Mapping Multi-Million Gate SoCs on FPGAs: Industrial Methodology and Experience

H. Krupnova

CMG/FMVG, ST Microelectronics
Grenoble, France
Helena.Krupnova@st.com

Abstract

Today, having a fast hardware platform for SoC software development prior to silicon is an important challenge to gain the time-to-market. The FPGAs offer an excellent prototyping basis for building hardware platforms since more than ten years ([1]). However, as the circuit complexity increases and project timeframes shrink, building a multi-FPGA prototype represents a real challenge from the complexity viewpoint. The paper describes the state-of-the-art mapping methodology, prototyping tools and flows, shows the most difficult mapping problems and the ways to overcome them. The paper is issued from the experience of mapping on FPGA platform of four latest highly complex ST Microelectronics SoCs ranging from 1.5 to 4 million real ASIC gates mapped to up to 9 highest capacity FPGAs.

1. Introduction

As the complexity of digital systems increases, the amount of the embedded application software increases even faster. To meet the time-to-market requirement and to reduce the number of silicon cuts, the software teams would like to start the software development as early as possible. The FPGA-based prototyping ([2], [5], [6], [7], [9]) allows building the hardware platform before the silicon becomes available.

Starting the software development in advance allows saving weeks and even months. In addition, very difficult hardware bugs may be detected only when running the real software with real applications. Thus, there is an increasing interest from the design teams to hardware prototyping technologies.

When the basic functional block development and IP integration sufficiently advances, the design team becomes able to produce the first assembled backbone (Figure 1). Usually it includes the processor(s), memories, memory controllers and the system bus. This backbone is delivered to the verification team and is ready to be ported to emulators ([11]) to debug the hardware. Once the hardware is working and stable, the mapping on the FPGA platform targeted for the software validation ([3]) can start.

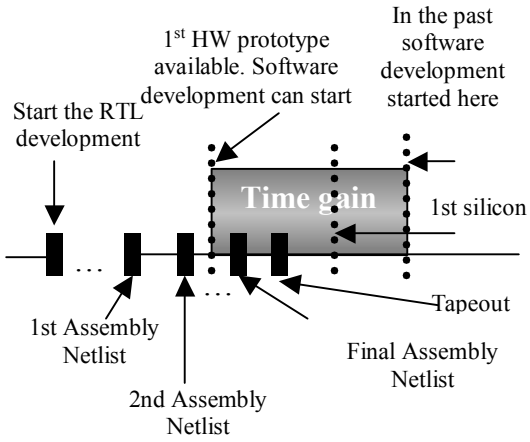

Fig. 1: FPGA prototyping timeframes

There are several validation iterations before the first tape-out. During each iteration, more IPs are added to the backbone. The design is thus validated incrementally. Building the FPGA prototyping platform follows the same schema. The objective is to have a usable FPGA platform early enough before the first silicon and in the same time containing the mature design attractive for the software development teams. From one side, the software teams want the design mapped on FPGAs to be mature enough (the tape-out version is the ideal one). From the other side, the FPGA prototype has to be ready as early as

1530-1591/04 $20.00 (c) 2004 IEEE

possible: weeks and even months before the tape-out. The FPGA prototyping team has to cope with this contradiction. The best way to satisfy both requirements is to have an extremely fast FPGA mapping flow.

The challenge today is the increasing complexity of the SoC projects: more than one core, several big memories, about ten main IP blocks developed in different world locations, ten to twenty or even more clocks (without counting the gated clocks), several dozens of small-size memories, etc. Mapping such a system on an FPGA platform (different technology, different clocking resources, etc.) requires more and more time. However, the challenge is to do this mapping in a very short time, just a few weeks and even few days.

If the fast FPGA prototype is not available in time, the software team can continue to use the emulator ([11]) for software development. In the beginning, the emulator is the only one available hardware platform. The disadvantage of using an emulator for software development is relatively low speed (10 to 100 times slower than an FPGA prototype) and availability: the user time on emulator is expensive and the machine is usually shared between several projects and several design teams.

The amount of practical data about the latest prototyping projects collected in this paper provides a view into the industrial FPGA prototyping challenges, methods and platforms.

2. The design data

The four case studies presented in this paper are listed in Table 1. They are the ST Microelectronics CMG projects called A, B, C and D. Design C is represented twice: C-1 corresponds to the first iteration; C-2 corresponds to the second iteration. The Tables 1-4 present a statistical data about the FPGA platforms created for these SoCs. The chip A is a digital TV processor and is already available on the market. The silicon for the chip B (a set top box system) already exists. The tape-out is already done for the chip D (a DVD chip). The tape-out for the chip C (a set top box system) is planned by the end of the year. All these chips contain the video processing blocks and some of them can also process the audio.

The Table 1 presents the estimated size in number of ASIC gates without counting the memories, the number and type of FPGAs that were used to build the most recent hardware platform for this chip, the processor cores and big size memories. The prototyping for the chip A was performed using Xilinx VirtexE FPGAs XCV2000 having the approximate ASIC gate capacity of 250K gates. The prototyping for the other SoCs was done using the Xilinx Virtex2 XC2V6000 ([12]) chips (approximate ASIC gate capacity 450K gates). The first assembled version of the chip C (C-1) uses 4 XC2V6000 FPGAs. The latest mapped version of the chip C (C-2) uses 7 XC2V6000 FPGAs and 2 XC2V8000. For the designs A, B and D, the data presented below corresponds to the latest mapped version.

	Size in ASIC gates (no memory)	Number of FPGAs	Embedded Cores	External Memories
Design A	1.5 Mln	6 XCV2000	ST20C2C200	4Mx16 SRAM
Design B	4 Mln	8 XC2V6000	ST40-103, ST20C2C201	2 x 4Mx32 DDRs
Design C-1	1.5 Mln	4 XC2V6000	ST20C2C201	8Mx16 DDR
Design C-2	4 Mln	7 XC2V6000 + 2 XC2V8000	ST20C2C201	8Mx16 DDR
Design D	3 Mln	7 XC2V6000	ST20C104, ST220	4Mx16 DDR

Tab. 1: Prototyping projects complexity

	Average FPGA filling	Maximal FPGA filling	Size of the biggest IP	Number of the top-level pins
Design A	70%	94%	91% of the XCV2000	3350
Design B	84%	99%	140% of the XC2V6000	8100
Design C-1	63%	99%	53% of the XC2V6000	2500
Design C-2	82%	93%	115% of the XC2V6000	8600
Design D	84%	99%	99% of the XC2V6000	6100

Tab. 2: FPGA prototype complexity

	System Frequency	Number of clocks used in more than 1 FPGA	Number of gated clocks	Maximal number of clocks on one FPGA
Design A	1.5MHz	5	~ 200	6
Design B	2MHz	7	~ 800	14
Design C-1	2MHz	4	~ 150	10
Design C-2	1MHz	4	~300	14
Design D	1MHz	6	~ 300	11

Tab. 3: FPGA prototype clocking data

	Fastest clock frequency	Number of mapping iterations	Complete design or a sub-system mapped	Pin multiplexing ratio
Design A	10MHz	3	Complete design	4x1
Design B	24MHz	3	Sub-system	4x1
Design C-1	24MHz	1	Sub-system	4x1
Design C-2	12 MHz	2	Complete design	4x1
Design D	12MHz	2	Sub-system	4x1

Tab. 4: Prototyping project data

Two of four presented SoCs contain each two processors. The average FPGA filling is ranging from 60% to more than 80%. The maximal FPGA filling goes up to 99%. Fitting the corresponding IP block inside the FPGA was in this case a real issue. In most cases, the size of the biggest IP did not exceed the FPGA size. However the design B contained the ST40 core, which was split on 2 XC2V6000 FPGAs and one of the video blocks demanding 120% of the XC2V6000 FPGA. This block was finally not mapped on FPGA and kept as dummy. In general, an IP block fills more than a half of the FPGA size. The ideal partitioning strategy is to put one (or more if the IPs have common clocks and communications) IP per FPGA. Splitting an IP block in several FPGAs is not desired because of the clocking, speed and complexity reasons.

All presented ST SoCs are based on the ST Bus architecture. The ST Bus is a configurable bus that is generated by the dedicated internal tools and is tailored for the application. It contains the protocol handlers, size converters and big crossbars for the communication. Having numerous advantages from the system viewpoint, for FPGA mapping the ST Bus is one of the difficult issues. Being based on crossbar logic, it introduces a huge number of nets in the top netlist (up to 10000 for the most complex SoCs). On the opposite, the maximal number of I/Os on the latest FPGAs is 1104 ([12]). Handling the ST Bus is thus the basic partitioning difficulty when mapping the ST designs. The FPGA implementation is not feasible without using the pin multiplexing. Because of the pin multiplexing, the system speed is divided and is situated between 1 and 10 MHz.

The presented complex SoCs require about 10 to 20 clocks. The FPGA prototype may require additional clocks to implement the pin multiplexing, make working the complex memories, model the special structures on FPGAs, etc. Due to the limited number of clock resources inside the FPGAs and on the board, the prototyping engineer has to simplify and if possible merge the clock domains. In the worst case he may have a situation when there are more than twenty clocks on the same FPGA while the FPGA has in total 16 global clock lines (and the place and route problems may start when exceeding 8 global lines) and 24 low skew lines with lower quality. The maximal number of clocks on one FPGA presented in Table 3 corresponds to the design after all possible optimizations, clock line merges, etc. The fastest clock frequency corresponds to the multiplexing or other “service” clock.

The presented data gives an idea about the complexity of building the FPGA implementations for the latest SoCs. Due to the huge complexity there is no time to build the custom FPGA board for these designs. The size of some IPs may exceed the biggest FPGA size, the demand in clock resources may exceed the FPGA resources and the required pin number exceeds the maximal FPGA pin number. Running the FPGA synthesis and place and route is also extremely time consuming. In addition, due to the presence of gated clocks and non-mappable ASIC structures the designs usually don’t work when running the first tests on the FPGA platform. They require to be debugged to make the FPGA implementation working. Mapping such SoC

designs is also highly stressful for the whole chain of the involved FPGA tools and platforms.

3. The FPGA platform

The Aptix System Explorer MP4 platform ([3]) was selected to target the described above requirements. This platform may host up to 12-14 FPGAs. As soon as the new generation of FPGAs is available, the platform can be upgraded with new FPGA modules. The MP4 board contains 4 crossbars – FPICs - that realize the programmable interconnect between the FPGAs. The total number of FPGA pins that may be routed by the crossbars is 2880. The MP4 board is based on folded clos architecture: when mounted, each FPGA is connected to all 4 FPICs. The FPICs have connections to FPGAs and between them. The board is always routable. Since the introduction of the Flex FPGA modules ([4]) it is also possible to realize the direct connections between the FPGAs. This makes possible to use up to 1100 FPGA I/Os. Because the connections are direct, they don't require pins from the free-hole area, thus increasing the board capacity and allowing use up to ten FPGAs. The system is really open and flexible. By changing the number of FPGAs mounted on the board and changing the way they are connected, it allows building dedicated FPGA architecture for each design (see Figures 2-6). In addition, custom interface and memory boards may be mounted on top of each FPGA module, allowing connections with real hardware or use of big memories. Figures 2-6 represent the FPGA architecture for each of four presented above SoCs. Design A was mapped before Aptix created its Flex modules ([4]) allowing FPGA-to-FPGA connections. All the other circuits were mapped using the Flex modules.

The MP4 board view containing a common configuration for both C and D SoCs is presented in Figure 7. Both designs use the external interface with ST Microconnect box allowing the interface with workstation-running application software through the Internet. Both designs have the DDR memories emulated using external SRAM modules.

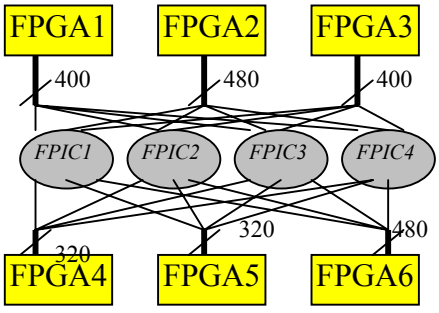

Fig. 2: FPGA board configuration for the design A

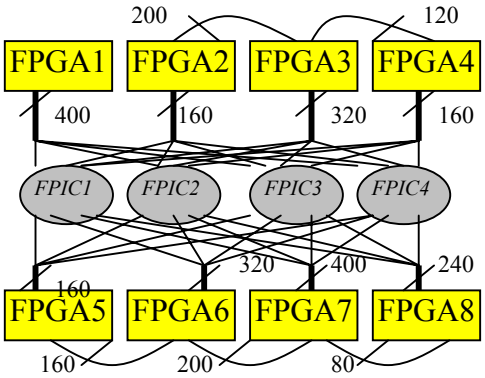

Fig. 3: FPGA board configuration for the design B

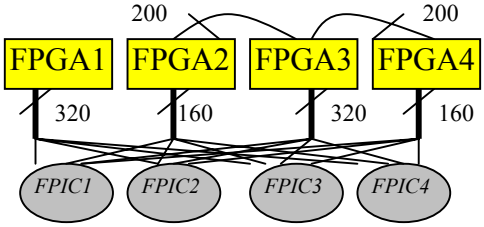

Fig. 4: FPGA board configuration for the design C-1

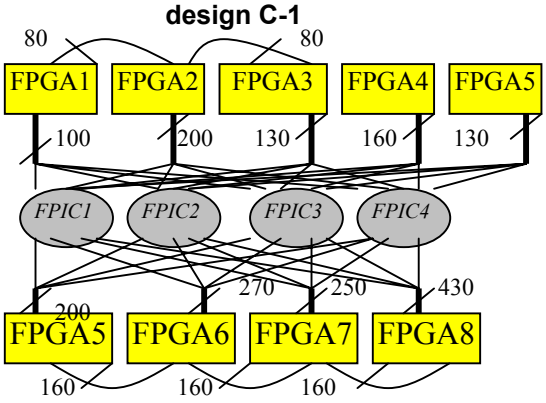

Fig. 5: FPGA board configuration for the design C-2

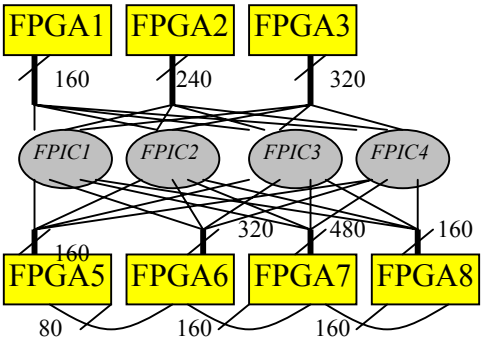

Fig. 6: FPGA board configuration for the design D

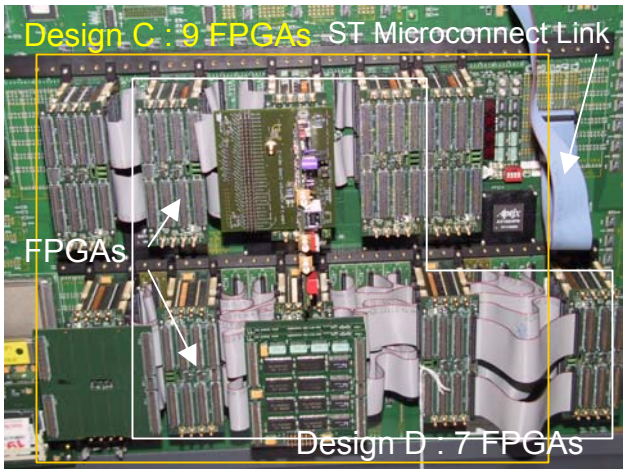

Fig. 7: Aptix board view for C and D designs

To debug the design, the MP4 platform allows the direct visibility (using the HP logic analyzer) of all the signals routed by FPICs. In addition, it is possible to see the internal FPGA signals using the Xilinx incremental probe routing capability supported by Aptix Explorer software.

4. The mapping flow

The mandatory conditions to start the FPGA prototyping are the following.

1) Each IP block was validated in simulation.
2) The assembled top design was validated on the emulator.
3) The synthesizable test bench is available and validated in emulation.

The FPGA mapping can be initiated when the design hardware is stable and operational. The mapping time usually consists of the mapping itself and the debugging and running the tests on the board. A number of mapping steps presented below reflects the complexity of the mapping process.

1) Mapping on Xilinx Virtex2 technology is done by specifying Xilinx as a target library inside the Synopsys DC. The reason to use the DC synthesis is its closeness to the ASIC implementation flow, thus avoiding all kind of synthesis problems. In addition, the design team often delivers the design to the verification team in internal Synopsys gate format (Gtech or Corelib).
2) There is a number of modifications that are required for the FPGA netlist: customizing the testbench to Aptix platform, introducing clock generation and reset logic, building the clock distribution on the FPGA platform, changing the non-mappable structures and gated clocks, etc. These modifications are performed using the Synopsys dc_shell. Script utilization is vital to be able to iterate during the mapping and the debugging process: it guarantees repeatability, eliminates human errors and allows reuse between different design versions and even different designs.
3) The FPGA mapping may also be done using the Synplicity Certify tool ([8]). It is used for FPGA synthesis, removing the gated clocks and design partitioning. The synthesis starting from RTL produces up to 40% improvement in area comparing to DC synthesis from Gtech. Synplicity synthesis is used for most critical IP blocks and in all cases when there is a problem with fitting a block into an FPGA. The RTL is requested from the design team to use the Certify.
4) The netlist viewing and analysis is performed using the Debussy tool from Novas ([10]). It makes possible analysis of the clock domains, gated clock structures, logic cones during the debugging, etc.
5) The ASIC memories need to be specifically modeled for the FPGA technology. Xilinx Coregen tool is used to build the Xilinx memories. In addition to modeling the low-level memory, the user needs to design and validate the memory wrapper.
6) The FPGA place and route is performed using the Xilinx ISE tool. It is encapsulated by the Aptix Explorer tool, which does the board place and route. The FPGA place & route jobs are dispatched to multiple workstations using the LSF tool. The place and route tool is also used in standalone to clean the FPGA place and route problems, to estimate the block sizes, to check the presence of the gated clocks and clean the clock problems.
7) The FPGA pin multiplexing is performed using Aptix multiplexing tool.
8) Aptix Logic AggreGater tool is used for multi-FPGA partitioning.
9) Celaro emulator from Mentor Graphics ([11]) is used as a reference for debugging. The design mapped on emulator is the closest to the FPGA prototype and is preferable than simulation.

The FPGA mapping flow is presented in Figure 8. Putting all steps together represents a tedious amount of work, which requires a significant amount of time, which is in contradiction with the requirement to build the FPGA platform in a short time.

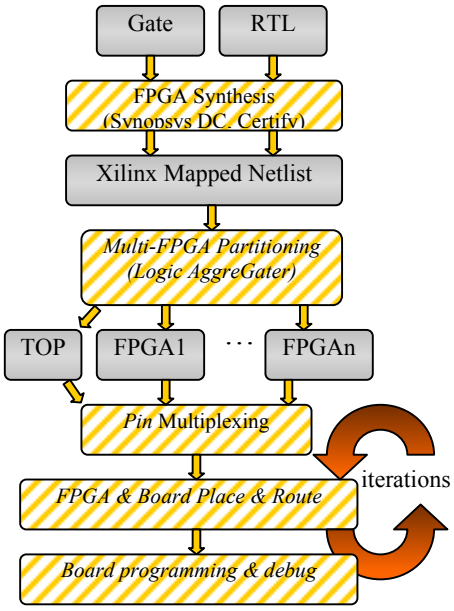

Fig. 8: The mapping flow

The most critical issues when mapping on FPGAs are related to timing and partitioning. The FPGA technology is different from the ASIC technology and this highly impacts clocking. The FPGA interconnect delays are not predictable and there are glitches. This makes it very hard to implement the correct functionality of the gated clocks. Unfortunately there are more and more of them in the designs. A significant part of gated clocks may be cleaned automatically using for example Certify tool. However, the designer still needs to clearly understand the clock gating to guide the "de-gating" process. Finally, there always exist special cases that must be arranged manually (divided clocks, removing of scan clock multiplexers, etc). In addition, ASIC circuits may contain structures impossible to map correctly on FPGAs that require to be modified. An example is presented in Figure 9. The signal OUT will have glitches near both clock edges. If it is registered using the same clock signal, there is a high probability that a wrong value will be registered.

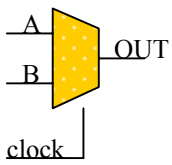

Fig. 9: Structure not mappable on FPGA

Today, introducing more debugging support and automatic cleaning of timing problems is made at a cost of reducing the platform speed (the Celaro emulators may be taken as an example). A lot of work must be done manually to build the fast FPGA platform exploiting the full advantages of the modern FPGA technology. Roughly, the tradeoff is: if a fast mapping is desired, the platform will be slow. If a fast platform is desired the mapping will be slow.

5. Conclusions

To cope with the SoC complexity, the FPGA mapping tools and platforms must evolve to provide better debugging capabilities, more powerful clock resources, clock analysis features, automatic clock mapping, knowledge bases enumerating the existing problems and solutions, design rule checkers, smart design partitioners taking into account clock resources, common test benches and constraint file formats for different hardware platforms, etc. The "ideal" future prototyping platform would be one, which combines the speed advantage of the modern FPGA technology with the ease of mapping offered by the big emulation platforms.

6. References

[1] S. Hauck: The Roles of FPGAs in Reprogrammable Systems. Proc. Of the IEEE, Vol. 86, No. 4 (1998): 615-639.
[2] http://www.aptix.com
[3] System Explorer MP4 Reference Guide, Aptix, 1999
[4] Aptix Product Datasheet : Xilinx Virtex-II FPGA Module, May 2002
[5] http://www.xilinx.com
[6] http://www.prodesigncad.de/chipit.htm
[7] http://www.eve-team.com
[8] http://www.synplicity.com/products/certify/index.html
[9] M. Pavesi: Modern FPGA capabilities made available to the FlexBench modular rapid prototyping platform. Proc. DATE 2002.
[10] http://www.novas.com/Products/Debussy
[11] http://www.mentor.com/celaro
[12] Xilinx, "Virtex-II Platform FPGA Handbook", December 2000