Fast Feasibility Estimation of Reconfigurable Architectures

Popp, Andreas; Le Moullec, Yannick; Koch, Peter

Published in:
4th IEEE Conference on Industrial Electronics and Applications, 2009. ICIEA 2009

DOI (link to publication from Publisher):
10.1109/ICIEA.2009.5138181

Publication date:
2009

Document Version
Accepted author manuscript, peer reviewed version

Fast Feasibility Estimation of Reconfigurable Architectures

Andreas Popp, Yannick Le Moullec, and Peter Koch Center for Software Defined Radio & Technology Platforms Section,
Department of Electronic Systems, Aalborg University
Fredrik Bajers Vej 7A, 9220 Aalborg Øst, Denmark
{anp,ylm,pk}@es.aau.dk

***Abstract*—Reconfigurable architectures are often said to be able to exploit the possibilities of resource savings by means of hardware time-sharing. However, existing literature does not point clearly at which conditions must be fulfilled for considering a reconfigurable architecture for the implementation of signal processing applications. Therefore, we propose a fast method to perform high-level pre-implementation feasibility-based evaluation of a reconfigurable hardware implementation. The method is based on a light architectural model to compute costs of a static reference as well as costs for globally and partially reconfigurable architectures. Two case studies have been performed for an FFT and an FPGA-based DAB application. Our results show that implementation on reconfigurable architectures is only feasible when the reconfiguration time is low, which generally means that a dynamically partially reconfigurable solution is preferred.**

***Index Terms*—Field programmable gate arrays, reconfigurable architectures, performance evaluation, feasibility**

I. Introduction

Reconfigurable hardware architectures have been introduced as a possibility to provide an intermediate solution between Application Specific Integrated Circuits (ASIC) or Application Specific Instruction-set Processors (ASIP) and Digital Signal Processors (DSP) [1]. Reconfigurable hardware is known to offer the opportunity of resource and energy savings for some applications due to the possibility of time-sharing of the hardware resources, as well as run-time circuit specialization allowing an accelerator that is ultimately customized to the task executing at any given moment of operation.

One of the most utilized reconfigurable architectures is the Field Programmable Gate Array (FPGA), where an example is the Xilinx Virtex series with the improved version of Dynamic Partial Reconfiguration (DPR) [2] in the newest Virtex-4 and 5 series. In DPR, also noted *partial reconfiguration* in the rest of this text, parts of the logic can be reconfigured while maintaining operation on the other parts. The application of the inherent flexibility of DPR has been demonstrated especially in the field of Software Defined Radio (SDR), among others by Delahaye et al. [3] and Ihmig et al. [4] where DPR allows the implementation of several functionalities without having to perform parallel implementations of all functionalities. Furthermore, extensive research efforts in both academia and industry have been put into *i)* synthesis tools and methods to perform scheduling of algorithms onto reconfigurable architectures by e.g. Bobda [5], and *ii)* technical solutions to reduce the reconfiguration overhead suggested by e.g. Hauck [6].

However, even though the use of reconfiguration in FPGA architectures seems promising, it is important for the designer to realize and remember that it is associated with certain costs to provide and use reconfiguration capabilities. Firstly, reconfiguration takes time (known as reconfiguration overhead) and consumes power. Secondly, reconfigurable architectures generally also consume more power, area, and have longer execution time than non-reconfigurable or static solutions. Finally, development time is also longer than for non-reconfigurable architectures, as reconfigurable hardware requires the developer to spend more time on design as well as test and debugging of the implementation.

Shoa & Shirani [8] have given a survey on reconfigurable systems in the context of digital signal processing operations. One conclusion is that FPGA implementation is suitable for data-intensive operations like FIR-filters, FFT and DCT transforms. In traditional FPGA implementations the reconfiguration capabilities are not utilized. However, the inherent lack of flexibility during run-time is a motivation for considering reconfigurable architectures. The survey concludes that reconfigurable architectures should be considered due to possibilities of run-time circuit specialization and logic resource savings by time-sharing among hardware resources.

Although many applications of reconfigurable architectures based on FPGAs have been built, there is, to the best of our knowledge, a lack of clear pointers in the direction of determining when a reconfigurable implementation is feasible. In this paper, feasibility is defined as a non-reduction in performance as compared to a static implementation. Thereby, the study does not include the implementation effort in terms of development costs. The ability to derive a pre-implementation estimate before conducting the final implementation is considered an important basis for deciding whether it is worth the man-hours to perform the implementation in reconfigurable hardware architectures versus non-reconfigurable hardware. Therefore, we have posed the question:

"What high-level characteristics of the application must be fulfilled, and in which conditions is it feasible to make an implementation using reconfigurable hardware?"

This is the author's version of the paper published in
4th IEEE Conference on Industrial Electronics and Applications, ICIEA, 2009

Previous approaches to answer similar questions have mainly been focused on developing a full implementation and comparing it to another implementation in static hardware or programmable processors. Typically, solutions are compared by means of a cost-function or metric that weighs time, silicon area or resource usage, energy or power consumption, and other factors such as numerical properties. Such a cost-metric is used by Wirthlin & Hutchings [7], who provide an estimation method to evaluate the feasibility of a reconfigurable implementation. The evaluation is based on functional density, D, which is a throughput oriented cost metric including area, A, and total operation time, T, and combining these by the expression $D = \frac{1}{A\,T}$. Feasibility is determined from

$$I_{\max} \geq f \quad ,$$

where $I_{\max} = \frac{D_{\text{reconfigurable}}}{D_{\text{static}}} - 1$ is the improvement in functional density over a static implementation and f is the configuration ratio defined as the relation between total time spent on configuration and the total time spent on execution. This means that in case the area A is reduced by a factor of two, a two-fold increase in execution time, T, gives exactly the same functional density, D. This leads to an improvement $I_{\max}$ of 0%, thus if any time is spent on reconfiguration, f will become greater than 0, and a reconfigurable solution is deemed infeasible despite that the static and reconfigurable solution have equal functional density D. Similarly, if execution time is only increased by a factor of 1.5, then as long as 33% or less of the execution time is spent on reconfiguration, the throughput will not be degraded compared to the static reference. While the work evaluates feasibility of reconfigurable architectures, it has two limitations:

- The throughput oriented metric does not reflect the possibility of time-sharing resources and thereby reduction of the area-costs.
- Partial reconfiguration cannot be evaluated, as DPR is not directly reflected in the configuration ratio.

Manet et al. [9] evaluated dynamic partial reconfiguration for non-consumer applications based on selected scenarios where DPR could be advantageous. The evaluation shows that DPR has clear advantages when changes occur in environment or functions (denoted "mission change"). Furthermore, advantages are shown by the use of hardware time-sharing to obtain hardware resource reduction. However, the evaluation of the advantages of DPR is subjective and an objective measure is desired.

A. Contribution

In this work we develop a light architectural model for globally and partially dynamically reconfigurable architectures. The model describes high-abstraction level characteristics of the architecture. The characteristics are considered adjustable to the architecture under consideration. The feasibility estimation method consists of two subsequent steps:

1) **Analysis** of the application from a high level of abstraction to determine execution patterns for the reconfigurable architecture.

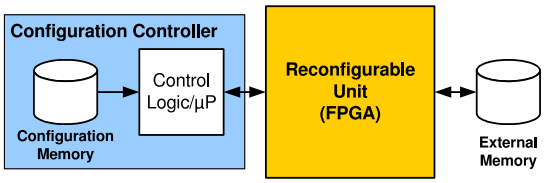

Fig. 1. The basic reconfigurable architecture. The controller can be on-chip or off-chip, but the control resources are separated from the computational resources. The external memory is used to save intermediate data that is used in subsequent configurations.

2) **Logic synthesis** of parts or modules to estimate costs. The costs can also be estimated based on a cost-library for basic functions. The estimates are input to the architectural model to evaluate the feasibility by means of a *cost-function*.

The focus of the cost-function is put on the time and area trade-off that is made possible by time-sharing of resources and not on the flexibility that is provided for applications. Area is counted based on logic resources and the costs of software processors or controllers are not considered.

In the following, the method is presented. This is followed by two case-studies and presentation of the result of these studies. Finally, the results are discussed followed by the conclusion.

II. Method

The method consists of a conveniently light architectural model to describe the characteristics of the architecture. This is followed by a description of the application analysis.

A. Architectural Model

The architectural model is limited to consider a reconfigurable unit, a controller with configuration memory, and external memory as depicted in figure 1.

The proposed architectural model is the basis of two cost models, describing globally reconfigurable architectures and partially reconfigurable architectures. The models describe the capabilities of the architectures from a high-level point of view and capture time and resource parameters. Time costs are categorized on the basis of time spent on execution/computation, reconfiguration, or data transfer. There are many possible area parameters for quantifying resource costs, such as Configurable Logic Block (CLB) and DSP slices, reconfiguration resources, and RAM/memory resources for data, configuration and intermediate data representation. However, in this work area resources are only counted in CLB slices allocated for execution, based on the results from logic synthesis.

The improvement or degradation caused by a reconfigurable implementation is based on a comparison to a static implementation. The static costs, C_{static}, are expressed by

$$C_{\text{static}} = A_{\text{static}} \cdot T_{\text{static}} \quad [\text{s} \cdot \text{slices}] \quad , \qquad (1)$$

where A_{static} is the total area in CLB slices of the architecture, and T_{static} is the total execution time in seconds. The area

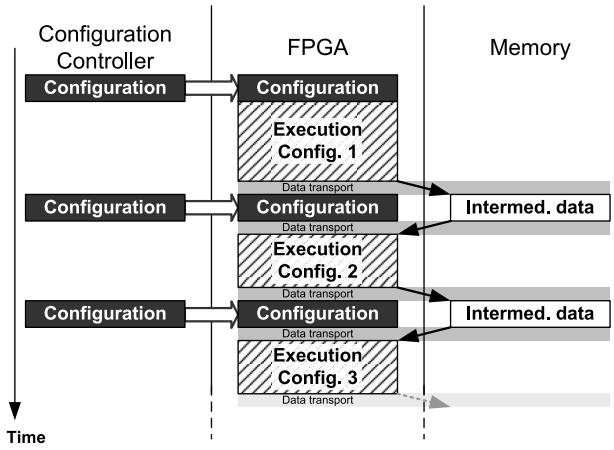

Fig. 2. Execution flow in global reconfiguration.

is inherently two-dimensional thus the costs are conveniently described in three dimensions.

In our proposed cost model, time and area are given equal weight in order to fully reflect the area-time trade-off in time-sharing of resources. In case certain area or time constraints must be fulfilled, these constraints are evaluated externally to the cost evaluation.

B. Dynamic Global Reconfiguration

In the case of dynamic global reconfiguration, it is assumed that reconfiguration and execution cannot be overlapped, which is a general assumption for globally reconfigurable FPGAs. The execution flow is illustrated in figure 2 and proceeds as follows: First, the controller configures the FPGA. Then the FPGA executes the tasks of configuration 1 and stores intermediate data in the external memory. Then configuration 2 is programmed into the FPGA, followed by reading the intermediate data from memory. This process repeats itself until all configurations have been executed.

Time costs can easily be described by the sum in (2) that describes time-consuming parts of execution in a globally reconfigurable system:

$$
\begin{aligned}
T_{\mathrm{exec}} &= \sum_i t_{\mathrm{exec},i} \\
T_{\mathrm{reconf}} &= \sum_i t_{\mathrm{reconf},i} \\
T_{\mathrm{transfer}} &= \sum_i t_{\mathrm{read},i} + \sum_i t_{\mathrm{write},i} \\
T_{\mathrm{global}} &= T_{\mathrm{exec}} + T_{\mathrm{reconf}} + T_{\mathrm{transfer}} \quad , \qquad (2)
\end{aligned}
$$

where the symbols are defined as in table I.

The total cost of the globally reconfigurable solution is given by multiplying equation (2) by the area in CLB slices, A_{global}, of the globally reconfigurable architecture:

$$
C_{\mathrm{global}} = A_{\mathrm{global}} \cdot T_{\mathrm{global}} \quad [\mathrm{s} \cdot \mathrm{slices}] \quad , \qquad (3)
$$

which can then be compared to C_{static}, (1).

TABLE I
DEFINITION OF SYMBOLS IN (2).

I	Total number of configurations
i	Configuration index, $i \in \{1, 2, \ldots, I\}$
T_{global}	Total time spent in the global reconfiguration scenario [s]
T_{exec}	Total time spent on execution [s]
T_{reconf}	Total time spent on reconfiguration [s]
T_{transfer}	Total time spent on data transfer [s]
$t_{\mathrm{exec},i}$	Execution time of configuration i [s]
$t_{\mathrm{reconf},i}$	Reconfiguration time of configuration i [s]
$t_{\mathrm{read},i}$	Memory read-time for input to configuration i [s]
$t_{\mathrm{write},i}$	Memory write-time for output from configuration i [s]

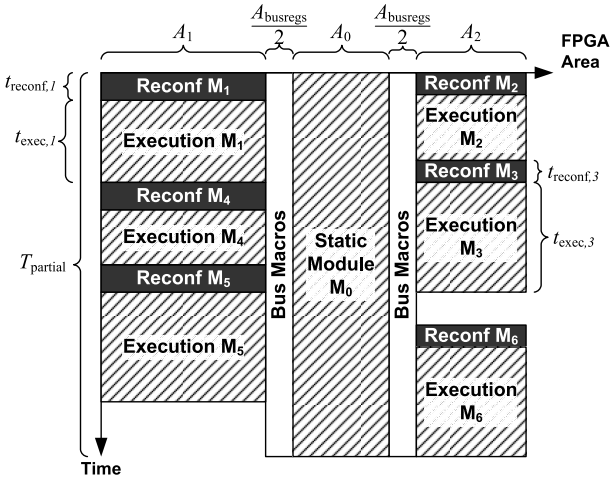

Fig. 3. Execution flow in dynamic partial reconfiguration.

C. Dynamic Partial Reconfiguration

The partial reconfiguration model is basically similar to the model for global reconfiguration. However, instead of multiplying the time and total area for global configurations, the sum of reconfiguration and execution time is multiplied by the resources consumed by each reconfigurable module.

The model assumes that transfer of data between modules is performed by special bus registers, so called Bus Macros in Xilinx tool flows [2], and the transfer delay across Bus Macros is assumed negligible. However, the bus registers consume area during the whole operation. Furthermore, the placement of bus macros is assumed fixed during operation, as this is similar to current DPR implementation in Xilinx FPGAs [2]. The execution flow is illustrated in figure 3. The figure has one static module, M_0, that is active during the whole execution T_{partial}. There are two bus macros that handle communication of data between the reconfigurable modules and the static module. The configuration of the static module and the bus macros is not included in the costs, as it is assumed being a part of the general start-up of the FPGA. The six static modules M_1-M_6 are reconfigured prior to their execution. As indicated in the figure, there are periods where some of the resources are unused for execution. This is not included in the costs, as the area is theoretically available for other functionalities.

TABLE II
DEFINITION OF SYMBOLS IN (4), ALSO ILLUSTRATED IN FIGURE 3.

j	Module index
A_j	Area of module j [slices]
A_{busregs}	Area of the bus registers [slices]
$t_{\text{exec},j}$	Execution time of module j [s]
$t_{\text{reconf},j}$	Reconfiguration time for loading module j [s]
T_{partial}	Total execution time [s]
$C_{\text{partial,proc}}$	Cost of processing and reconfig. in DPR [s·slices]
$C_{\text{partial,comm}}$	Cost of communication in DPR [s·slices]
C_{partial}	Total cost of dynamic partial reconfiguration [s·slices]

The total cost of a partially reconfigurable implementation, C_{partial}, is expressed by

$$C_{\text{partial}} = C_{\text{partial,proc}} + C_{\text{partial,comm}} \quad [\text{s} \cdot \text{slices}] \tag{4}$$
$$C_{\text{partial,proc}} = \sum_j A_j \cdot t_{\text{exec},j} + \sum_j A_j \cdot t_{\text{reconf},j}$$
$$C_{\text{partial,comm}} = A_{\text{busregs}} \cdot T_{\text{partial}} \quad ,$$

where the symbols are defined as in table II. C_{partial} can be compared to C_{static}, (1), as well as C_{global}, (3).

D. Application Analysis and Logic Synthesis

The application analysis is performed by an examination of the application to demonstrate how to extract the parameters of the architecture model described in the previous section.

From a high level of abstraction the application and specifications are analyzed to determine the deadline, T_{deadline}, at which the task-set, (i.e. all operations), must be finished. The task-set can either be a one-time running application or periodic tasks. For periodic tasks, the deadline is equal to the longest period of the tasks. Since the static reference occupies all resources from execution start to deadline, T_{static} is set to T_{deadline}.

In the second part of the analysis, the application is examined to determine whether it can be divided into configurations that can be executed sequentially thus suitable for global reconfiguration. It may, however, be that it is judged more suitable for partial reconfiguration, and tasks are then grouped or defined by modules. The process can either be performed manually, or by an automated scheduling approach including temporal partitioning and placement similar to Bobda [5]. The latter does however, require knowledge or estimates of execution time and resource usage. Those estimates have to be acquired by logic synthesis, as described in the next paragraph.

The determined configurations or modules are provided as input to a synthesis program to obtain estimates of execution time and area consumption. The reconfiguration time, $t_{\text{reconf},i}$, is estimated by dividing the bitstream size estimate by the speed of the configuration interface (up to 100 MHz using the Xilinx Virtex-4 SelectMAP interface [2]) as in

$$t_{\text{reconf},i} = \frac{W + 1312}{100} \quad [\mu\text{s}] \quad , \tag{5}$$

where W is the configuration array size of 147600, 726520, and 426810 for the LX15, LX80, and SX35 Virtex-4 FPGAs respectively [11]. In a similar manner, the data transferred between the configurations are quantified and divided by the read/write speeds of the external memory.

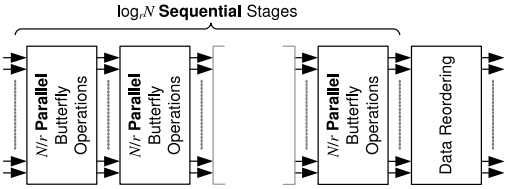

Fig. 4. Organization of an FFT.

Finally, the costs are calculated, and the use of reconfigurable architectures is deemed feasible if the conditions (6) and (7) are satisfied. The left hand side arguments in curly braces indicate that only one argument is considered at a time; This is determined by the selection of global or partial reconfiguration:

$$\{C_{\text{global}}, C_{\text{partial}}\} \leq C_{\text{static}} \quad \text{AND} \tag{6}$$
$$\{T_{\text{global}}, T_{\text{partial}}\} \leq T_{\text{deadline}} \ , \tag{7}$$

which ensures that the total cost is lower than or similar to the static implementation, and that the deadline is fulfilled.

In case T_{global} or T_{partial} are lower than T_{deadline}, it may be considered to utilize reconfiguration capabilities even further i.e. trade off execution time for area reduction, or select an architecture with a lower clock speed as idle resources are available.

III. CASE STUDIES

The previous sections described our proposed architecture model and how to do the application analysis. This is demonstrated by two case-studies in this section. The first study considers global reconfiguration, whereas the second study considers both global and partial reconfiguration.

A. Fast Fourier Transform

The Fast Fourier Transform (FFT) algorithm is widely used in multimedia applications and communications systems. In the latter case it is known as an efficient implementation of orthogonal frequency-division multiplexing (OFDM). An FFT is composed of parallel butterfly operation blocks that are executed sequentially followed by data reordering as illustrated in figure 4. N is the number of points in the FFT and r is the radix of the butterfly operations. The computation consists of $\log_r N$ sequential blocks of $\frac{N}{r}$ parallel radix-r butterfly operations.

The case is selected to be a 32 point radix-2 FFT ($N = 32$ and $r = 2$) operating at 16 bit resolution. The static reference is a fully parallel implementation with constant twiddle-factor multipliers synthesized for a Virtex-4LX80 FPGA with 35840 CLB slices [10] executing an FFT-operation at a rate selected to be 1 kHz. The reordering of data at the output is not considered for the static reference.

TABLE III
DETAILS OF CONFIGURATIONS FOR THE FPGA-BASED DAB RECEIVER [4]:

Configuration i	$t_{\text{exec},i}$ [ms]	Content
0	2.26	Mixer, FIR filter, and fine frequency offset correction
1	1.14	Fast Fourier Transform
2	0.48	Coarse freq. offset correction, demodulator, frequency and time deinterleaving
3	0.11	Viterbi decoding and energy dispersion

The alternative implementation is a globally reconfigurable solution at which each stage is implemented as a full configuration, theoretically making it possible to reduce the hardware area by a factor of $\log_r N = 5$. Reconfiguration time is estimated by (5) for a Virtex-4LX15 FPGA containing totally 6144 CLB slices [10], as the number of CLB-slices in this FPGA is close to 5 times smaller than the Virtex4-LX80.

The memory read and write times are estimated based on SDRAM memory running at 266 MHz, by using the expression

$$t_{read} = t_{write} = \frac{n_{\text{bytes}}}{4\text{bytes} \times 266\text{MHz}} + \frac{3}{266\text{MHz}} \quad , \quad (8)$$

where the last part of the sum is based on the latency of the memory. In this case study, the transferred amount of data, n_{bytes}, was 128 Bytes.

The static and globally reconfigurable implementations were synthesized in Xilinx ISE 9.1 based on VHDL code to obtain the necessary estimates.

B. FPGA-based DAB Receiver

The second case is based on a study of the results by Ihmig et al. [4]. The work consists of a digital audio broadcasting (DAB) receiver that is investigated for combining the tasks in a sequential execution on a Xilinx Virtex-4 SX 35 FPGA. The reference is a pipelined architecture consisting of 10 stages running at multiple rates, and is characterized by having a very relaxed latency requirement. The authors investigate a solution where the 10 stages are partitioned into four configurations, listed in table III, that are executed sequentially at a higher clock frequency (100 MHz) than the pipelined architecture (8.2 MHz).

The buffered sequential implementation is assumed based on a 50 Hz cycle, which determines the time, T_{static}, of the static implementation. In their work, the read/write time for external memory is not listed, and is therefore estimated as in (8). The transferred amount of data between configurations, n_{bytes}, is conservatively assumed based on the maximum data rate of 8192 kbytes/s, which gives 8192/50 kbytes between each configuration.

The static and global area were both determined by the size of the FPGA to 15360 CLB slices [10] and the reconfiguration time was estimated as described in (5).

The above referenced work also considers partial reconfiguration, where the four configurations are set to the size of the largest configuration of 2048 CLB slices. Reconfiguration time was given to be 750 μs, and $C_{\text{partial,comm}}$ was estimated by multiplying the memory controller area (668) by the total period of 20 ms.

TABLE IV
RESULTS: FAST FOURIER TRANSFORM:

Static Implementation (Virtex-4LX80)		
Time	$T_{\text{static}} = T_{\text{deadline}}$	1 ms
Area	$\{A_{\text{static}}$,synthesis$\}$	{35840, 24516} slices
Cost	C_{static}	35.8 s·slices
Globally Reconfigurable Implementation (Virtex-4LX15)		
Time	$\{T_{\text{exec}}, T_{\text{reconf}}, T_{\text{transfer}}\}$	{27.0E-6, 7.4, 1.32E-3} ms
Area	$\{A_{\text{global}}$,synthesis$\}$	{6144, 5815} slices
Cost	C_{global}	45.8 s·slices

TABLE V
RESULTS: FPGA-BASED DAB RECEIVER:

Static Implementation (Virtex-4SX35)		
Time	$T_{\text{static}} = T_{\text{deadline}}$	20 ms
Area	A_{static}	15360 slices
Cost	C_{static}	307 s·slices
Globally Reconfigurable Implementation (Virtex-4SX35)		
Time	$\{T_{\text{exec}}, T_{\text{reconf}}, T_{\text{transfer}}\}$	{4.4, 17.1, 0.63} ms
Area	A_{global}	15360 slices
Cost	C_{global}	340 s·slices
Partially Reconfigurable Implementation (Virtex-4SX35)		
Time	$t_{\text{reconf},0}, \ldots, t_{\text{reconf},3}$	750 μs
	$\{t_{\text{exec},0}, \ldots, t_{\text{exec},3}\}$	{2.26,1.14,0.48,0.11} ms
	T_{partial}	20 ms
Area	$A_0, \ldots, A_3$	2048 slices
	A_{busregs}	668 slices
Cost	$C_{\text{partial,proc}}$	15.1 s·slices
	$C_{\text{partial,comm}}$	13.4 s·slices
	C_{partial}	28.4 s·slices

IV. RESULTS

The results were obtained as described in section III. The results from the FFT case study are shown in table IV. A_{static} and A_{global} are the actual values for the FPGAs [10], whereas "synthesis" is the synthesis result obtained by ISE 9.1. In addition to CLB slices, DSP48 resources were also utilized. However, these are not included in the cost-model, thus not showed in the table.

From the synthesis results, it is clear that one FFT-stage only consumes 16% of the FPGA's resources in the full static implementation. However, due to the high reconfiguration overhead, the costs and time are higher, 28% and 540% respectively, than for the static reference.

For the second case of the DAB receiver, the results are shown in table V. The results are a combination of extracts from [4] and the estimates described in section III-B.

V. DISCUSSION

For the investigated FFT-case, the results clearly showed that a globally reconfigurable implementation had significantly higher costs than a static implementation, in spite of the possibility of HW sharing. The cost can be reduced by packing

more operations into each configuration, and thereby reduce the number of reconfigurations. However, this will not make the reconfigurable solution feasible for this case, as (6) and (7) still cannot be fulfilled. It can be argued that the investigated fully parallel FFT implementation is not a realistic reference and is inefficiently implemented in the FPGA. However, we find that the suggested scenario describes the problem of feasibility estimation for block-processing applications in an illustrative and easily understandable way.

For the investigated DAB-receiver case, the global reconfiguration did not fulfill the conditions (6) and (7), and it was thereby concluded that a globally reconfigurable implementation is not feasible compared to a static solution. This is mainly caused by the long time spent on reconfiguration as shown in table V. However, the reconfiguration time can be decreased by selecting a smaller FPGA - thus reducing the cost of the globally reconfigurable implementation.

The partially reconfigurable solution for the DAB-case did show a significant reduction in cost and only 9.3% of the resources were utilized. The rest of the resources can either be utilized for other functionalities or a smaller FPGA can be selected. The feasibility conditions (6) and (7) were fulfilled, so a partially reconfigurable implementation is feasible for this application.

The investigated cases show that a reconfigurable implementation may be feasible and may satisfy the time-constraints either due to a very relaxed deadline, or by running the reconfigurable architecture at a higher clock-speed than the non-reconfigurable implementation. Increasing the clock-speed leads to an increased power-consumption, thus we suggest extensive evaluation of power-consumption for future work.

An advantage of the methodology is that it is relatively simple to obtain the estimates and set up the feasibility conditions. However, it requires that the designer performs the partitioning of the application into configurations or modules and performs logic synthesis of these configuration or modules. The partitioning of the application can be performed by an automatic scheduling approach as suggested in section II-D.

So far, our methodology does only consider the CLB slices, but other conditions are currently being investigated for memory blocks and DSP slices.

VI. Conclusion

In this work we propose a method to evaluate the feasibility of implementing signal processing applications in reconfigurable architectures.

A general condition for feasibility of a globally reconfigurable architecture is closely related to the reconfiguration time and thus the size of the reconfigurable area. The size must be carefully selected so that the reconfiguration time does not exceed the execution time of the static configuration reference. However, as the reconfiguration time is potentially significantly smaller for partially reconfigurable implementations than for globally reconfigurable implementations it is generally preferable to choose a partially reconfigurable solution.

We conclude that the proposed cost-metric makes it possible to evaluate the feasibility considering area-usage and timing. An observation is that timing constraints may be fulfilled by adjusting the clock-speed, thus consideration of power-consumption in the cost-metric is suggested as future work.

References

[1] R. Hartenstein, "Trends in reconfigurable logic and reconfigurable computing," *9th International Conference on Electronics, Circuits and Systems*, vol. 2, September 2002, pp. 801–808.

[2] P. Lysaght et al., "Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of xilinx fpgas," *International Conference on Field Programmable Logic and Applications*, 2006.

[3] J. P. Delahaye et al., "Software radio and dynamic reconfiguration on a dsp/fpga platform," *3rd Karlsruhe Workshop on Software Radios*, 2004.

[4] M. Ihmig et al., "Resource-efficient sequential architecture for fpga-based dab receiver," *5th Karlsruhe Workshop on Software Radios*, 2008.

[5] C. Bobda, "Synthesis of dataflow graphs for reconfigurable systems using temporal partitioning and temporal placement," Doctor's dissertation, Faculty of Computer Science, Electrical Engineering and Mathematics of the University of Paderborn, May 2003.

[6] S. Hauck, "Configuration prefetch for single context reconfigurable coprocessors," *ACM/SIGDA sixth international symposium on Field programmable gate arrays*, pp. 65–74, 1998.

[7] M. J. Wirthlin and B. L. Hutchings, "Improving functional density using run-time circuit reconfiguration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, no. 2, pp. 247–256, June 1998.

[8] A. Shoa and S. Shirani, "Run-time reconfigurable systems for digital signal processing applications: a survey," *Journal of VLSI Signal Processing Systems*, vol. 39, no. 3, pp. 213–235, 2005.

[9] P. Manet et al., "Evaluation of dynamic partial reconfiguration in professional electronics applications," *DASIP Workshop on Design and Architectures for Signal and Image Processing*, November 2007.

[10] Xilinx Inc., "Virtex-4 FPGA User Guide," *UG070*, June 2008.

[11] Xilinx Inc., "Virtex-4 FPGA Configuration User Guide," *UG071*, April 2008.