Cost-and Power Optimized FPGA based System Integration: Methodologies and Integration of a Low-Power Capacity- based Measurement Application on Xilinx FPGAs

Katarina Paulsson, Michael Hübner, Jürgen Becker
Universität Karlsruhe (TH), Germany
http://www.itiv.uni-karlsruhe.de/
{paulsson, huebner, becker}@itiv.uni-karlsruhe.de

Abstract

The application of Field Programmable Gate Arrays (FPGAs) in low power and low cost industrial mass products has become an important issue for designers of electronic systems. The flexibility and performance offered by reconfigurable hardware architectures often stands in the opposite to increased power consumption in comparison to Application Specific Integrated Circuit (ASIC) solutions. By exploiting the flexibility of reconfigurable hardware architectures, e.g. the capability of run-time HW reconfiguration of some modern FPGA devices, power consumption of FPGA-based solutions can be further decreased. This paper presents an approach for cost- and power optimized system integration of a low-power capacity-based measurement system by exploiting the dynamic and partial reconfiguration capability of Xilinx FPGAs.

Keywords: Low-power applications, reconfigurable architectures, hardware reconfiguration

1. Introduction

The application of Field Programmable Gate Array’s, FPGAs, has mainly been investigated for highly computational applications e.g. signal processing or multimedia devices. The typical characteristics (e.g. the structure of programmable switch boxes for the routing and programmable Look-Up Tables for realizing the logic behaviour) of this kind of hardware architecture make it very suitable for high performance applications that can exploit the possibility for parallel calculations. Normally however, it is not possible to exploit the flexibility of FPGAs for low-power applications (e.g. battery-driven applications) due to the higher static and dynamic power consumption in comparison to ASICs or low-power microcontrollers.

In a currently on-going project, the application of FPGAs for implementation in a low-power, low-cost capacity-based measurement system has been investigated. This application is normally realized with low-power microcontrollers, but in order to satisfy the increasingly complex requirements of this kind of system, it is necessary to further increase system flexibility and run-time adaptation. For example, this application will in a near future experience requirements on failure detection and recovery, flexibility regarding the available communication interfaces and fast run-time adaptation of the data processing algorithms. For this reason, Xilinx Spartan 3 FPGAs are investigated for the system implementation, since they offer the advantages regarding flexibility and performance and are also optimized for low-power and low-cost applications. Furthermore, they are dynamically and partially reconfigurable, which can in fact be exploited in order to reduce total power consumption by finding a trade-off between dynamic and static power consumption.

Dynamic and partial hardware reconfiguration allows functions that are not constantly required to be stored in a low-power memory and configured dynamically on-demand or according to a pre-defined schedule on the FPGA. This can reduce the required chip size and thereby decrease static power consumption, and is in fact one of the few possibilities for a system designer to decrease static power consumption of an FPGA-based system.

This paper presents the integration of the low-power and low-cost capacity-based level measurement application on a Xilinx Spartan 3 FPGA. The Spartan 3 family was selected due to its cost- and power optimized features, and since it allows the possibility of dynamic and partial hardware reconfiguration. Typical design considerations will be presented and proposed for power-efficient implementation of low-power applications on FPGA. This is important in order to allow even low-power applications to exploit the high flexibility of reconfigurable architectures. The paper is organized as follows: chapter 2 describes the target application: the capacity-based level measurement system. Chapter 3 gives a brief overview of dynamic reconfiguration as well as how this feature can be exploited to decrease power consumption. Chapter 4 presents the system integration on a Spartan 3 FPGA as well as novel considerations for low-power design. Finally chapter 5 concludes the paper and presents some future work.
2. Application scenario

The target application is a low-power capacity-based level measurement system. The system measures the level of material in a tank by monitoring the change of capacity within the tank. The level of the material is then given by the calculated capacity. Normally this application is implemented on low-power microcontrollers, but the exploitation of FPGAs is investigated for increasing system flexibility and adaptivity to dynamically changing requirements.

For example, FPGAs allow the implementation of efficient failure detection mechanisms as well as fast runtime adaptation of the data processing algorithms, which can be exploited for optimizing the calculations and the system implementation to changing requirements on power consumption and performance.

![Figure 1. FPGA based level measurement system](image)

The basic functionality of the measurement system can be described the following way: the system generates an 8-bit wide digital signal, which is converted into an analog sinus signal by an external DA converter. The signal is then applied to the tank in which the level measurement is performed. Every measurement cycle the current is detected back from the tank and converted into digital data by an external AD-converter. The digital data is then processed and the level of material in the tank is calculated. An overview of the complete FPGA-based system is given in Figure 1.

The main required system components are the sinus generator for generating the sinus signal, the external AD-converters for converting the measurement and reference signals into digital data and components for calculating the level out of the collected measurement data. On an FPGA this can either be done in software by using a soft-core microcontroller or in hardware by implementing the algorithms as data processing modules. Furthermore, some different interface components are used such as Ethernet and profibus components. The result of the current level may also be displayed on an external display, which is controlled by an UART component.

However, as already mentioned, an FPGA integration of the system would initially lead to increased power consumption in comparison to a low-power microcontroller implementation. Therefore, focus was made on cost-and power efficient system integration. Three approaches have been investigated and are presented in section 4 in this paper.

3. Dynamic and partial hardware reconfiguration

Some of Xilinx FPGA families (e.g. Virtex 2, Virtex 4 and Spartan 3) offer the possibility to perform dynamic and partial hardware reconfiguration. The exploitation of this feature has been investigated in several research projects for different kinds of applications, e.g. in [9] and [13].

The main part of the earlier investigations where state-of-the-art FPGAs have been used are based on Virtex 2 or Virtex 4 FPGAs, which include the Internal Configuration Access Port, ICAP. This component enables the system to perform internal self-reconfiguration.

When designing dynamically and partially reconfigurable systems based on these architectures, several design considerations must be made. For example, an internal controller (e.g. a hard/soft-core microprocessor) is required to manage the reconfiguration process (fetching the bitstreams from an external memory and write them to the configuration port) as well as other static components such as interfaces for communication that may be constantly required and therefore not dynamically reconfigured.

![Figure 2. Reconfigurable measurement system on FPGA](image)

These components must be implemented on a static part of the FPGA, while the reconfigurable components are configured on the dynamic side of the FPGA. The dynamic side must also be partitioned in reconfigurable slots of specific sizes by considering the sizes of the reconfigurable functions. Figure 2 shows a graphical overview of a typical dynamic and partial hardware reconfigurable system that is divided in a static and dynamic part. This specific system (the level measurement system) only uses 1 reconfigurable slot.
Slice based busmacros [8] are used for the communication between the static and dynamic areas. In [5] the dynamic interconnection of reconfigurable modules is discussed. The authors present two algorithms for dynamic routing of the communication between reconfigurable modules. This could be interesting in order to decrease the bitstream overhead and thereby reduce memory requirements for the reconfigurable modules. The design of the static and dynamic areas and the dynamic scheduling of tasks are especially complicated for large reconfigurable systems. In [10], an approach for automatic generation of the static and dynamic parts is presented. Automatic tools for the design of on-demand reconfigurable systems with real-time requirements will be required in order to make dynamic reconfiguration suitable for industrial applications in a long-term perspective.

4. System integration and application exploration

As already mentioned, the original system implementation is based on a low-power microcontroller. In the first implementation of an FPGA-based prototype the original realization was simply ported and a soft-core microcontroller (MicroBlaze [6]) was used to execute the same software algorithms as the original system. Due to the high requirements of internal BRAM memory of this first implementation, the following work was focused on optimizing the implementation for FPGAs, by following three approaches that are presented in this section. The first approach considers the complete system, and optimizes the implementation for FPGAs by integrating discrete components into the FPGA system. The second approach aims at optimizing the implementation for FPGAs by realizing the original software algorithms in hardware as well as exploiting dynamic and partial hardware reconfiguration for decreased resource utilization. Finally, the third approach deals with optimizing the actual layout of the FPGA design for reduced power consumption.

The Spartan 3 [2] FPGA family was selected due to its cost-and power optimized features. Furthermore, this architecture also allows dynamic and partial hardware reconfiguration. Unfortunately the Spartan 3 does not include an internal configuration port such as the ICAP, but in [11], the implementation of a virtual internal configuration port based on the JTAG interface is presented.

4.1. Integration of external digital components

The capacity-based level measurement system requires external components such as DA and AD converters. In the first FPGA-based implementation of the system, these components were implemented as external components on the FPGA board. FPGA tools allow a System-On-Chip design flow based on so-called Intellectual Property, IP, cores and they also provide possibilities for complete hardware/software simulation and test. Therefore, one way to simplify the design process, and thereby reduce manufacturing costs, is to integrate the external digital components in the FPGA system. IP cores can also be designed to be parametrizable, which enables uncomplicated adjustment to different product variations by simply setting the specific parameters and re-generate the hardware.

Xilinx offers delta-sigma DA and AD converters for the Spartan 3 FPGA family. The initial step towards optimized FPGA implementation was therefore to integrate the digital parts of the external AD and DA converters into the FPGA system. Another possible optimization of this would be to only configure the DA/AD converter/s when they are required, which is restricted to the initial phase of each measurement cycle. Naturally only digital signal processing can be performed on FPGA; so simple external filters are still required.

The sinus generator was first implemented on FPGA as a look-up table stored with sinus values and an address counter. The address counter generates the addresses to the look-up table that contains the pre-defined sinus values. The sinus values are then read out of the look-up table and written to the external DA converter. The frequency of the measurement signal should be 500 kHz, so therefore the look-up table was filled with 32 sinus values and the address counter was running with a frequency of 16 MHz. The external DA converter was also running with a sample frequency of 16 Mega Sample per Second, MSPS.

For the improved FPGA-based prototype, the external DA converter was replaced by the delta-sigma converter core from Xilinx, as well as an external low-pass filter and anti-alias filter to eliminate the high-frequency components. Figure 3 gives an overview of the sinus generator based on the internal DA converter. A fixed implemented Digital Clock Manager, DCM, was used to generate the different clock frequencies.

The Xilinx delta-sigma DA converter is typically suitable for audio applications, and a sample frequency of 16 MSPS cannot be achieved from this converter. However,
by performing real hardware tests and Fourier analysis it was concluded that the delta-sigma DA-converter could run with a frequency high enough to generate a 500 kHz sinus signal. The delta-sigma DA-converter IP core includes an interface to the IBM On-Chip Peripheral Bus [3]. For this system however, the interface was not required and was therefore removed to save resources. The total resource utilization was therefore restricted to ca. 50 slices for the complete sinus generator.

This implementation may be further improved by performing dynamic and partial reconfiguration of the DA/AD converter/s so that they only utilize FPGA resources when they are required, which is in the beginning of every measurement cycle.

4.2. Reconfiguration for power optimization

As described in section 2, dynamic and partial hardware reconfiguration of state-of-the-art FPGAs can be applied in order to increase system flexibility and improve the system run-time adjustment to application requirements. However, dynamic and partial hardware reconfiguration may even be exploited to reduce both static and dynamic power consumption, which will be further explained here.

In the original microcontroller based implementation the data processing algorithms for calculating the level from the data of the AD converters were implemented in software. In the initial FPGA-based implementation, the identical software algorithms were used and executed by the MicroBlaze soft-core processor. The large memory utilization of these algorithms made this implementation unsuitable for FPGA, since FPGAs normally include restricted number of internal BRAM blocks. In fact, in the first implementation of the measurement system on a Spartan 3- 200 FPGA, the software algorithms required more than 60 Kbyte of memory, which made it necessary to store the code in external SRAM.

In order to optimize the implementation for FPGA, the software algorithms were implemented as hardware components in the System Generator tool from Xilinx. This is further described in [11], but will be briefly explained here.

Figure 4 gives an overview of one measurement cycle. The complete system was then partitioned in a static and a dynamic part, where slice-based bus macros were used for communication over the boarder between the two parts. A direct signal communication interface, the Fast Simplex Links (FSL), from Xilinx was used for communication and was extended with busmacros over the boarder between the static and dynamic areas.

The main reason for implementing the data processing algorithms as reconfigurable hardware modules was to reduce the memory requirement of the system. Otherwise a much larger FPGA would have to be selected in order to provide enough internal BRAM components, but the logic and routing resources would not be completely utilized and therefore wasted. Instead the algorithms were implemented in hardware, which reduced the BRAM utilization and increased the processing performance of the system. The increase in performance due to parallel computations is important because it allows a reduced clock frequency, which further reduces dynamic power consumption. Furthermore, dynamic and partial hardware reconfiguration was exploited to divide the hardware implementation of the processing algorithms into reconfigurable modules, which allows implementation on a smaller chip, since not all modules are configured simultaneously.

Figure 5 shows the implementation of the reconfigurable measurement system as it looks like in the FPGA Editor tool [4]. Since the reconfigurable modules were partitioned according to functionality, they differ in size.

Figure 4. Tasks performed in one measurement cycle

The complete system was then partitioned in a static and a dynamic part, where slice-based bus macros were used for communication over the boarder between the two parts. A direct signal communication interface, the Fast Simplex Links (FSL), from Xilinx was used for communication and was extended with busmacros over the boarder between the static and dynamic areas.

The main reason for implementing the data processing algorithms as reconfigurable hardware modules was to reduce the memory requirement of the system. Otherwise a much larger FPGA would have to be selected in order to provide enough internal BRAM components, but the logic and routing resources would not be completely utilized and therefore wasted. Instead the algorithms were implemented in hardware, which reduced the BRAM utilization and increased the processing performance of the system. The increase in performance due to parallel computations is important because it allows a reduced clock frequency, which further reduces dynamic power consumption. Furthermore, dynamic and partial hardware reconfiguration was exploited to divide the hardware implementation of the processing algorithms into reconfigurable modules, which allows implementation on a smaller chip, since not all modules are configured simultaneously.

Figure 5 shows the implementation of the reconfigurable measurement system as it looks like in the FPGA Editor tool [4]. Since the reconfigurable modules were partitioned according to functionality, they differ in size.
The system shown in Figure 5 includes the module for calculating the amplitude and phase of a signal, and this module is the largest one, which is shown in Table 1.

Figure 5. Reconfigurable system in the FPGA Editor

The system was implemented on a Spartan 3 1500 FPGA, but by re-partitioning the modules into e.g. 5 reconfigurable modules of smaller sizes, the system could be implemented on a Spartan 3 200 FPGA.

Table 1 Resource utilization of the system (Spartan 3 1500 FPGA)

<table>
<thead>
<tr>
<th>Amp &amp; Phase Calc. Comp.</th>
<th>Capacity Comp.</th>
<th>Filter Comp.</th>
<th>Static area (MicroBlaze,FS L, RS232, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices</td>
<td>2980</td>
<td>1778</td>
<td>423</td>
</tr>
</tbody>
</table>

Implementing the complete system without exploiting reconfiguration would require more than 6000 slices and at least a Spartan 3 1000 FPGA. By exploiting hardware reconfiguration the FPGA size could be reduced from a Spartan 3 1000 to a Spartan 3 200. Furthermore, the processing performance increased with approximately a factor 1000, from 7 ms of processing time for the software-based algorithms to 7 µs (without performing reconfiguration). This allows a decreased clock frequency that also decreases dynamic power consumption.

It is also very important to consider the time overhead induced by the reconfiguration process. The ICAP core [11], which can be used for performing dynamic and partial reconfiguration on Spartan 3 FPGAs, offers a reconfiguration rate which is lower than the one provided by the ICAP interface. However, in [11], where the implementation of the ICAP core is described more in detail, it is also described how the reconfiguration rate provided by the ICAP core may be increased.

4.3. Power optimized Place-and-Route

Not only the HW/SW implementation can be optimized for reduced power consumption, but even the actual layout of the FPGA system. Dynamic power consumption of an FPGA-based system greatly depends on the capacitance in the design.

Xilinx FPGAs include different signal lines for implementing the routing. There are longer signal lines with higher performance that cause higher power consumption and there are the shorter signal lines (direct, double), which provide less performance but also cause less power consumption. Earlier investigations has shown that using multiple shorter signal lines instead of 1 long signal line can reduce dynamic power consumption in an FPGA design [12]. Naturally the requirements on performance must be considered while performing these adaptations of the layout.

Performing adaptation of the routing for reduced power consumption will not bring the same effect on all signal nets. Optimizing the nets with higher communication rates first will lead to better results. Furthermore, the placement of the utilized slices also affects the routing. The logic of the nets with higher communication rates can be placed closer during the Place-and-Route process in order to decrease the distance for the signal routing. In order to estimate the communication rates of the design, a Post-Place-and-Route Simulation was performed while generating a so-called Value Change Dump, VCD, file. The VCD file can be imported into XPower [4], where estimation of the communication rates was performed.

Figure 6. Optimized signal net in the FPGA Editor; 1) Before reallocation, 2) After reallocation

This was done for the hardware implementation of the data processing modules, and results from some of the net optimizations will be presented here. Naturally this kind of technique can only be performed in a restricted manner if the approach is not considered automatically by the Place-and-Route tools. The results presented here simply show that a large reduction of power consumption is possible if this technique is integrated in the design tools.
Figure 6 shows the optimization of one net as it looks like in the FPGA Editor. The net was selected according to its communication rate, and as can be seen in Figure 6 the reallocation of the logic lead to an improvement of the signal routing.

The net shown in Figure 6 consumed ca. 53.92 μW before optimization, which was reduced to 23.60 μW by the reallocation of logic functions to other slices. This corresponds to a reduction of 56%.

Table 2 Example of power optimized signal nets

<table>
<thead>
<tr>
<th>Signal net</th>
<th>Power dissipation (originally, in μW)</th>
<th>Power dissipation (after replacement, in μW)</th>
<th>Reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ce_2_sg</td>
<td>1089.2</td>
<td>1016.1</td>
<td>6.7</td>
</tr>
<tr>
<td>multi../N224</td>
<td>51.9</td>
<td>36.9</td>
<td>28.8</td>
</tr>
<tr>
<td>multi../N2235</td>
<td>45.6</td>
<td>34.2</td>
<td>24.9</td>
</tr>
<tr>
<td>multi2../N359</td>
<td>144.4</td>
<td>133.7</td>
<td>10.7</td>
</tr>
</tbody>
</table>

Table 2 presents some additional nets that where the logic was reallocated. Note that not all optimized signal nets are listed here. The table presents the power consumption before and after optimization and also the reduction in percentage. The reduction, which is given in Table 1, is the reduction of power consumption of the specific net, not the complete of the complete power consumption. After every reallocation process it was however verified that the dynamic power consumption had decreased and not increased.

It is important to stress here though that the approach that is presented in this paper is not a methodology that a designer should use straight off to decrease power dissipation in a design. This approach and the achieved results simply show that even FPGA designs can be significantly optimized for consuming less power. Clearly it must be integrated in FPGA tools and performed during the place-and route phase in order to bring most efficiency.

5. Conclusions

This paper presents three approaches towards cost-and power efficient implementation of a low-power capacity-based level measurement system on FPGA. The first approach aimed at integrating external components into the FPGA system in order to simplify the design process and thereby reduce costs and power dissipation.

The second approach was to exploit dynamic and partial hardware reconfiguration for decreased static and dynamic power consumption and the final approach deals with power optimized implementation of the design. Investigations have shown that the Place-and-Route process can be adapted for reducing dynamic power consumption by minimizing the capacity in the design. Pre-simulation of the design can be performed to generate the communication rates for the different nets in order optimize the routing of the nets with the highest communication first.

The increasing complexity of the requirements (e.g. multiple interfaces, fault tolerance, run-time adaptation of the data processing algorithms) of this kind of application forces industry to investigate alternative hardware architectures. The main contribution of this paper was to demonstrate that the usage of reconfigurable hardware could be enabled even for low-power systems.

6. References