Standard SVEC Gateware

Programmer and User manual (Git revision: `gateware-v3.0-2-g1888b96`, November 2014)

CERN BE-CO-HT / Tomasz Włostowski

Table of Contents

1 Introduction

This document describes the default bitstreams that come preloaded with every SVEC card. As there are two FPGAs on the SVEC, there are two default bitstreams:

- **Bootloader bitstream**, residing in the System FPGA whose role is to start up the Application FPGA,
- **Golden bitstream**, residing in the Application FPGA, which allows the SVEC device driver to enumerate the mezzanines.

Note: The Golden bitstream is used exclusively by the SVEC device driver (see [3], page 8) in conjunction with the OHWR VME64x Core (see [4], page 8). If you are using a different VME core in your FPGA design (such as the `VmeIntfce` which is widely used at CERN), **the Golden Bitstream chapter is irrelevant for your application and there is no need to install the SVEC driver.**

2 The Bootloader

The System FPGA bootloader allows to boot the Application FPGA from the VME bus or from the onboard Flash memory and reprogram both the System and Application bitstreams in the flash via VME (see Figure 2.1). The boot process goes as follows:

- Power up,
- SFPGA checks for presence of a valid bitstream file for the Application FPGA in the Flash memory,
- If a valid bitstream has been found, the AFPGA is initialized with it,
- if not, the Bootloader enters passive mode. Upon reception of a boot sequence, if gives access to the Flash for the host or lets it program the AFPGA directly.

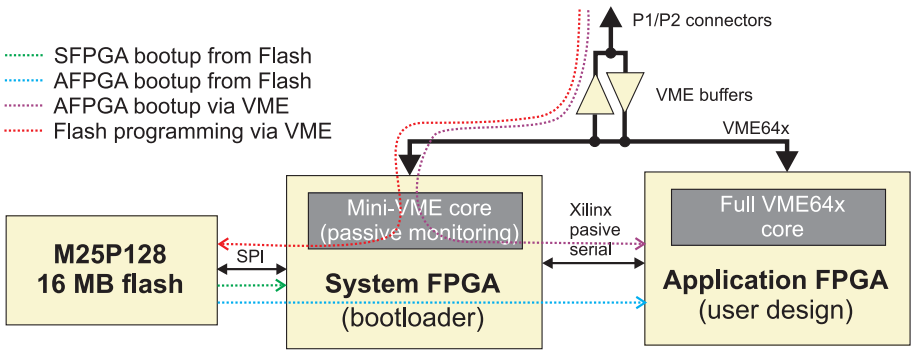

Figure 2.1: Boot modes of the SVEC card.

2.1 Bootloader versions

There are three versions of the bootloader in use:

- **Version 1** which only allows booting the Application FPGA via VME (no flash support). This version of the bootloader has been programmed in older SVEC cards. If the VME flasher (`svec-flasher`) tool is unable to detect the flash memory, you're likely running the old bootloader.
- **Version 2** which supports everything described in this manual except for providing SPI flash access to the AFPGA.

- **Version 3** which supports everything described in this manual, including accessing the SPI flash from the AFPGA.

The new version is software-compatible with the old one, there is no need to update any drivers. The register description applies to both versions, except that for the version 1, the Flash Access Register (`FAR`) is not functional.

2.2 VME Interface

The bootloader core supports only 32-bit data CR/CSR accesses from/to address range `0x70000` - `0x70020`, allowing for plug&play reprogramming of the cards only knowing their physical slot locations. All other transfers are ignored. The base address is `0x70000`, and corresponds to the `CSR` register. When the card is powered up, the VME interface stays in passive mode, monitoring VME accesses without ACKing them. This is to prevent conflicts with the CR/CSR space of the VME core in the Application FPGA.

2.3 Entering bootloader mode

In order to enter the bootloader, one needs to write the magic sequence of 8 following transfers: `0xde`, `0xad`, `0xbe`, `0xef`, `0xca`, `0xfe`, `0xba`, `0xbe` to the `BTRIGR` register (for register definitions, see [System FPGA Register Map], page 9).

In order to check if the bootloader has been activated, read the `IDR` register. It should be equal to `SVEC` ASCII string encoded in HEX. Any other value indicates that the boot trigger sequence was not correctly recognized, the System FPGA is unprogrammed, the geographical address of the card is wrong or that the card itself is faulty.

Note 1: Triggering bootloader mode causes automatic reset (un-programming) of the Application FPGA.

Note 2: Since the bootloader core supports only 32-bit transfers, one must extend the magic values with zeroes (e.g. `0x000000de`, etc.) and write full 32-bit words. Attempts to write the magic sequence as single bytes (D8 transfer mode) will be ignored.

Note 3: Trigger sequence must not be interleaved with other accesses to the bootloader address range of the same card.

Note 4: Write operations to `BTRIGR` register while the bootloader is in passive mode will not be acknowledged on the VME bus and may sometimes cause bus errors to be reported by the host VME driver. They are not harmful, though.

2.4 Programming the AFPGA

Programming the Application FPGA directly via VME involves the following steps:

- Reset the Xilinx Passive Serial boot interface by writing the `CSR.SWRST` bit,
- Set download speed by writing the `CSR.CLKDIV` field. Default value is `1`,
- Write the `CSR.START` bit and set endianness via the `CSR.MSBF` bit,
- Write the bitstream to the FIFO registers, observing FIFO full/empty status. The last transfer should have `FIFO_R1.XLAST` bit set to 1,
- Wait for assertion of `CSR.DONE`. `CSR.ERROR` bit indicates a problem during configuration (most likely, an invalid bitstream),
- Exit bootloader mode by writing 1 to `CSR.EXIT` bit.

A code example is available in the repository (see [2], page 8). Successful gateware download to the Application FPGA is indicated by turning on the “AFPGA DONE” LED in the middle of the PCB.

2.5 Programming the Flash

The SFPGA also allows raw access to the Flash memory (M25P128) via the `FAR` register. The code below shows how to execute a single SPI transaction (command + N data bytes).

Low-level details about programming M25Pxxx series Flash memories can be found in their datasheets (see [1], page 8). A simple VME flasher is provided in `software/vme-flasher` directory.

Note 1: It is advised to protect the region of the flash containing the system FPGA bitstream from being accidentally overwritten, as this will result in rendering the card unusable and will require re-programming the flash via JTAG. Details on memory protection can be found in the M25P series datasheet.

Note 2: The freshly-programmed bitstreams will be loaded into the FPGAs after power-cycling the card. In order to avoid the power cycle, one can boot the Application FPGA directly using the same bitstream.

2.6 Flash memory organization

The flash memory of the SVEC contains 16 Megabytes of data, that is 65536 pages of 256 bytes. The first 6 MBs are used for bitstream storage. The flash format is compatible with the SDB filesystem, with the SDB descriptor table located at offset `0x600000`. Locations of the bitstreams are fixed to:

- `0`: Raw bitstream for the System FPGA (up to 1 MB).
- `0x100000`: Raw bitstream for the Application FPGA (up to 5 MB).
- `0x601000 - 0xffffff`: Free space, foreseen for the AFPGA's private data storage.

An example script for building the default flash filesystem (containg the bootloader and golden bitstreams) is located in the `software/sdb-flash` subdirectory in the SVEC project's repository(see [2], page 8). Presence of the SDB descriptor table at `0x600000` is checked by the bootloader to prevent booting up from a corrupted or unprogrammed flash.

Note: Both bitstreams must be in raw (`.bin` file extension) format. `.bit`, `.mcs`, `.xsvf` and other formats will not work.

3 The Golden Bitstream

The SVEC Application FPGA golden bitstream is loaded by the SVEC driver during its startup. Its purpose is to:

- Query the board's serial number.
- Check the presence of the FMC mezzanines.
- Read out their I^2C identification EEPROMs.

The bitstream does not drive any of the mezzanine user/clock pins to protect from electrical damage resulting from mismatched I/O standards.

3.1 Block diagram

The bitstream is based on a generic "Golden Bistream" core and a One-Wire master core from the `general-cores` library. For further details, refer to the library's manual.

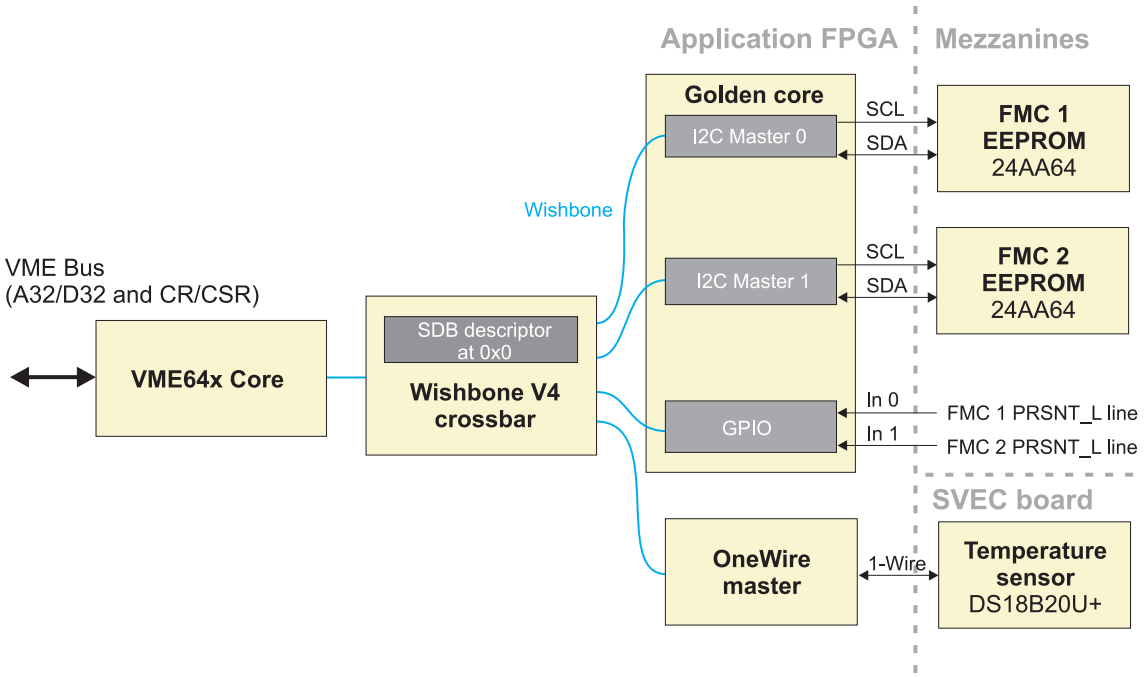

Figure 3.1: Block diagram of the SVEC golden gateware.

The Golden core is responsible for:

- FMC insertion detection, through `FMC_PRESENT` bits of the `CSR` register.
- FMC EEPROM readout, done by bit-banging the `I2CRx` registers.

The OneWire core allows reading the board's serial number (equal to S/N of the DS18B20U+ temperature sensor) and its temperature. The clock freqency for I^2C and OneWire dividers calculation is 62.5 MHz.

3.2 Memory map

Note: Please do not hardcode the base addresses of the cores, query them from the SDB descriptor. The SDB address of `0x0` is guaranteed to stay constant. Only A32/A24/D32/CSR address modifiers are supported.

Core	Base address	Library	Description
`sdb_rom`	`0x0`	`general-cores`	SDB descriptor.
`golden_core`	`0x10000`	local	Golden Bitstream core.
`xwb_onewire_master`	`0x12000`	`general-cores`	OneWire Master (temp sensor).

4 Flashing the SVEC

4.1 Programming Application FPGA Flash through VME

The SVEC Application FPGA can be programmed with the `svec-flasher` tool, located in `software/vme-flasher` subdirectory of the SVEC project repository. It requires a `.bin` format bitstream, that can be generated by Xilinx ISE by selecting "Generate binary configuration file" in *Generate Programming File* options.

The flasher requires the slot number as the first argument and the file with the bitstream as the second, just like in the example below:

```
# ./svec-flasher 5 file.bin
Bootloader version: 2
Programming the Application FPGA flash with bitstream file.bin.
Programming page 0/0.
Verification...
Programming page 16495/16495.
Verification...
Programming OK.
```

Note 1: Before programming the flash, please unload the SVEC kernel driver:

```
# rmmod svec
```

Note 2: The flasher must be run as root.

Note 3: The card must be rebooted for the new bitstream to be loaded to the Application FPGA.

4.2 Loading bitstream to the Application FPGA through VME

The `svec-flasher` tool can be also used to load a bitstream directly to the Application FPGA, without programming the Flash chip. In order to do so, pass the `-f` option to the flasher tool:

```
# ./svec-flasher 5 file.bin -f
Bootloader version: 2
Booting the Application FPGA with bitstream file.bin.
Bitstream loaded, status: OK
```

Warning: The programming operation will not work correctly if the SVEC driver is loaded. Please unload the SVEC kernel driver before proceeding.

4.3 Updating the bootloader through VME

Starting from the version 2, the bootloader can update itself via VME by using the `svec-flasher` tool. In order to update the bootloader, invoke the flasher with `-b` parameter.

```
# ./svec-flasher 5 svec-boot-v3.bin -b
Programming the Application FPGA flash with bitstream svec-boot-v3.bin.
Bootloader version: 3

WARNING! You're about to update the SVEC bootloader.
If this operation fails (due to incorrect bitstream or power loss),
the card can be only recovered through JTAG.
Type 'yes' to continue or Ctrl-C to exit the program: yes
Programming page 1331/1331.
Verification...
Programming OK.
```

Note: Programming incorrect bootloader bitstream or a failure of bootloader update process will render your card unusable. In such case, the card can be recovered by programming the bootloader through JTAG. The flasher tool asks for additional confirmation before reprogramming the memory.

4.4 Updating the bootloader through JTAG

Certain older SVEC cards have been shipped with the first version of the bootloader that does not support booting the AFPGA from the Flash memory. In order to use from the VME Flasher, an update is necessary. The procedure goes as follows:

- Download the latest bootloader `svec-bootloader-[latest release].mcs` from http://www.ohwr.org/projects/svec/wiki/Releases.
- Connect Xilinx JTAG programmer to the JTAG connector of the card to be updated.
- Launch ISE iMPACT.
- Double-Click "Boundary Scan" in the left pane ("iMPACT flows").
- Open the right-click menu in the main work area and select "Initialize chain" or press Ctrl+I.
- Right click on the "SPI/BPI ?" box above the "xc6slx9" FPGA and select "Add SPI/BPI Flash".
- Pick the `svec-bootloader-[latest release].mcs` file.
- Select flash type: SPI PROM, M25P128, data width: 1.
- Right click on the "FLASH" chip above the "xc6slx9" and select "Program". Select the "Verify" option and click OK.
- If everything went fine, "Programming succeeded" message will appear.
- Reboot the VME crate to use the new bootloader.

Note: Updating the bootloader does not require updating the drivers, as it is backwards-compatible.

4.5 Application FPGA Flash programming through JTAG

This method of programming may be useful during factory JTAG programming of SVECs, when both the bootloader and the application bitstream need to be loaded simultaneously. It requires a special flash image that contains both the bootloader and application bitstreams along with some SDB filesystem structures that allow the bootloader to correctly load the AFPGA file.

Preparing the flash image is straightforward:

- Copy your AFPGA bitstream (`.bin` format) to `software/sdb-flasher/fs/afpga.bin`.

- Run the script: `software/sdb-flasher/build.sh`. The script will produce the file called `image.mcs`.
- Program the Flash (`image.mcs`) via iMPACT, using the procedure described in the previous section.

The script requires the `gensdbfs` utility. The easiest way to install it is to:

- Clone the `fpga-config-space` repository:

```
git clone git://ohwr.org/hdl-core-lib/fpga-config-space.git.
```

- Compile it:

```
cd fpga-config-space/sdbfs/lib
make
cd ../userspace
make
```

- Copy `gensdbfs` binary to a directory within the `PATH`.

4.5.1 Note for Windows users

Preparation of the flash image described above requires some (currently) Linux-only tools. An altarnative method for flashing without these tools is described below:

- Open a hex editor and create an empty file of 6 MB and 4 bytes (`0x600004`).
- Place the bootloader bitstream (`.bin` extension) at offset `0x0`.
- Place the AFPGA bitstream (`.bin` extension) at offset `0x100000`.
- Place an SDB filesystem signature: `0x53 0x44 0x42 0x2D` starting at offset `0x600000`.
- Store the image in Intel HEX (`.mcs` extension format) and flash using Xilinx Impact.

4.6 Accessing the SPI Flash from the Application FPGA

The version 3 of the bootloader allows the Application FPGA to access the SPI interface of the Flash memory. Once the boot process is done, the System FPGA routes the following AFPGA pins directly to the Flash memory's SPI interface (Xilinx UCF file syntax):

```
NET "flash_sck_o" LOC=AG26;
NET "flash_mosi_o" LOC=AH26;
NET "flash_cs_n_o" LOC=AG27;
NET "flash_miso_i" LOC=AH27;

NET "flash_sck_o" IOSTANDARD = "LVCMOS33";
NET "flash_mosi_o" IOSTANDARD = "LVCMOS33";
NET "flash_cs_n_o" IOSTANDARD = "LVCMOS33";
NET "flash_miso_i" IOSTANDARD = "LVCMOS33";
```

5 References

1. http://www.micron.com/parts/nor-flash/serial-nor-flash/m25p128-vme6gb - M25P series SPI Flash memory datasheet
2. http://www.ohwr.org/projects/svec/repository/ - Git repository containing this document's sources and revision history (`doc` subdirectory) and bootloading code examples (`software/sveclib` subdirectory).
3. http://www.ohwr.org/projects/svec-sw/ - SVEC Linux Device Driver
4. http://www.ohwr.org/projects/vme64x-core/ - OHWR VME64x Core Project

Appendix A System FPGA register map

Note: All registers are 32 bits-wide. Unaligned accesses, or accesses with data width other than 32 bits are **ignored**. Bits not specified in tables are not used (writes are ignored, reads return undefined values).

A.1 Memory map summary

Address	Type	Prefix	Name
0x0	REG	CSR	Control/status register
0x4	REG	BTRIGR	Bootloader Trigger Register
0x8	REG	FAR	Flash Access Register
0xc	REG	IDR	ID Register
0x10	FIFOREG	FIFO_R0	FIFO 'Bitstream FIFO' data input register 0
0x14	FIFOREG	FIFO_R1	FIFO 'Bitstream FIFO' data input register 1
0x18	REG	FIFO_CSR	FIFO 'Bitstream FIFO' control/status register

A.2 CSR - Control/status register

Bits	Access	Prefix	Default	Name
0	W/O	START	0	Start configuration
1	R/O	DONE	X	Configuration done
2	R/O	ERROR	X	Configuration error
3	R/O	BUSY	X	Loader busy
4	R/W	MSBF	0	Byte order select
5	W/O	SWRST	0	Software resest
6	W/O	EXIT	0	Exit bootloader mode
13...8	R/W	CLKDIV	0	Serial clock divider
21...14	R/O	VERSION	X	Bootloader version

Field	Description
START	write 1: starts the configuration process. write 0: no effect
DONE	read 1: the bitstream has been loaded read 0: configuration still in progress
ERROR	read 1: an error occured during the configuration (DONE/INIT_B timeout) read 0: configuration was successful
BUSY	read 1: the loader is busy (can't start configuration yet) read 0: the loader is ready to re-configure the FPGA
MSBF	write 1: MSB first (big endian host) write 0: LSB first (little endian host)
SWRST	write 1: resets the loader core write 0: no effect
EXIT	write 1: terminate bootloader mode and go passive (VME only)
CLKDIV	CCLK division ratio. CCLK frequency = F_sysclk / 2 / (CLKDIV + 1)

A.3 BTRIGR - Bootloader Trigger Register

Bits	Access	Prefix	Default	Name

7...0	W/O	BTRIGR	0	Trigger Sequence Input

Field	Description
BTRIGR	Write a sequence of 0xde, 0xad, 0xbe, 0xef, 0xca, 0xfe, 0xba, 0xbe to enter bootloader mode (VME only)

A.4 FAR - Flash Access Register

Provides direct access to the SPI flash memory containing the bitstream.

Bits	Access	Prefix	Default	Name
7...0	R/W	DATA	X	SPI Data
8	R/W	XFER	0	SPI Start Transfer
9	R/O	READY	X	SPI Ready
10	R/W	CS	0	SPI Chip Select

Field	Description
DATA	Data to be written / read to/from the flash SPI controller.
XFER	write 1: initiate an SPI transfer with an 8-bit data word taken from the DATAfield write 0: no effect
READY	read 1: Core is ready to initiate another transfer. DATA field contains the data read during previous transaction. read 0: core is busy
CS	write 1: Enable target SPI controller write 0: Disable target SPI controller

A.5 IDR - ID Register

Bits	Access	Prefix	Default	Name
31...0	R/O	IDR	X	Identification code

Field	Description
IDR	User-defined identification code (g_idr_value generic)

A.6 FIFO_R0 - FIFO 'Bitstream FIFO' data input register 0

Bits	Access	Prefix	Default	Name
1...0	W/O	XSIZE	0	Entry size
2	W/O	XLAST	0	Last xfer

Field	Description
XSIZE	Number of bytes to send (0 = 1 byte .. 3 = full 32-bit word)
XLAST	write 1: indicates the last word to be written to the FPGA

A.7 FIFO_R1 - FIFO 'Bitstream FIFO' data input register 1

Bits	Access	Prefix	Default	Name
31...0	W/O	XDATA	0	Data

Field	Description
XDATA	Subsequent words of the bitstream

A.8 FIFO_CSR - FIFO 'Bitstream FIFO' control/status register

Bits	Access	Prefix	Default	Name
16	R/O	FULL	X	FIFO full flag
17	R/O	EMPTY	X	FIFO empty flag
18	W/O	CLEAR_BUS	0	FIFO clear
7...0	R/O	USEDW	X	FIFO counter

Field	Description
full	1: FIFO 'Bitstream FIFO' is full 0: FIFO is not full
empty	1: FIFO 'Bitstream FIFO' is empty 0: FIFO is not empty
clear_bus	write 1: clears FIFO 'Bitstream FIFO write 0: no effect
usedw	Number of data records currently being stored in FIFO 'Bitstream FIFO'

Appendix B Golden Core register map

Note: All registers are 32 bits-wide. Unaligned accesses, or accesses with data width other than 32 bits are **ignored**. Bits not specified in tables are not used (writes are ignored, reads return undefined values).

B.1 Memory map summary

Address	Type	Prefix	Name
0x0	REG	CSR	Control/Status reg
0x4	REG	I2CR0	I2C bitbanged IO register for mezzanine 0
0x8	REG	I2CR1	I2C bitbanged IO register for mezzanine 1
0xc	REG	I2CR2	I2C bitbanged IO register for mezzanine 2
0x10	REG	I2CR3	I2C bitbanged IO register for mezzanine 3

B.2 CSR - Control/Status reg

Bits	Access	Prefix	Default	Name
3...0	R/O	SLOT_COUNT	X	Number of FMC slots
7...4	R/O	FMC_PRESENT	X	FMC presence line status

Field	Description
SLOT_COUNT	Number of FMC slots provided by this carrier
FMC_PRESENT	State of presence lines in the respective slots (1 = mezzanine inserted). Bit N = mezzanine (N+1).

B.3 I2CR0 - I2C bitbanged IO register for mezzanine 0

Bits	Access	Prefix	Default	Name
0	R/W	SCL_OUT	1	SCL Line out
1	R/W	SDA_OUT	1	SDA Line out
2	R/O	SCL_IN	X	SCL Line in
3	R/O	SDA_IN	X	SDA Line in

B.4 I2CR1 - I2C bitbanged IO register for mezzanine 1

Bits	Access	Prefix	Default	Name
0	R/W	SCL_OUT	1	SCL Line out
1	R/W	SDA_OUT	1	SDA Line out
2	R/O	SCL_IN	X	SCL Line in
3	R/O	SDA_IN	X	SDA Line in

B.5 I2CR2 - I2C bitbanged IO register for mezzanine 2

Bits	Access	Prefix	Default	Name
0	R/W	SCL_OUT	1	SCL Line out
1	R/W	SDA_OUT	1	SDA Line out
2	R/O	SCL_IN	X	SCL Line in
3	R/O	SDA_IN	X	SDA Line in

B.6 I2CR3 - I2C bitbanged IO register for mezzanine 3

Bits	Access	Prefix	Default	Name
0	R/W	SCL_OUT	1	SCL Line out
1	R/W	SDA_OUT	1	SDA Line out
2	R/O	SCL_IN	X	SCL Line in
3	R/O	SDA_IN	X	SDA Line in

Appendix C Important File Locations

All necessary gateware files are located in the "Releases" page of the SVEC project: http://www.ohwr.org/projects/svec/wiki/Releases. This is the **only** place where the official binaries will be published. We will not provide support for bootloader/golden bitstreams downloaded from other sources.