ISSN 2278-3083

International Journal of Science and Applied Information Technology (IJSAIT), Vol.2 , No.2, Pages : 35-39 (2013)
Special Issue of ICCTE 2013 - Held during 11-12 March, 2013 in Hotel Crowne Plaza, Dubai

Mapping of Multiple Data Flow Graphs of DSP Applications onto ASIC/Reconfigurable Architectures

Awni Itradat, Thaier Hayajneh, Ahmad Qatoom
Department of Computer Engineering, Hashemite University, Zarqa, Jordan,
Email: itradat@hu.edu.jo

Abstract: This paper presents a novel technique for the mapping of set of DSP applications onto architectures targeting an ASIC/Reconfigurable implementation embedded on the same chip. Synthesis for such a hybrid implementation is carried out by developing a technique to partition the RTL structures corresponding to a set of DSP applications into a fixed base design part suitable for ASIC implementation and a non-base design that varies with the applications and suitable for FPGA implementation. Experimental results reveal that the proposed scheme is efficient in exposing the hidden functional commonality in a set of RTL structures respecting some well-known benchmark problems. We show through a set of test cases that our approach offers significant area saving relative to the state-of-the-art.

Key words: DSP Data flow graphs, RTL structure, Graph merging, ASIC/reconfigurable platform

INTRODUCTION

The DSP applications are among the most important applications that demand high computational power, and must be executed at a very high speed to enable real-time processing. Due to the parallelism within the DSP applications, parallel processing architectures are a natural choice for the synthesis of these applications. Moreover, the need to reduce the design time of digital systems, and thus, the time to market is increasingly crucial factor to have a competitive edge. Such an advantage can be achieved if there are efficient techniques to reduce the time taken by each of the individual steps of the design process. The design cycle of a digital system is composed of three major steps: RTL design, Physical implementation and Verification or Validation at various levels of design and implementation. There exist a variety of architectural synthesis techniques, most of which target DSP algorithms onto multiprocessor architecture using basic functional units such as adders or multipliers [1][2]. This however results in inferior implementation of DSP applications, since the regularity feature of DSP applications are not exploited.

The most popular hardware implementations of DSP applications are: (1) application specific integrated circuit (ASIC), (2) field programmable gate arrays (FPGAs), or (3) a set of instructions running on an application-specific processor. Various implementations of the same application allow trade-offs for optimizing the hardware in terms of multiple design parameters such as power consumption, area, processing speed, and re-configurability of the system. There are a number of advantages and disadvantages for these three implementations. An ASIC implementation has fully customized data paths and logic. It allows designers to optimize the hardware resources for one or more of the design parameters. However, traditional ASIC implementation is not flexible, since it does not allow reconfiguring itself and cannot be used in a wide range of applications. FPGAs, on the other hand, consist of arrays of prefabricated logic blocks. FPGAs can provide a reconfigured implementation of a certain design. The property of re-configurability allows the designers to reuse the resources in variety of applications. Although FPGAs have the capability of programming functional units and wires, it has several inherent limitations. FPGAs usually consume much higher power than an ASIC implementation. They also have higher performance penalty and require larger silicon area because of their generic platform. Another common method to implement a DSP application is to use application-specific processors such as a DSP processor. DSP processors are designed for general-purpose; and hence, they are not area, performance, and power efficient.

Furthermore, hardware flexibility is a crucial factor in today's system design. However, such flexibility must not be gained at the expense of performance and area, as is the case with general-purpose reconfigurable fabrics such as field programmable gate-arrays (FPGAs) [3]. Hybrid FPGAs and reconfigurable cores provide hardware flexibility, their coarse integration of fixed logic and reconfigurable fabric results in performance, area and power penalties [4]. New techniques have therefore been explored to add flexibility to individual hardware components without the penalties associated with FPGAs. To overcome the latter penalties, small-scale reconfiguration would minimizes the area and delay penalties by inserting into fixed-logic only the minimum amount of reconfigurable logic and interconnect and by reusing the main part of the available logic and by changing the status of a few control signals to achieve the desired component functionality. A great deal of research work have explored numerous methods to reduce the gap between ASIC and FPGAs implementation, thus merging the advantages of both the worlds. Several researchers have investigated more efficient utilization of FPGA resources to achieve improvement in performance and area. Dynamically reconfigurable FPGA systems use a dynamic allocation scheme that reallocates resources at run-time to achieve higher performance. Zhang and Ng [5] have surveyed synthesis techniques for dynamically reconfigurable systems. On the other hand, in [6] authors have introduced a coarse-grained FPGA architecture that allows the designers to customize FPGAs for a specific applications.

In [7] it has been tried to find common substructures among different configurations and to generate a processing unit that can run in various configurations. They use iterative improvement and simulated annealing to minimize the interconnection cost. Reference [8] has developed behavioral synthesis techniques that may be extended to reconfigurable ASICs. The original technique uses built-in self repair (BISR) to dynamically replace a faulty module by another heterogeneous module, thus providing a fault-tolerant design. If one considers the operation under a faulty situation as one of the possible operating configurations, their techniques are applicable to reconfigurable ASICs.

In this paper, a scheme for the architectural synthesis of a set of DSP applications targeting an ASIC/reconfigurable architecture is proposed. The hidden functional commonality among a set of DSP applications are identified by employing the so called graph merging leading to dividing the final RTL structure of multiple behaviors into two parts, an ASIC-based fixed structure, and an FPGA-based structure.

METHODOLOGY

The high-level synthesis, namely, the tasks of scheduling and allocation are used for the transformation of a behavioral description of an algorithm into an RTL structure that consists of a set of connected components. The concept of design reuse allows a previous design to be reused in a new design, since the previous design may still meet the requirements of the new design. The design reuse can increase the design efficiency leading to lower design cost and a decreased time to turn a design to market. The cost of testing of the entire system can also be reduced, since the part of the previous test scheme can also be employed in the test of the overall new design. The concept of design reuse is employed in our context to create a “base design” that implement a substantial number of functions from a set of applications and use this base design with minimal or no changes by adding to it the implementation of the remaining functions of an application in order to complete the overall design of a specific application. We will refer to the design corresponding to the implementation of the remaining functions of an application as "non-base design", since this part of the design will vary from application to application. Digital image and signal processing applications are examples in which this concept could be useful.

The need for changing the hardware configuration supporting multiple designs should compel the designers to focus on developing a scheme in which an ASIC-based base design may be embedded with an FPGA-based non-base design leading to the implementation of a reconfigurable system implementing a number of applications on the same chip. With the incorporation of the FPGA module together with the ASIC module on the same chip, it would be possible to use the programmability of the former to realize various DSP applications. This approach would, therefore, greatly reduce the effort and the cost associated with multiple designs each focusing only on a single application.

An example of embedding a fixed ASIC-based modules into an FPGA-based platform can be found in the XilinxR Virtex-5 DSP48E in which there is blocks to support many independent functions, including multiplier-accumulator (MAC), multiplier followed by adder, three-input adder, barrel shifter, and support for pipeline parallelism. For example, if we want to implement a DSP filter targeting this platform, we should consider using these specific embedded ASIC-based blocks. In order to achieve that, we must find the possible sub-graphs in the CDFG that are isomorphic to these modules. Detecting these structures in the graph is a crucial task since otherwise the final implementation would be inefficient and expensive or unrealizable.

One of the key challenges associated with the proposed scheme is as to how to identify efficiently the part of the logic functions from a set a reasonable number of applications which can be resorted as the case design. This work is aimed at identifying the base design from the RTL structures of a set of DSP applications. In DSP applications, DFG’s are generally constructed out of several, commonly used “building blocks” like dot-product, butterfly, etc. In order to extract RTL-base architecture for a set of DSP application, one can explore such commonly used isomorphic building blocks between the given set of applications. The techniques presented here identify candidate behaviours for mapping on to a single RTL module and merge the selected RTL modules into one. We now describe how we identify candidate RTL modules for merging, and then how to merge the selected RTL modules into one.

SCHEME FOR MAPPING OF DSP APPLICATIONS ONTO AN ASIC/RECONFIGURABLE PLATFORM

The data flow graph (DFG) is proven to be an efficient representation of the system specification due to its ability to expose the hidden concurrency between the operations of the underlying algorithm. Since DSP applications are known for their inherent parallelism, the DFG model is suitable for the DSP applications. Moreover, regularity is an inherent feature of several VLSI systems, especially those we find in signal processing applications. Data flow graph representations is an efficient to exploit such regular characteristics of such applications. In the proposed scheme, a datapath of a DSP application i is modeled as a directed graph $Gi\ (Vi\ , Ei\ )$, where the vertices in Vi represent the hardware blocks in the datapath, and the edges in Ei are associated with the interconnections between the hardware blocks. The types of hardware blocks (e.g., adders, multipliers, registers, and so on) are modeled by a labeling function Li of Vi , such that, for each vertex $u \in Vi$, $Li(u) \in Typij$ is a label that represents the type of the hardware block associated with u. More specifically, we say that vertex u in graph Gi is associated with the j th hardware block of type T.

The proposed technique for extracting the RTL ASIC-based for a set of DSP application is build based on a modified version of the DFG merging technique given in [9]. To make this paper self-contained, the scheme in [9] is summarized as follows. Given a data flow graphs representations of a given set of RTL structures representing a set of DSP applications, the presented scheme start first by building a compatibility graph H for the two graphs G_i and G_j, where each vertex of H corresponds to a possible mapping of two edges, one from G_i and another from G_j. There exists an edge (arc) between two vertices of H if the arc mappings represented by the vertices are compatible. In order to build the compatibility graph, the notion of mapping compatibility has been defined. Two arc mappings are incompatible if and only if they map the same vertex of G_i to two different vertices of G_j, or vice-versa. In order to determine the resulting minimum base graph (RTL-based), the technique find the maximum number of arc mappings that are compatible to each other. This can be achieved by apply a heuristic solution to compute the maximum clique of the weighted compatibility graph H.

Since we are looking for a method leading to partitioning of the RTL structures of a set of DSP applications obtained into two parts. The first part consists of the RTL components that are common to all the applications and suitable for ASIC-base, whereas the second part consists of those that differ from application to application in the given set, and can be implemented in FPGA (reconfigurable). In our proposed scheme, the weighted graph is modified by incorporating to the weights a metric for the usage-frequency of an RTL component which counts the number of mappings between behavior operations or nodes and RTL component. In other words, the usage-frequency for a component is measured by the number of operations this component is used by a given set of applications according to the scheduling and resource allocation. The RTL components that have a higher usage-frequency represent a strong demand in the given set of DSP applications, and therefore, need to be included in the base design. Hence an ASIC-based RTL structure should be composed of these high usage frequency RTL components.

By using above modification, the proposed technique divides then the final RTL structure of multiple behaviors into two parts, an ASIC-based fixed structure which includes those high usage-frequency RTL components, and an FPGA-based structure corresponding to the low-usage frequency components. The number of low usage-frequency components can be used to identify the size of FPGA segment of the hybrid chip.

In Fig. 1, three RTL structures represent three DSP algorithms are shown. By applying the proposed scheme an RTL ASIC-based structure is obtained as shown in that fig 1. It seen in this example that only common RTL components are only included in the ASIC-based structure. On the other hand the RTL structures that differ from one application to the other are suitable to be mapped into an FPGA-based structure embedded on the same chip. Hence they are not included in the base RTL structure.

EXPERIMENTAL RESULTS

To show the effectiveness of our approach, we have made several experiments with some well-known set of DSP applications commonly used in the literature of architectural synthesis. In order to synthesis the obtained RTL structures,

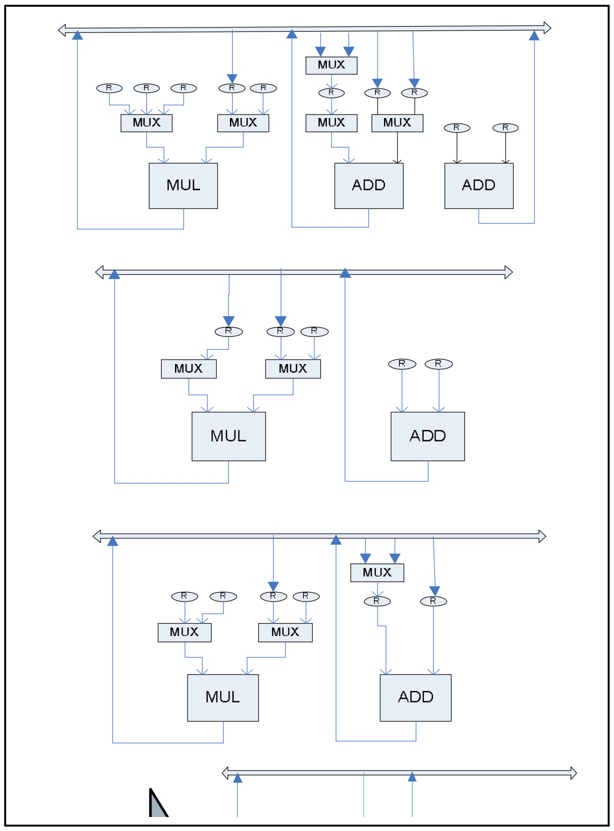

Fig. 1: An RTL ASIC-Based Structure for a Set of DFGs

Table 1: Different Sets of DSP applications conducted in the experiments

LMS16, FIR16	SET1
FIR19, FIR15, FIR11, FIR7	SET2
AR,FDCT, FIR	SET3
ELLIP, EDGE	SET4
HOUGH-TRNS, BIQUAD, FFT, PFILTER	SET5

we have used the Synopsys Design Compiler (DC) synthesis tool, which performs different levels of optimization, i.e., architectural, logic- and gate-level optimization. Moreover, two approaches are considered in our experiments. The first approach is by finding a combined RTL structure (comb-RTL) for a set of DSP applications, i.e., an RTL structure which is obtained by simply adding the individual RTL structure without optimization or merging. The second approach (the proposed one) in which the resulting common RTL structure is only ASIC-based RTL structure and the RTL structures that differ from one application to the other is mapped to FPGA-based structure. Different set of DSP applications are used in the conducted experiments. Table I shows the different sets of DSP applications from the literature used in our experiments.

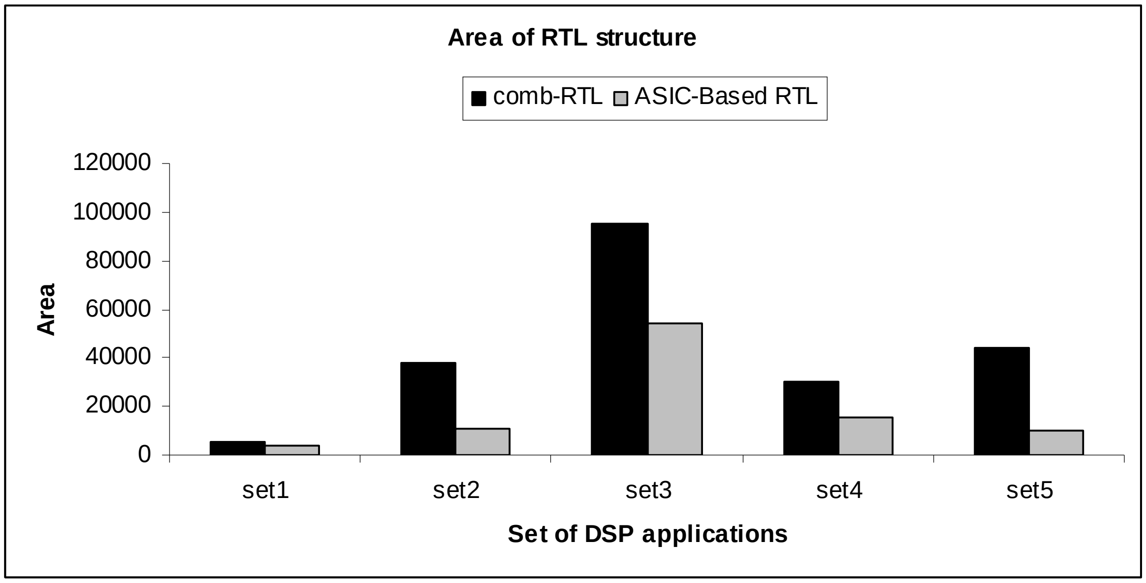

Fig. 2: Area of ASIC-based RTL Structure versus that Obtained by a Combined RTL Structure for Sets of DSP Data Flow Graphs

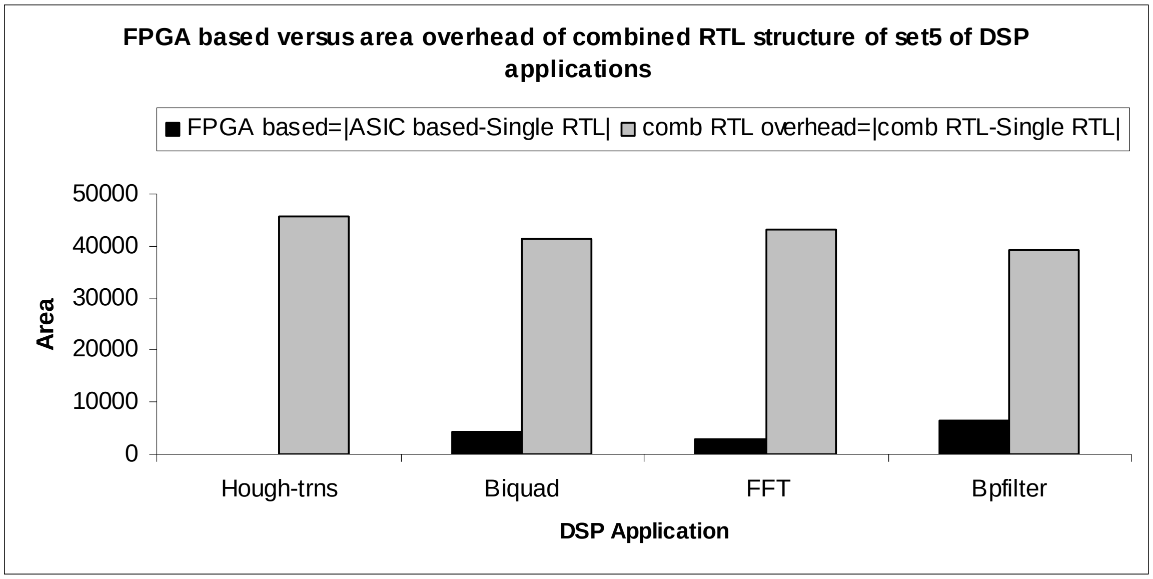

Fig. 3: The FPGA-based RTL Required for each Individual DSP Application in Set5 versus Area Overhead of the Combined RTL with Respect to the Single RTLs

We choose five sets of DSP applications for our experiments. The five sets are all well known benchmarks in the literature of the high-level synthesis (set1=LMS16, FIR16, set2=FIR19, FIR15, FIR11, FIR7, set3=AR, FDCT, FIR, set4= ELLIP, EDGE, set5=HOUGH-TRNS, BIQUAD, FFT, PFILTER). An ASIC-based RTL structure is extracted for each set of applications and then compared, as shown in Fig. 2, with the comb-RTL. It is seen from Fig. 2 that a significant saving in terms of the area is obtained by using the proposed approach. Moreover, it is also shown in Fig. 3 that the area required to implement the FPGA-based structure for each individual application (Single RTL) in set5 is much less than area overhead (with respect to each single RTL in the set) resulting from using the combined RTL approach to synthesis set5 of applications.

CONCLUSION

The need for mapping of a set of DSP applications onto reusable hardware platform of ASIC/Reconfigurable implementation embedded on the same chip has been advanced. It has been shown that the challenge of accomplishing this lies in identifying a base design for a set of DSP applications targeting an ASIC implementation on the hybrid chip. A novel method using graph merging has been proposed by developing a technique to partition the RTL structure of multiple behaviors into two parts, an

ASIC-based fixed structure which includes high usage-frequency RTL components, and an FPGA-based structure corresponding to the low-usage components. Experimental results reveal through a set of test cases that our approach offers significant area saving relative to the state-of-the-art (comb-RTL).

REFERENCES

[1] Y.-N. Chang, C.-Y. Wang, and K. K. Parhi. “Loop-list scheduling for heterogeneous functional units,” in *Proc.6th Great Lakes Symposium on VLSI*, pp. 2–7, March 1996.

[2] Z. Shao, Q. Zhuge, C. Xue, and E.H.-M. Sha “Efficient assignment and scheduling for heterogeneous DSP systems**,”** *IEEE Tran. on Parallel and Distributed Systems* vol. 16, pp. 516-525, June 2005.

[3] K. Compton and S. Hauck. “Reconfigurable computing: A survey of systems and software,” *ACM Computing Surveys*, vol 34, no. 2, 2002, pp.171–210.

[4] R. Tessier and W. Burleson., “Reconfigurable Computing for Digital Signal Processing: A Survey,” *Journal of VLSI Signal Processing*, vol 28, no. 1, pp.7–27, 2002.

[5] X. Zhang, K.W. Ng. "A review of high-level synthesis for dynamically reconfigurable FPGA," in *Proc Microprocess-Microsystems*, pp. 199–211, 2000.

[6] J. Anderson, S. Sheth, and K. Roy. "A coarse-grained FPGA architecture for high-performance fir filtering," in *Proc. of ACM/SIGDA 6th International Symposium on Field Programming Gate Arrays*. pp. 234–243, 1998.

[7] V.A Derwerf, E. Aerts, M. Peek., and W. Verhaegh. "Area optimization of multifunction processing units," in *Proc. International Conference on Computer Aided Design*, pp. 292–299, 1992.

[8] L. Guerra, M. Potkonjak, and J. Rabaey. "Behavioral-level synthesis of heterogeneous BISR reconfigurable ASIC’s," *IEEE Trans. VLSI Systems.* vol. 6, no.1, pp. 158–167, March, 1998.

[9] Z. Huang and S. Malik, “Managing dynamic reconfiguration overhead in systems-on-a-chip design using reconfigurable datapaths and optimized interconnection networks,” in *Proc. Design Automation Test Eur. Conf.*, 2001, pp. 735–740.