Systematic Considerations for the Application of FPGAs in Industrial Applications

Falk Salewski
Embedded Software Laboratory
RWTH Aachen University, Germany
salewski@informatik.rwth-aachen.de

Adam Taylor
Processor Products Group
EADS Astrium, UK
adamp.taylor@astrium.eads.net

Abstract— **From the functionality point of view, FPGAs became very interesting for industrial applications. Reasons are the constantly decreasing costs of microelectronics and improvements in the corresponding design tools as well as the increasing need of complex real-time functionalities in these applications. However, other non-functional requirements have to be considered. Therefore, the potentials of FPGAs for industrial applications are considered in this paper on basis of *hardware attributes* representing the contribution of these systems to *system qualities* as for example performance, reliability and marketability.**

I. Introduction

Field Programmable Gate Array (FPGA) technology has seen significant development and investment since their invention more than twenty years ago. Besides improvements in their generic structure and their corresponding design tools, both will be described briefly in the following, dedicated logic blocks as multipliers or block RAM have been added to their architectures allowing more efficient designs.

The basic building block of the FPGA is the logic cell which typically consists of a Look Up Table implementing combinatorial logic functions, a D Type Flip Flop and an output multiplexer allowing the logic cell output to be either synchronous or combinatorial. Along with these logic cells, configurable I/O blocks exist implementing the interface of the logic design to the outside world. All these blocks are interconnected by a programmable interconnection network. While the basic logic cells may be similar across devices there are a number of different technologies used to implement the physical device. Typically these physical devices can be classed as either reprogrammable or one time programmable (OTP). Reprogrammable FPGAs are SRAM or FLASH based with SRAM based devices requiring configuration at power up from a non volatile memory. One time programmable devices are based upon Fuse / Antifuse or EPROM technology and once programmed their contents cannot be changed or modified. Further information on the architecture of FPGAs can be found in e.g. [19], [23], [24] and on the web pages[1] of the three major FPGA companies Xilinx, Altera, and Actel.

FPGA designs are typically described in one of two high level languages (Verilog or VHDL) collectively called hardware description languages (HDL). Following completion of design entry the HDL requires synthesis. Synthesis will attempt to extract the logic described within the HDL and map this to the target technology primitives (Multipliers, RAM blocks, I/O, Flip Flops, combinatorial logic). Synthesis tools are provided by both FPGA manufacturers and third parties. The final stage of implementing the design is to use the manufacturers' place and route software which takes the logic resources and connections as defined by the synthesis tool and attempts to place these into the available FPGA resources while meeting the user commanded timing and performance constraints.

FPGAs differ significantly from other hardware platforms used in embedded systems, as microcontrollers (MCU) and digital signal processors (DSP), with respect to their internal structure, the design languages, and the available tools. However, any given functionality can be implemented in principle on FPGAs as well as MCUs or DSPs. In fact, several promising applications of FPGAs are already known in the industrial domain (a collection of applications can be found in [19]). For the system designer, now having the choice between several devices, the choice for a certain platform depends on several aspects. The most obvious ones are the performance and the costs of a certain device, but several other aspects have to be considered, as will be presented in the following section.

A decision for the use of FPGAs in a specific application is often not easy, especially if up to now FPGAs are not used widely in the corresponding domain (as in industrial applications). The aim of this paper is to provide a systematic approach for evaluating the advantages and disadvantages of FPGAs for a specific application. This approach will be introduced briefly in the following section II and known properties of FPGAs are surveyed with respect to their impacts on system qualities in section III. After a brief summary of specific properties of reconfigurable SoC based on FPGAs in section IV and final remarks on FPGA design in section V, a conclusion is given in section VI.

II. Systematic Hardware Selection

A systematic approach of hardware platform selection has been developed for embedded systems education at our institute [26] and a corresponding interactive web system called SHPS has been built up that is accessible online[2]. The approach of systematic hardware platform selection is based on two steps presented in Fig. 1. First, *hardware attributes*

[1]www.xilinx.com, www.altera.com, www.actel.com

[2]SHPS URL: www-i11.informatik.rwth-aachen.de/SHPS/

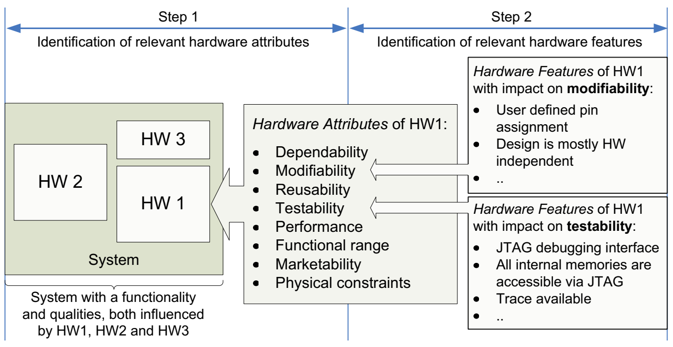
Step 1
Identification of relevant hardware attributes
Step 2
Identification of relevant hardware features
HW 3
HW 2
HW 1
System
System with a functionality and qualities, both influenced by HW1, HW2 and HW3
Hardware Attributes of HW1:
Dependability
Modifiability
Reusability
Testability
Performance
Functional range
Marketability
Physical constraints
Hardware Features of HW1 with impact on modifiability:
User defined pin assignment
Design is mostly HW independent
..
Hardware Features of HW1 with impact on testability:
JTAG debugging interface
All internal memories are accessible via JTAG
Trace available
..

Fig. 1. Approach to systematically identify impacts of hardware platform selection on the overall system

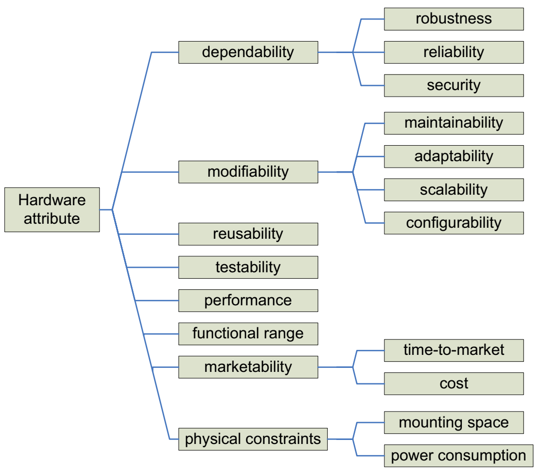

Fig. 2. The hardware attribute tree

of a specific hardware platform are identified and evaluated (left part of Fig. 1). They represent how much a given platform contributes to the corresponding system qualities (e.g. the *hardware attribute testability* of a specific hardware platform represents how this platform contributes to the *system quality testability*). The hardware attributes considered in our approach are depicted in Fig. 2 in form of an attribute tree.

Second, all hardware features[3] that have a possible impact on the different attributes have to be considered to allow an evaluation of the hardware attributes for a specific hardware platform (right part of Fig. 1). As an example, the existence of a JTAG debugging interface (hardware feature) has an impact on the testability of the corresponding hardware platform (hardware attribute). The hardware attribute tree and the relationship between *hardware attributes* and *hardware features* is described in more detail in [26].

Based on the result of this two step approach, the hardware attributes of different hardware platforms can be compared. The designer can decide which hardware attributes are the most important ones and which are of minor interest when doing the comparison. Thus, an overall picture is used for comparison and single aspects are not forgotten which might turn out to be important later on. While the application of the selection process is described in [26], this paper focuses on hardware features of FPGAs which represent an important part of the selection approach presented.

III. Hardware Attributes of FPGAs

In the following subsections, the hardware attributes of FPGAs are evaluated based on several hardware properties (right part of Fig. 1). The evaluation of the impact of these attributes on the overall system is application dependent and has to be done by the system designer for the individual application (left part of Fig. 1). According to the limited space in this paper, the evaluations had to be kept briefly, but additional references are given for further information.

A. *Performance and Functional Range*

Modern FPGAs offer not only an currently increasing number of traditional logic cells and interconnect but dedicated multipliers, DSP blocks, Block RAM, clock management, multiple I/O standards, Multi Gigabit Transceivers and even embedded processors (hardwired). This has seen FPGA applications increase from simple glue logic to high performance applications such as Software Defined Radio (SDR), Digital Signal Processing and Cryptography.

The parallel nature of FPGAs means modules within a design will operate concurrently. This concurrency makes FPGAs well suited for use in industrial applications allowing several independent (real-time) functions to be implemented on a single chip.

Most applications will also require sequential operations to be performed. Support for implementing sequential structures is typically provided by the HDL selected. One popular method of implementing sequential operation is to implement dedicated state machines. State machines are used within designs as control mechanisms. Data paths which require a high data throughput might permit the use of a state machine (overhead) but can be implemented by connecting the corresponding modules in series. As the HDL is describing the actual hardware to be implemented, this provides the basic and most important aspects of a hard real time system, namely determinism and short response time. For advanced sequential control mechanisms embedded processors can be used. These can be either softcore or hardcore processors. Softcore processors are implemented using logic cells and resources within the FPGA while hardcore processors are fabricated upon the device silicon in addition to the FPGA resources. These softcore processors are available either in a device specific implementation (e.g. [5], [35]) or device independent implementations (e.g. [8], [10]). The availability of both hard- and softcore processors has lead to FPGAs being used as programmable system on chips (SoC) in which one or more of these processors is combined with functions implemented within the FPGA logic to achieve a system. A major advantage resides in the ability to develop custom peripherals implemented within the FPGA logic allowing complex operations to be off loaded from the processor [11].

[3]also named *hardware properties*

More about FPGAs as programmable SoC platforms can be found in section IV.

The structure of FPGAs allows efficient implementing of fixed point mathematics and the inclusion of dedicated arithmetic resources (multipliers, DSP blocks, and fast carry chains) has led to FPGAs being used within high performance applications. These dedicated resources make multiplication and multiply accumulates much simpler, faster and less resource intensive than they were in previous generations allowing FPGAs to be capable in many cases of exceeding the performance of traditional digital signal processors due to the parallel nature of the FPGA [4].

Of all the mathematical operations division has always been the most difficult to implement within an FPGA. While the other arithmetic functions are typically synthesisable from the operators (+, - ,*), division in this manner is generally only possible when the divisor is a power of two. However, there are numerous algorithms available for implementing a division within FPGAs and many Intellectual Property (IP) modules are available for this purpose (can be found e.g. in [3], [20], [34]). Further, more complex mathematical functions (SINE, COSINE, TAN, SINH, COSH, TANH, EXP, LN, SQR, etc.) can also be implemented within an FPGA, e.g. by applying CORDIC algorithms which allow most transcendental functions to be calculated [7]. This algorithm is fully pipelinable and can be implemented using shifts, additions and subtractions along with a small look up table. The use of CORDIC algorithms to perform EXP and LN functions also allows transformation from and to the logarithmic number system (LNS). Use of the LNS system allows the mathematical operations of multiplication, division and powers to be calculated using addition and subtraction however, addition and subtraction becomes more complicated [21].

While the detail is outside of the remit of this paper the new VHDL standard as proposed by accellera [2] introduces synthesisable fixed and floating point libraries which include a number of concepts which will further ease the implementation of mathematical operations and ease code understanding and maintenance.

Industrial applications typically interface to external sensors which are used to acquire plant parameters. Due to the parallel structure of the FPGA, multiple sensors can be processed in real-time concurrently, even if fast and complex signal pattern have to be evaluated or if each sensor requires a specific drive waveform such as a Sine, Triangular, or Pulse Width Modulation (PWM). Moreover, real parallel control of multiple actuators is possible with FPGAs. This includes simple digital signals (e.g. relays, valves) as well as computation intensive functions (e.g. Motion Control, Fuzzy Logic and Neural Networks [19]). Microprocessor systems on the other hand implement the acquisition-control-actuation process sequentially creating two major side effects: latency and software dependency. The latency, resulting from the sequential processing, can be reduced by specific algorithms, low level programming and increased clock frequencies. Moreover, on chip peripherals (e.g. multi purpose PWM peripheral) present in microcontrollers allow certain parallelism reducing the latency. Software dependency (e.g. changes in the software of a single software module could affect the whole application, especially in real-time systems) can be reduced by high level programming and operating systems. However, the measures reducing latency and software dependency typically conflict in these systems which makes trade offs necessary. On the other hand, changes in a single FPGA module affect only this module locally if design guidelines are followed.

Finally it is stated in [23] that the performance of DSPs is limited by their fixed architecture (fixed memory and data sizes, low number of MAC units, low number and limited performance of buses, and limited I/O resources) while the FPGA provides a much more flexible configurable architecture and an example is provided to demonstrate the impact of the number of MAC units on the performance.

B. Marketability

1) Costs: The costs can be subdivided into the costs for the device itself and the costs for development tools needed (costs for the actual development activities will be represented by time-to market). The cost of the FPGA device may be considered higher in comparison to that of a microcontroller traditionally used to implement industrial applications. However, the recent introduction of high performance, low cost FPGAs aimed at volume production from e.g. Altera and Xilinx (Cyclone and Spartan series) provides a viable cost effective alternative, especially if they allow to replace high performance microcontrollers or DSPs.

If sufficient resources are available within the FPGA device, any digital functionality (e.g. timer/counter units, specific controllers, other application specific circuits) present in the system could be integrated within the FPGA device reducing the number of devices and the printed circuit board area needed typically resulting in reduced hardware costs. Moreover, first FPGAs including also analog circuitry (analog inputs and analog to digital conversion) are available from Actel (*Fusion* devices) and Xilinx (*System Monitor* in *Virtex 5* devices).

Further costs associated with developing software-based approaches are the tools (e.g. Synthesize, Place&Route, Optimization, Simulation, Debugging) used within development. Design environments and synthesis tools can be purchased by third party companies as well as by FPGA vendors. The latter typically offer a free version of their tools. Due to the highly propriety nature of FPGA architectures it is difficult for third party companies to develop place and route tools. Device vendors therefore provide free development tools allowing the HDL to be implemented with their devices. These tools also include a host of other support applications as well e.g. programming tools and intellectual property (IP) generation.

Further on, the use of IP modules designed by third parties, which will be discussed in the next section, may reduce the design time. In case of IP modules which are not available free of charge this approach is resulting in additional costs.

Finally, FPGAs are a cost effective alternative to ASICs for low to medium volume production. Further on, designing with

FPGAs represents a lower risk than the corresponding ASIC design, as design changes are possible at any time.

2) Time to Market: As with any new development, one of the key driving factors in deciding upon following a given architecture, methodology or technology is the time it will take to get the newly developed product out in the market place.

FPGA based designs have several key aspects which can result in a reduced time to market. The most important aspect is the availability of a large number of customisable IP modules (sometimes called virtual components). IP modules allow the design engineer to quickly implement even the most complicated functions while at the same time providing a high level of confidence that the module will perform as required. These IP modules often allow the designer to customise the implementation to best suit the application. For example, a memory interface module will allow customisation of its timing parameters to match those of the memory device it will be controlling. A large number of these IP modules are available and FPGA manufacturers offer many of them free of charge. Those IP modules (purchased or free) are typically pre tested, which coupled with a large user base across a diverse range of applications ensure swift detection and correction of problems noticed with the IP module behaviour. The availability of behavioral HDL models of external components (Memories, Encoders, Decoders, FIFOs, Processors etc) allows the simulation of the overall system at early design stages. This approach increases confidence in the performance of the system and allows faults to be identified and corrected earlier in the design cycle reducing the overall design time.

While IP modules could be taken from external parties it could be also desirable to generate a number of own modules which can be developed into a library of modules reducing the design time on future products. This concept requires a slightly different approach to developing a module intended for single use only. The module should be designed to allow configuration of parameters if applicable through generics as opposed to using constants located within the design or packages. Ideally, a simple data sheet should be produced describing the modules IO type, functionality, timing and the effects of changing the generics.

Increasing the level of design abstraction and utilising a model based design approach can lead to a reduced time to market [29], which is also applicable in FPGA design. *Matlab* as one of the most common modeling tools can be used in combination with *Xilinx System Generator* or *Mathworks Simulink HDL Coder* to implement Matlab models in FPGAs. Moreover, specific tools can ease the design, for example filter design tools. In this case, the filter specification can be quickly turned into usable filters (HDL description) taking advantage of a higher level of design abstraction.

Other aspects can have a negative impact on the time to market. One of these aspects is the way of developing an FPGA design. As already mentioned in section III-A, it takes a few thoughts before a certain function (e.g. division) can be implemented in FPGAs. Approaches are present to tackle these issues, but this additional effort is not needed in MCU design. However, if DSPs need optimizations on assembly level, their implementation can take a comparable amount of design time [19].

The time required for an FPGA to be implemented can be considerable. The majority of the time taken will be within the mapping and routing stages where the design software has to not only fit the design within the selected device but meet a series of design constraints allowing the implemented design to operate at the required frequencies. This time can be considerable for larger designs easily consuming several hours which increases the time between iteration during debugging or design improvements. To aid faster recompile time several manufacturers allow partitioning of the design to enable reimplementation of the FPGA, with only the changed partitions being recompiled [6], [33].

3) Availability of Target Hardware: The availability of target hardware[4] is another important aspect for the overall life cycle. How long a device is available depends on the individual product and the policies of the corresponding companies. As with any other device, a sufficient availability of a chosen FPGA device is desirable.

In the case that a device is not available at a certain time in the life cycle, the design has to be transfered to another hardware platform. Depending on the device used in first place the effort for doing so can range from minimal to not acceptable. The effort is determined by three major aspects: 1) The effort to transfer the SW from one device to another, 2) the possibility to use the original design environment and tools with the new platform, and finally 3) the changes needed in the printed circuit board (PCB) layout.

In case of an FPGA design with a HDL, technology independence from both the tool chain and target device is easier to achieve than in case of most MCU/DSP designs. Issues in the latter case (e.g. processor endian and bit width, on chip peripherals and drivers, floating point support) can result in changes to the underlying hardware or tool chain having a big impact upon software source code making it less technology independent and more prone to obsolesce problems than an FPGA based approach. Use of a generic coding style will allow the HDL code to be synthesised by different synthesis tools and targeted at different devices reducing the risks due to devices going obsolete. A generic coding style will not instantiate any primitives available within the target device (RAMs, Multipliers, shift register, etc.). Instead, the code will be written in such a style that allows the synthesis tool to implement the correct primitives (hence inferring the primitives). This approach provides a design which is target and tool chain independent, which is portable and allows the synthesis tool to perform optimisations upon the logic described in the HDL [32]. A generic HDL coding style in theory would even allow the design to be completed and verified before the actual target device was chosen.

[4]*Availability of target hardware* is not present in Fig. 2 as it is considered as a *hardware feature* of FPGAs influencing the *hardware attribute marketability*.

Changes needed in the PCB layout depend on differences between the packages and the pin configuration of the new and the old device. Since FPGAs are typically available in different packages and almost all FPGA pins are freely configurable (exceptions are clock and power supply pins) these changes are typically not very critical.

C. *Modifiability*

FPGAs allow a high degree of modifiability which is not possible with microcontrollers. A good example are fieldbus interfaces which come with a high number of different standards. To achieve a flexible design, an FPGA based fieldbus communication controller has been proposed [28]. We will use this example to illustrate the different sub-issues of modifiability. For all four cases, it is assumed that sufficient free logic blocks are available in the FPGA.

1) Adaptability: If a bus interface has to be changed, e.g. from *Profibus* to *CANopen*, the communication controller can be reprogrammed if it is based on reconfigurable technology as FPGAs. The drivers for the physical layer (PHY) might have to be changed. Alternatively, several PHY of the most common fieldbuses can be integrated resulting in a comparatively low cost overhead.

2) Scalability: In case of a need for additional fieldbus interfaces, they can be simply added to the FPGA design. Again, just additional PHY devices are needed.

3) Configurability: Changes in the bus protocol (e.g. migration to an improved protocol version) can be achieved by reprogramming the FPGA without changing any hardware.

4) Maintainability: FPGAs allow to monitor all internal signals allowing in depth debugging which could ease maintenance.

D. *Reusability*

Since HDL modules represent electric circuits their reuse is not limited by specific properties of a CPU (e.g. bit-size, instruction size, special function registers, available processing time, side effects between different interrupts). Though, as in any design, care has to be taken that the modules created are of reasonable size and have well defined interfaces to allow effective reuse. A limitation for the reuse of HDL modules can be seen in incompatibilities of the interfaces used, but approaches are known to overcome this problem as the interface methodology proposed in [15].

E. *Dependability*

In this context, dependability defines the trustworthiness of a computing system which allows reliance to be justifiably placed on the service it delivers. As can be seen in Fig. 2, dependability is composed of robustness, reliability and security. Safety is not present here, since safety is a system property, and typically not the property of a hardware platform itself.

1) Robustness: As with CPU based systems, disturbances (e.g. radiation, electro magnetic interference (EMI), high voltages at pins) can lead to undesired behavior of the device. This can be counteracted by shielding the chip, by having protective circuitry at the I/O pins and by using robust silicon techniques. Moreover, it is indicated in [12] that it is not only the feature size, but also the supply voltages and the architecture itself which influence the hardware reliability. Further aspects of device robustness can be found in [27] and no major differences between the robustness of FPGAs and microcontrollers were identified in this work.

Industrial systems typically must be able to operate correctly within an electrically noisy environment. For this reason, it may be desirable to filter the sampled data to remove the effects of the noisy environment. The ability of FPGAs to perform mathematical functions with a low overhead in performance and resources is of use for this filtering that could be as simple as taking an average over a number of cycles of the main interference signal (if the frequency is known and fixed) or a digital band pass filter allowing through only the sensor excitation frequency.

2) Reliability: The term *reliability* targets the fault handling of HW and SW faults inside the system. One type of fault handling is *fault avoidance*, which can be achieved for HW faults part wise by a certain robustness of the device. Further measures for fault avoidance can be applied in the design process, the architecture and by verification measures at design time. According to the parallel nature of the FPGA, encapsulation of real-time functions is possible in dedicated hardware modules which can be seen as an advantage (avoidance of SW faults) in comparison to CPU based systems in which side effects are present by common memories and the common CPU [25].

The other important aspect of fault handling is *fault tolerance*. Fault tolerance is typically based on redundancy. While certain issues can be handled with information redundancy (e.g. checksum to tolerate memory errors), faults in the control structure or the computing elements are harder to detect. In this case, the parallel structure of the FPGA is again an advantage, since monitoring elements or redundant units can be implemented in parallel hardware allowing the detection of faults in real-time. If fault tolerance is desired, triple modular redundancy (TMR) can be applied. If, as in the case of TMR on a single FPGA, redundant channels are implemented on a single chip, *common cause failures* have to be considered. While clocks can be applied separately and minimum distances between the logic blocks in the FPGA can be achieved, the power supply remains a possible common cause failure in today's FPGAs which has to be considered.

Further discussions of the fault handling in FPGAs, especially in comparison to microcontrollers, can be found in [27].

3) Security: Two aspects of security have to be considered in this context: First, the intellectual property of the FPGA design itself has to be protected, both from reverse engineering and undesired modification. Several methods for IP protection in FPGAs are known and can be found e.g. in [16], [23],

[30]. Second, potentials of FPGAs for contributing to the security of the application itself can be of great importance. According to [30], FPGAs are attractive for executing the actual cryptographic algorithms and of particular importance from a security point of view for this reason.

F. Testability

One of the important aspects, not only during the design of industrial systems, is the verification of the design which can require up to 70% of the development cycle [14]. The verification cycle for FPGAs will involve verifying both, the individual modules as they are developed and the complete design. Verification in this context is nowadays dominated by simulation and testing whereas simulation can be seen as testing on a model of the system.

Simulation in case of FPGA designs is possible on different levels of hardware abstraction. The simulation can take place on behavioral level, to assure that the functionality is implemented as intended. Additionally, timing constraints of the hardware can be considered during simulation allowing a realistic simulation of real-time applications. While simulation comes with the advantage that no real hardware is needed, the time required to perform simulations can be considerable long, especially in larger designs. Therefore, at least in later stages of the design cycle and if reprogrammable FPGAs are used, it is useful, to use the physical FPGA device for testing.

While modern microcontrollers come typically with an inbuilt debugging system which can be interfaced via dedicated interfaces as JTAG, FPGAs do not included debugging facilities by default. A straight forward and universal option for debugging an FPGA is to break out internal signals to unused pins and connect these to a logic analyser. Moreover, there are numerous tools available which insert an *in chip logic analyser* into the FPGA (*Chip scope*, *Identify*, or *Signal Tap*) allowing internal logic signals to be monitored. These in chip logic analysers can be interfaced through the JTAG port and allow detailed in system debugging. According to the parallel nature of the FPGA, these *in chip logic analysers* have no effect upon the behavior of the implemented design and can even be left in the design if required to allow fault diagnosis in the field.

The testability of FPGAs is considered as very good for the reasons stated above, especially when it comes to real-time properties. However, certain effort is needed to integrate the modules needed for testing into the FPGA.

G. Physical Constraints

Physical constraints as power consumption and mounting space can be of great importance, depending on the application (e.g. battery powered devices).

In comparison with standard cell ASICs of the same technology, nowadays FPGAs have typically a more than 10 times higher power consumption [13]. The reason is due to a large number of transistors for field programmability and a low utilization rate of FPGA resources [17]. However, new approaches are known which try to overcome this problem, e.g. by switching off unused parts of the FPGA [13] or switching to a lower frequency when the logic is not in use [31]. Moreover, the performance of an FPGA may allow lower clock frequencies than an MCU or DSP performing the same task resulting in reductions of the original power consumption. For this reason, a direct comparison of FPGAs and MCUs/DSPs with respect to power consumption is difficult (an example can be found in [1]). Due to this application dependence, FPGA manufacturers provide power estimation software which uses the output of gate level simulations [6], [33] allowing the engineer to undertake efforts to reduce the power consumption if required [18].

FPGAs also allow the integration of numerous digital circuitry or even *system on chip* (SoC) approaches. This integration can typically reduce the number of external devices and thus the area needed on the printed circuit board.

IV. FPGAs as platform for SoC

The availability of both hard and soft core processors in FPGAs has lead to FPGAs being used as reconfigurable system on chips (rSoC) in which one or more of these processors is combined with functions implemented within the FPGA logic to achieve the system requirements. Commercial tools are available from FPGA manufacturers to support this approach (a good introduction can be found in [15]) and certain soft core processors are available free of charge.

A SoC based approach has several benefits including a reduced number of components and thus a reduction of the circuit board space, a global performance improvement and gain of freedom for the designer [9]. Further advantages can be seen in the increased portability of a HW design described at register transfer level (RTL), but the explicit use of hard cores in the FPGA counteracts to this advantage [9].

Also the reliability of a system can be affected by SoC approaches. At first, the number of components, connectors, and soldering points is reduced which has a positive impact on the reliability. However, according to the close coupling of several functions, common cause failures (according to e.g. high temperature, high voltage, shock) are more likely and have to be considered. Since SoC approaches allow an overall reduction of components, it might be reasonable to duplicate (completely or partly) a SoC in order to improve reliability.

However, due to the larger size and complexity of SoC designs, both compile and simulation times may increase.

V. Final Remarks on FPGA Design

In the previous sections it has been often stated, that parallel FPGA modules interact only via their interfaces and have less side affects for this reason. While this is true in a correct synchronous design, side effects are possible according to bad design practice. An example of this unwanted interaction is a design with two clock domains that are connected without specific measures. This may lead to a design which works intermittently as the two clocks are asynchronous and therefore have unknown phase relationship leading to metastability as the clock domains are crossed. Metastability can also be introduced, if input signals are not correctly synchronised to the correct system clock. These aspects require that the design engineer is aware of these issues and takes the appropriate steps within the design. Those hardware related aspects mentioned above typically do not have to be considered in microcontroller or DSP design (e.g. synchronisation stages are present at all MCU inputs by default). Naturally, challenges are present in theses systems also, as deadlocks, problems resulting from interrupts and memory violations, but to mitigate these problems only limited knowledge of the underlying hardware is required. While hardware related aspects are important in FPGA design, these issues are considered as uncritical, if good design practice (as e.g. proposed in [22]) is followed. Moreover, trends to develop FPGA designs on higher levels (e.g. by using *Xilinx Embedded Development Kit*(EDK)) can be observed. This approach might ease the handling of hardware effects in FPGA designs.

Finally, it has to be noted that versions of an algorithm developed for a microprocessor typically cannot be directly migrated to an FPGA (except soft cores are applied to execute the algorithm on the FPGA) [23].

VI. Conclusion

A systematic approach for hardware platform selection has been presented. This approach has then be applied to consider the potentials of FPGAs with a specific focus on industrial applications.

The high modifiability and reusability of FPGAs can be seen as their major advantages as well as their good performance, especially with respect to real-time applications. Further advantages are the possibility to implement reconfigurable *system on chips* which come along with further advantages.

These days, FPGA design needs more knowledge about the underlying hardware to avoid design faults than MCU/DSP design, but the amount of assistance given by design tools is increasing. Additionally, long compile times can be a draw back in FPGA design. Further on, FPGAs typically come with a higher power consumption than comparable ASICs. A comparison with MCUs/DSPs with respect to power consumption is application dependent.

Other aspects are very application dependent as for example the marketability of a device. However, the approach presented can help to perform the trade offs needed.

Finally, further investigations of the impacts of hardware features on the hardware attributes are needed to strengthen the results presented in this paper.

References

[1] A. Abnous, K. Seno, Y. Ichikawa, M. Wan, and J. Rabaey. Evaluation of a low-power reconfigurable DSP architecture. In *Workshop on Parallel and Distributed Processing*. Springer, 1998.

[2] Accellera. *Accellera VHDL-2006-D3.0*. Accellera, 2006.

[3] Altera. *AN343: OpenCore Evaluation of AMPP Megafunctions*, 2004.

[4] Altera. FPGAs for High-Performance DSP Applications. *FPGAJournal*, 2005.

[5] Altera. *Nios II Processor Reference Handbook*, 2007.

[6] Altera. *Quartus II Development Software Handbook v7.2*. Altera, 2007.

[7] R. Andraka. A survey of CORDIC algorithms for FPGA based computer. Technical report, Andraka Consulting Group, 1998.

[8] ARM. *ARM Cortex-M1 processor*. URL: http://www.arm.com/fpga/index.html (accessed Nov.26th, 2007).

[9] A. Astarloa, U. Bidarte, and A. Zuloaga. A reconfigurable SoC architecture for high volume and multichannel data transaction in industrial environments. In *28th Conf of the Industrial Electronics Society (IECON 02)*. IEEE, 2002.

[10] Gaisler Research. *GRLIB IP Library Users Manual (LEON 3 processor)*, 2007. URL: www.gaisler.com.

[11] R. Griffin. The root of all evil. *TechXclusives*, Xcel, 2005.

[12] R. Katz, K. LaBel, J. Wang, B. Cronquist, R. Koga, S. Penzin, and G. Swift. Radiation effects on current field programmable technologies. *IEEE Transactions on Nuclear Science*, 44, 1997.

[13] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. In *Proc of the 14th int. Sym. on Field Programmable Gate Arrays (FPGA'06)*. ACM, 2006.

[14] W. K. Lam. *Hardware Design Verification: Simulation and Formal Method-Based Approaches*. Prentice Hall PTR, 2005.

[15] T.-L. Lee and N. W. Bergmann. An interface methodology for retargetable FPGA peripherals. In *Int. Conf. on Engineering of Reconfigurable Systems and Algorithms (ERSA'03)*, 2003.

[16] A. Lesea. IP security in FPGAs. White paper WP261, Xilinx, 2007.

[17] Y. Li, F. Li, and L. He. Power modeling and architecture evaluation for FPGA with novel circuits for vdd programmability. In *Proc of the 13th int. Sym. on Field Programmable Gate Arrays (FPGA'05)*. ACM, 2005.

[18] S. McKeown, S. Fischaber, R. Woods, J. McAllister, and E. Malins. Low power optimisation of DSP core networks on FPGA for high end signal processing systems. In *Military and Aerospace Programmable Logic Device International Conference (MAPLD'06)*, 2006.

[19] E. Monmasson and M. N. Cirstea. FPGA design methodology for industrial control systems - a review. *IEEE Transactions on Industrial Electronics*, 54(4):1824–1842, 2007.

[20] ORSoC. *Open Cores*. URL: www.opencores.org.

[21] S. C. Panagiotis Vouzis and M. Arnold. Cotransformation provides area and accuracy improvement in a HDL library for LNS subtraction. *Euromicro Conference on Digital Systems Design*, 10th:85, 2007.

[22] C. Rockwood. Keeping time with clock domain crossings in FPGAs. *Chip Design Trends Reports*, 2007. www.chipdesignmag.com.

[23] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes. Features, design tools, and application domains of FPGAs. *IEEE Transactions on Industrial Electronics*, 54(4):1810–1823, Aug. 2007.

[24] J. Rose, A. E. Gamal, and A. Sangiovanni-Vincentelli. Architecture of field-programmable gate arrays. In *Proceedings of the IEEE*, 1993.

[25] F. Salewski and S. Kowalewski. Exploring the differences of FPGAs and microcontrollers for their use in safety-critical embedded applications. In *IEEE Symposium on Industrial Embedded Systems (IES'06)*, pages 1–4. IEEE, Oct. 2006.

[26] F. Salewski and S. Kowalewski. Hardware platform design decisions in embedded systems - a systematic teaching approach. In *Special Issue on the Second Workshop on Embedded System Education (WESE)*, volume 4, pages 27–35. SIGBED Review, ACM, Jan. 2007.

[27] F. Salewski and A. Taylor. Fault handling in FPGAs and microcontrollers in safety-critical embedded applications: A comparative survey. In H. Kubatova, editor, *10th Euromicro Conference on Digital System Design (DSD'07)*, pages 124–131. IEEE, Aug. 2007.

[28] M. Valdes, M. Moure, M. Dominguez, and E. Mandado. Improving industrial communications using reconfigurable devices. In *28th Conf. of the Industrial Electronics Society (IECON02)*. IEEE, 2002.

[29] J. Wilber. BAE systems proves the advantages of model-based design. Technical Report 15, The MathWorks, Sep 2006. MATLAB Digest.

[30] T. Wollinger, J. Guajardo, and C. Paar. Security on FPGAs: State-of-the-art implementations and attacks. *ACM Transactions on Embedded Computing Systems (TECS)*, 3:534 – 574, 2004.

[31] Xilinx. Virtex-II Pro and Virtex-II Pro X Platform FPGAs:Complete Data Sheet.

[32] Xilinx. *WP231: HDL Coding Practices to Accelerate Design Performance*. 2006.

[33] Xilinx. *Constraints Guide 9.1i*. 2007.

[34] Xilinx. *ISE 9.2i Software Manuals - Section Core Generator*, 2007.

[35] Xilinx. *MicroBlaze Processor Reference Guide - Embedded Development Kit EDK 9.2i*, 2007.