ACTUAL FPGAS - THE WAY OUT OF MANIFOLD HARDWARE PROBLEMS?

Authors: M. Sayed, W. Panschow
Gesellschaft für Schwerionenforschung (GSI), Darmstadt, Germany

Controls installations in accelerator facilities have to be used for much longer times than the usual innovation cycles in the electronics area. During utilization time provision of spare parts has to be ensured. Unfortunately many electronic components are no longer available after some years, so old boards often cannot be manufactured again. Moreover, the spare part problem is increased by the need to provide additional functionality when the accelerator operation is refined. Developing and manufacturing new components when more features are needed would lead to a lot of different components which have to be maintained and kept on stock. FPGA technology seems to be a solution of the service problem: Complexity and variety is moved from hardware wiring and MSI/LSI logic ICs to a programmable logic design. The same hardware platform can incorporate completely different functionality and designs can easily be extended to add features later on. At GSI more and more old designs are re-implemented using FPGA technology. Even more, commercially manufactured processor boards could be rebuilt when these boards were no longer available. FPGA technology is successfully used to reduce the number of physical different boards. Newest concepts include the ability of re-configuration remotely by the control system.

SITUATION OF GSI CONTROL INTERFACES UNTIL TODAY:

There are different parallel I/O-buses for front-end I/O-crates in use which is historical and impossible to change in a short time. The elder electronic parts of the facility in Darmstadt are now almost 20 years in use. Many I/O-cards and interfaces exist with standard and special functions, for example Gate Pulse Generators listening to the distributed timing events for 'tailoring' variable and remote controllable trigger pulses often used for special beam diagnosis hardware, Timing Sequencers are used to generate sequences of timing event signals to be propagated over the whole facility by the control system to initiate dedicated actions at the right time. Timing Event Decoders are the receiver units for the events being distributed to generate local trigger signals. This function is part of the Gate Pulse Generators as well but without front panel operation option. Field Bus Interfaces with on-board function generator macros used to generate Signals with a high output data rate without charging the field bus and the real-time CPU with DAC-data too much. This feature is of interest for ramped dipole magnets resp. their power supply controls. 64 Bit-Parallel I/O-Cards with additional logic function macros for special beam-chopper interlock which can be chosen by a configuration switch without changing firmware and of course, some ADC/DAC-cards +/-10V, 16bit at 50kS/s, etc. or 16bit at 1MS/s including averaging modes and different trigger modes. And many 'exotic' cards, some of them exist only 2 or three times.

Different proprietary parallel buses still have to be supported and 15-year old VME Slave CPUs are being exchanged step by step by a GSI development which includes a FPGA to replace all obsolete peripheral chips used on the old CPU board to prolong control system life without changing software for 300 Slave CPUs with many different device software versions too much.

COMMON PROPERTIES OF EXISTING HARDWARE DESIGNS:

More than 85% (estimation) of hardware functionality are identical, if it can be represented by logic resources with clock rates up to 300MHz (nowadays to be titled as general purpose), they almost all have a parallel bus interface to communicate with a field bus gateway over one of the proprietary GSI-parallel buses, and varying user I/O which often is TTL-level logic on the other side. The former classification of functionality depending which TTL-MSI-functions were included in the design as counters, decoders, comparators, delay-timers and so on, are obsolete because of the abilities of actual FPGAs with ten-thousands of macro cells. It makes no much sense to distinguish between counters, general purpose I/O, decoder functions or even complete hard-real-time control loops written in VHDL code, as long as the FPGA-on-chip-resources are no

limitation. So there is almost no influence of the chosen FPGA to the possible primary functions of the board as long as it is pure logic and the chip has enough of it. Only at the high-end concerning dedicated silicon included in FPGAs (big Memory blocks, Signal Processors, differential lines, Gbit- serialisers) required functionality and chosen parts have to be thoroughly matched first.

'REAL' DIFFERENCES BETWEEN HARDWARE DESIGNS:

Special level logic (ECL, TTL, LVTTL, NIM, usually with 50Ohm cable impedance, rugged PLC-24Volts...) are usually depending on the kind of hardware used from the different accelerator and experimental workgroups, analogue input/outputs with different sample rates, resolutions and voltage levels do not seem to get standardised so easily. The same with high speed triggers, RF-circuits and/or –multiplexers. They are not subject of standardisation so far, maybe they will never be. But different pin-outs for user-I/O occur even with similar functions (In fact, there is no strict layout rule for these connections). So, the wish for having fewer boundaries for the direction of I/O-lines leads to the extreme requirement of bit-by-bit programmable I/O-pins.

Operation modes, communication and signalling might be very different, so at least different firmware versions are necessary or, in case of newer designs, logic macro functions can be chosen by configuration switches on the card. Connectors and electromechanical details always make trouble- and the question comes up: where to connect, at the rear or front side of the cabinet or crate? So, there is some variation to be considered as a part of the requirements users in-house might have.

ESTABLISHED WAYS TO REDUCE HARDWARE VARIETY:

Classical solution in case of firmware variations:

Pre-programmed hardware with identical layout representing several functional variations has to be kept on stock even on different locations of GSI. These boards *have to be labelled for identification*. So there is still effort for the versions used in operation even though the newer hardware is programmable on board. At least handling of programmed chips and de-socketing tools are necessary almost no more, which often was the case at GSI until the mid-1990s.

Electromechanical view on I/O-cards and crates:

Connectors to the 'real-world' can be made as adaptors between standardised user-I/O-pins (internal, on rear side of a backplane) and front/rear-side of the rack with user-required connectors (LEMO coaxial, LEMO biaxial, DSUB 9, 15, 25, 37.., 3M ribbon cable connector, screw-mount, BNC, ...) So it is possible to re-use them in the next project. *Keep specific connectors out of complex but in fact general purpose cards if ever possible!* Learn from professionals (like NI!) They do not fulfil any wishes of this kind and do not give much help to solve these problems with their I/O-cards.

At GSI standard I/O-cards with 32 bit and 64bit density, 16bit-wide organized had been developed in the years after 1998. The pin layout of these general-purpose cards was the de-facto-standard for later developments of special functions which needed some adaptation to dedicated hardware. This step to standardisation helped a lot to get rid of logic cards with non-reusable pin layout or signal mix.

Primitive I/O-functions including physical drivers have to be kept simple!

By progresses in miniaturisation level-adaptation and galvanic insulation might be integrated with logic at the same card as long as it is common enough to be used in different applications and can be combined with the same connector – adaptors like other I/O-cards. This rule, besides the hardware progresses due to higher integration and functionality of FPGAs, is the basis of a real 'unitized construction system' of I/O-interface components.

Macro functions in more powerful FPGAs (ALTERA 1K, 10K, 20K-Series)

By using the huge resources of FPGAs it is possible to bring in more than one complete functionality into the same chip. This way was chosen in developments at GSI after the year 2000. The basis of this development was the 32bit-I/O-card pin layout which was extended with a 5-row

VME64 card-connector in a backward-compatible way. The function to be activated is chosen by a configuration switch. It is a quite comfortable solution for adding 'exotic' functions into a general purpose I/O-card without needing different hardware versions on stock, labelling, etc.

In case of adding more macros, it is recommended to *update all cards* at least those on stock, better even those already in operation to avoid confusion. For tracking the state of firmware actually loaded in operational hardware, it is essential to have at least one version register on every card. Every macro function should have its own revision state to be checked by control system software. This logic macro design methodology leads to additional multiplexing structures which reduce the maximum clock speed of the design because of additional paths that are only necessary for choosing one of the macro functions, having broader combinatorial logic, etc. And there will be the point even the big resources of these chips cannot fulfil all requirements at the same time. So, what's to do then? This leads to some thoughts considering the impressive properties of the actual FPGA families and the experiences concerning necessary standardisation of inner-system pin-layouts and adaptors.

WAYS OUT OF DIVERSITY:

User-I/O flexibility to be improved by FPGAs

I/O-cells of modern FPGAs are rugged enough to be used without additional bus drivers in many cases. So, there is no byte- or word-wide fixing of data direction at the user-I/Os necessary anymore. Nevertheless, it is recommended to define input and output channels at least byte-wide for better overview avoiding mistakes and easier diagnosis. For different internal bus layouts with a little luck the same card can understand more than one bus at least as long as the power supply pins are not different between these bus definitions. By switching the direction of I/O-pins connected to the bus as needed in the system, the same hardware design can replace old hardware of more than one bus system and helps to reduce hardware reproduction problems. This way was chosen in recent GSI developments to increase the potential for replacing hardware in different environments.

Hot-plugging ability for easier service

Sometimes FPGAs provide even more: Hot-plugging without damaging the logic is possible with newer FPGA families. Hardware defects caused by changing cards without switching off the cabinet's power supply will belong to the past. Old TTL-logic often withstood this kind of handling, but CMOS-based designs definitely did not. Disadvantage is the intolerance to 5Volt-levels of actual FPGA-families. In case of mixing up new 3.3Volt with old 5Volt hardware, Bus Switches like 74CB3T-series are necessary to keep the mentioned advantage of bit-by-bit direction programming. The driving ability of the I/O cells is often sufficient, if not the highest speed is necessary on a backplane with low impedance.

Firmware-related problems to be solved:

By usage of PLD/FPGA logic, functions of interest can be completely standardised if the pin layout of the cards is as strictly defined as possible. But this leads to another problem coming up:

How to manage firmware versions and variations? First: firmware versions of PLDs and FPGAs have to be kept in central data storage (server with backup or, maybe better, a database) to avoid data loss and confusion about valid releases. This kind of problem is well known by software engineers, but sometimes not by hardware people. But: Who knows *how* to 'burn' *the right* firmware files into the hardware in case of trouble? Procedures have to be invented here, service people to be informed and trained... The more programmable hardware has to be maintained in the facility from service personnel without special education on this field, the more this issue comes into focus. Any possibility to take away responsibility for this from the service personnel the easier it will be to keep a larger facility in operation. This leads to the vision of a more automated handling of firmware in front-end electronics.

ACTUAL DEVELOPMENT STEP AT GSI:

Goal of a recent hardware development was to avoid the disadvantages of macro functions and the effort to change firmware by taking out hardware of the control system to reprogram it at the desk or go to every place the hardware is used to reprogram it which is big effort when the facility gets larger and needs more electronic devices to be maintained, as GSI plans for the FAIR project which will at least double the interface count and bring about three times more average distance between the interfaces and the service people.

The In-System (Re-) Programmability of general purpose I/O-hardware is the choice to avoid problems that can be solved remotely. Functionality will brought in right at the place of operation and has to be transmitted over the usual data path of the control system. Before this can be safely done, some considerations have to be made:

Requirements for Reprogrammability of universal I/O-cards

The access to the programming feature should use the same methods and paths as normal communication to the front-end I/O does. The status of transmission has to be controllable by reading status information. The content of the programmed configuration memory area can be read back.

A failure of reprogramming the hardware in the field should never lead to a complete functional failure. This means a corrupted programming file has to be identified and never will be loaded from the configuration device to the SRAM cells of the FPGA or will not be activated if loaded. It is necessary to be able to program the card once more even in case of a load failure.

Design

If during the loading process a power down or other malfunctions never occurred, no dedicated loader hardware or additional independent memory might be necessary and could be implemented in the reprogrammed chip itself. But this definitely cannot be excluded and would cause never-come-back situations. Errors can occur during transmission by having errors on the network, a power down somewhere, cabling trouble or maybe strong noise (high voltage discharges) which could lead to malfunction of field hardware.

So, concepts for reprogramming have to include a loader functionality which cannot be lost. Fail-safe configuration data will be activated after downloaded data are not recognized as a valid data set . In case of a microprocessor board, program flash memory can include the configuration data. The loader chip has to include a non-volatile bus interface for communication to get data for programming. I/O-hardware of the control system at GSI usually does not include microprocessors on every board, so this way is not possible.

Basic Design Decisions

One way to ensure start up even after a corrupted download and some hardware malfunction, is a basic (fall-back) firmware needed for safe communication even in case of hardware or system software trouble.

At first, a data path independent from the primary (fail-safe) configuration chip to the FPGA has to be implemented. A common solution uses the JTAG-Interface which is included in many complex chips. There are already some applications using this approach This path can be interpreted as a second interface to program the SRAM cells of the FPGA and the interesting point is that this interface can be activated instead the usual configuration at power -up. At this moment, we get a possibility to influence whether the FPGA gets its firmware from the configuration chip directly connected to the FPGA, or from the JTAG interface. The communication to this JTAG interface can be done by some additional, independent logic, which might be realised in a CPLD (MAX3000-Series). Some concepts (like some processor boards with FPGAs and additional loader logic) only use the JTAG-interface and abandon the usual configuration chip.

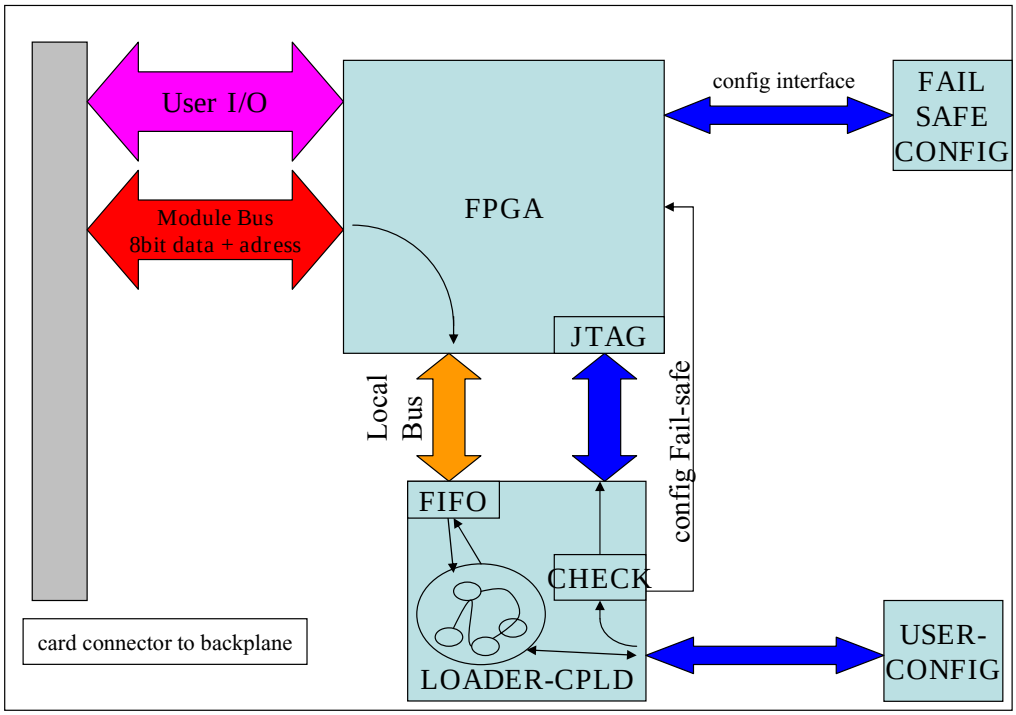

Figure 1: Block Scheme of Reconfigurable Logic Board with Fail-Safe Configuration

Loader functions

If the programming path over JTAG is used by the additional CPLD, this loader logic has following tasks:

- Use a dedicated additional interface to connect with the system bus interface over the FPGA to get data from the control system. So the interface to the system bus is not doubled as it was done in designs of VME boards with on board signal processor developed at GSI.
- Take received data packets from the system bus during a download procedure and send them to a configuration chip, which will store the user-configuration.
- Activate JTAG-interface after download and transmit data (user firmware) to the SRAM-area in the FPGA by implementing a loader mechanism (conditionally, after check, see below)
- Check consistency of user-firmware before this configuration will be activated.
- In case of failure (checks not successful) the JTAG-interface has to be released and the fall-back configuration ("fail-safe") in the configuration-chip connected to the FPGA itself is activated by forcing the FPGA to reconfigure over by pulling down the fail-safe control line to load the FPGA from the dedicated configuration-interface.
- The board can be forced to load the failsafe configuration anyhow by a jumper or by system access to a control register.

Loader software and access to the loader hardware

To communicate with the front-end electronics, there are some pre-definitions needed. The complete loading process will be controlled by the system software. There is some granularity of data transmission, 256 bytes of data which is depending on the block size of the used flash–memory based configuration chips. These blocks are serially written to this chip and need some time to be 'burned'. The end of this time is observed by the programming software. Basic functions

are: Reading the status of the loader including which configuration is actually loaded, errors which can occur during access to the FIFO memory in the loader chip, if programming of the last sent block was completed, and so on.

CONCLUSION AND OUTLOOK

The causes for having a plenty of different hardware in use and further developing specialised electronics have been discussed. By using modern, high density FPGAs with JTAG-interface including new hardware and software concepts lead to another level of flexibility of hardware usage and re-use. Software control over functionality of front-end I/O-electronics can bring a new quality into control systems .On one hand, this will change the manner of hardware service completely and on the other hand initialisation handling on the software side including questions concerning diagnosis. By inventing more strictly rules of version or even source code control which are well-known in software engineering, a quality step in handling firmware versions is in sight and will help to overcome problems of unclear hardware status in the field.

Combined with a consequent step to standard-pin-layouts of digital logic boards and keeping out user-specific connectors and interfaces, it is possible to avoid special hardware development with new circuits and layouts in at least quite a lot of cases or reduce this to some adaptor development. New functionality really is added just by firmware development on identical boards. By this strategy, the risks of more complex hardware redesigns and layout changes are taken out of projects that need only a few pieces of a special interface.

First steps are done, but there is still much work to do on the side of system software addressing the different hierarchy levels including data bank design and a new workflow for firmware and software development staff. Hardware and firmware versions and revisions no longer are 'black magic' and will be subject to exchange information between HW and SW developers in a more cooperative and better documented manner. So, for some people it seems remote reconfiguration to be even a kind of 'cultural revolution' instead of being 'just another hardware feature'.