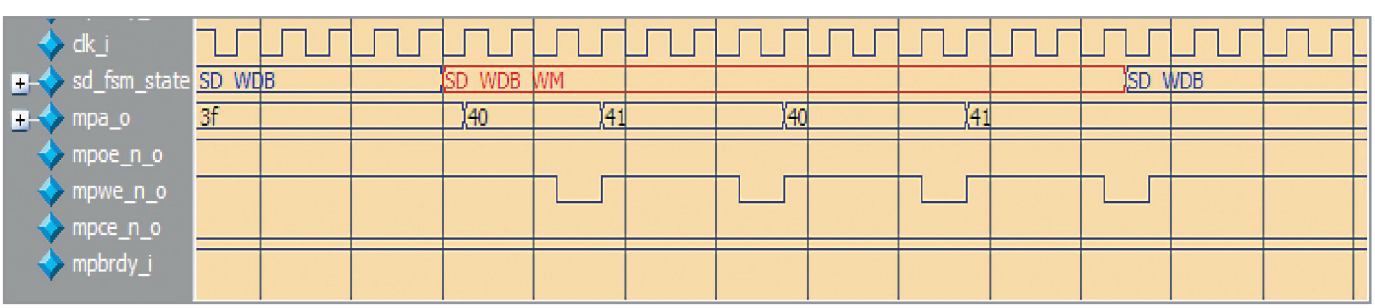

Рис. 5. Симуляция работы конечного автомата контроллера CF

готовности принимать данные (сигнал *mpbrdy_i*). Более подробно временные диаграммы операций чтения и записи буфера данных описаны в [1].

Для проверки работоспособности контроллера использовалась программа XilinxCOM, исходные коды которой приведены в архиве *XilinxCOM.zip*. Также необходима флэш-память с максимальным размером сектора 2 Гб и файловой системой FAT12 или FAT16. Эти ограничения обусловлены контроллером System ACE. Рассмотрим более подробно процесс верификации контроллера.

Верификация контроллера

Программа XilinxCOM обменивается данными с контроллером UART. Первый байт данных, переданных контроллеру, содержит код команды. Для верификации контроллера CF необходима только одна команда записи данных 8'h33. Базовый адрес записи для всех команд равен 32'hFFFF0000. Различные варианты команд в текстовом формате приведены в архиве *test_seqs.zip*. Рассмотрим некоторые из них.

test33_srr.asm: 33 FFFF0000 00000001 00000013. Для всех команд первые десять байт данных – это служебные слова, декодируемые контроллером UART; 33 – код команды; FFFF0000 – адрес в адресном пространстве шины Wishbone (именно по этому базовому адресу контроллер CF декодирует команды). Адрес FFFF0000 декодируется как команда чтения одного регистра; 00000001 – параметр, определяющий, сколько 32-битных слов данных необходимо записать через шину Wishbone; 00000013 – адрес регистра, который необходимо прочитать. В результате выполнения этой команды программа XilinxCOM отображает одно 32-битное слово. Младшие 8 бит представляют прочитанный байт данных.

test33_srw.asm: 33 FFFF0001 00000001 00007910. Адрес FFFF0001 декодируется как команда записи одного байта. Адрес байта – 10, значение байта – 79.

test33_mrr.asm: 33 FFFF0002 00000001 00000015. Адрес FFFF0002 декодируется как команда чтения четырёх байт данных. Адрес первого байта – 15. Четыре прочитанных байта переданы как одно 32-битное слово.

test33_mrw.asm: 33 FFFF0003 00000002 00000018 43ECBEA7. Адрес FFFF0003 декодируется как команда записи четырёх байт данных. Адрес первого байта – 18; 43ECBEA7 – данные для записи. В регистр по начальному адресу 18 будет записан наиболее значимый байт 43.

test33_rdb.asm: 33 FFFF0004 00000001 00000010. Адрес FFFF0004 декодируется как команда чтения буфера данных. Параметр 00000010 задаёт количество буферов – 16.

test33_wdb.asm: 33 FFFF0005 00000009 00000001 11111111. Адрес FFFF0005 декодируется как команда записи буфера данных. Параметр 00000001 задаёт количество буферов для записи – в данном случае один. Вслед за параметром следуют 32 байта буфера данных.

Как упоминалось выше, команды чтения/записи регистров не требуют предварительной подготовки, тогда как для команд работы с буфером данных требуется готовность буфера. Поэтому командам чтения/записи буферов всегда предшествуют команды, обеспечивающие готовность буфера и задающие некоторые другие дополнительные параметры.

Рассмотрим цепочку команд в файле *test33_prg_seq_16.asm*:

```
33 FFFF0001 00000001 00000218
33 FFFF0003 00000002 00000010
80200000
33 FFFF0001 00000001 00000114
33 FFFF0001 00000001 00000315
33 FFFF0001 00000001 00008218
33 FFFF0004 00000001 00000010
```

Первая команда устанавливает бит 1 регистра 18. По этому адресу расположен контрольный регистр *CONTROLREG*. Бит 1 – *LOCKREQ* – устанавливает блокировку со стороны контроллера. Установка этого бита означает, что контроллер CF выполняет операции обмена данными.

Следующая команда задаёт количество секторов, которые необходимо прочитать – по адресу 14 записывается значение 1. Один сектор равен 512 битам – минимальная единица обмена данными. В данном случае поступил запрос на чтение одного сектора – 512 байт (16 буферов).

Далее в регистр по адресу 15 записывается команда для контроллера System ACE; 3 – это команда чтения данных *ReadMemCardData*. После этого выполняется команда перезапуска контроллера System ACE – устанавливается бит *CFGRESET* регистра *CONTROLREG*.

После выполнения всех этих команд контроллер System ACE начинает считывать данные из флэш-памяти, т.е. буфер наполняется данными. Поэтому следующая команда считывает данные из 16 буферов (10 в шестнадцатеричной форме). После считывания одного сектора сигнал готовности буфера становится равным нулю – все данные переданы и буфер опустел. В данной версии контроллера этот сигнал, *mpbrdy_i*, был выведен на один из светодиодов для визуальной индикации заполнения буфера.

Аналогично происходит процесс записи. Здесь, так же как и при операции чтения, минимальной единицей является один сектор – 512 байт. Пример команд для записи буфера приведён в файле *test33_prg2.asm*.

При тестировании работы контроллера также является полезной цепочка команд (см. файл *test33_read_cfg_regs.asm*), считывающих значения всех контрольных регистров. Наиболее важным из служебных регистров является *STATUSREG*, который находится по адресам 7'h4 – 7'h7 и отображает состояние контроллера System ACE.

Заключение

В статье описана модель контроллера флэш-памяти для отладочной платы SP605. Этот контроллер работает в паре с аппаратным контроллером System ACE, установленным на плате SP605, и обеспечивает обмен данными между флэш-памятью и устройствами на ПЛИС. Данная модель может быть интегрирована в более сложную систему на кристалле. Контроллер CF был синтезирован для ПЛИС Spartan 6 и показал удовлетворительные результаты при тестировании.

Литература

1. System ACE CompactFlash Solutions. DS080 (v2.0). Xilinx. 2008. October 1.
2. *Гребенников А.* HDL-реализация асинхронного приёмопередатчика. Современная электроника. 2011. № 4.
3. SP605 board schematics, xtp067_sp605_schematics.pdf.