CDFG Merging for Reconfigurable Architectures

Nahri Moreano *Guido Araujo*
Cid C. de Souza

Technical Report - IC-03-018 - Relatório Técnico

October - 2003 - Outubro

CDFG Merging for Reconfigurable Architectures

Nahri Moreano[*†] Guido Araujo[*] Cid C. de Souza[*]

Abstract

Reconfigurable systems have been proved to achieve significant performance speedup through architectures that map the most time-consuming application kernel modules or inner-loops to a reconfigurable datapath. As each portion of the application starts to execute, the system reconfigures the datapath so as to perform the corresponding computation. The reconfigurable datapath should have as few and simple hardware blocks and interconnections as possible, in order to reduce its cost, area, power consumption, and reconfiguration overhead. Thus hardware blocks and interconnections should be reused across the application as much as possible. We represent each piece of the application as a control/data-flow graph (CDFG) and merge them together, synthesizing a single reconfigurable datapath. The CDFG merging process enables the reuse of hardware blocks and interconnections by identifying similarities among the CDFGs, and producing a resulting datapath that can be dynamically reconfigured to work for each CDFG and has a minimum area cost, when considering both hardware blocks and interconnections. In this report we formalize the CDFG merge problem and we present a proof that it is NP-complete, by reducing the subgraph isomorphism problem to it.

1 Introduction

It is well known that embedded systems must meet strict constraints of high-throughput, low power consumption and low cost, specially when designed for signal processing and multimedia applications [5]. These requirements lead to the design of application specific components, ranging from specialized functional units and coprocessors to entire ASIP processors. Such components are designed to exploit the peculiarities of the application domain in order to achieve the necessary performance and to meet the design constraints.

With the advent of reconfigurable systems, the availability of large/cheap arrays of programmable logic has created a new set of architectural alternatives for the design of complex digital systems. Reconfigurable logic brings together the flexibility of software and the performance of hardware [1]. As a result, it became possible to design application specific components, like specialized datapaths, that can be reconfigured to perform a different computation, according to the the specific part of the application that is running (for instance kernel modules and/or inner loops of the application). At run-time, as each portion

[*]Institute of Computing, University of Campinas, 13083-970 Campinas, SP

[†]Department of Computing and Statistics, Federal University of Mato Grosso do Sul, 79070-900 Campo Grande, MS

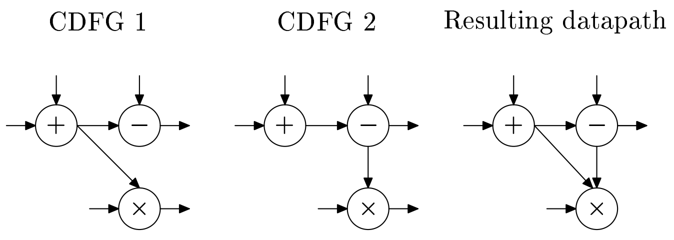

Figure 1: Control/data-flow graph merging

of the application starts to execute, the system reconfigures the datapath so as to perform the corresponding computation. Recent work in reconfigurable computing research has shown that a significant performance speedup can be achieved through architectures that map the most time-consuming application kernel modules or inner-loops to a reconfigurable datapath ([4, 3]).

The reconfigurable datapath should have as few and simple hardware blocks (functional units and registers) and interconnections (multiplexors and wires) as possible, in order to reduce its cost, area, and power consumption. Thus hardware blocks and interconnections should be reused across the application as much as possible. Resource sharing has also crucial impact in reducing the system reconfiguration overhead, both in time and space.

To design such a reconfigurable datapath, one must represent each selected piece of the application as a control/data-flow graph (CDFG) and merge them together, synthesizing a single reconfigurable datapath. The control/data-flow graph merging process enables the reuse of hardware blocks and interconnections by identifying similarities among the CDFGs, and producing a resulting datapath that can be dynamically reconfigured to work for each CDFG. Ideally the resulting datapath should have the minimum area cost, when considering both hardware blocks and interconnections.

Figure 1 illustrates the concept of control/data-flow graph merging. When CDFGs 1 and 2 are merged, one possible resulting datapath produced is shown in the figure[1]. Notice that in the resulting datapath there are interconnections originated from only one CDFG (e.g., the $(+,\times)$ interconnection from CDFG 1) and interconnections shared by both CDFGs (e.g., the $(+,-)$ interconnection).

In this report we formalize the CDFG merge problem and we present a proof that it is NP-complete, by reducing the subgraph isomorphism problem to it.

This report is organized as follows. In the next section we describe our datapath architecture model. Section 3 presents the problem more formally, exposing its difficulty. We present in Section 4 the proof that the CDFG merge problem is NP-complete. Finally, Section 5 concludes the work.

[1]For simplicity, the multiplexor which selects the multiplier input from the adder or the subtractor, is not showed in the figure.

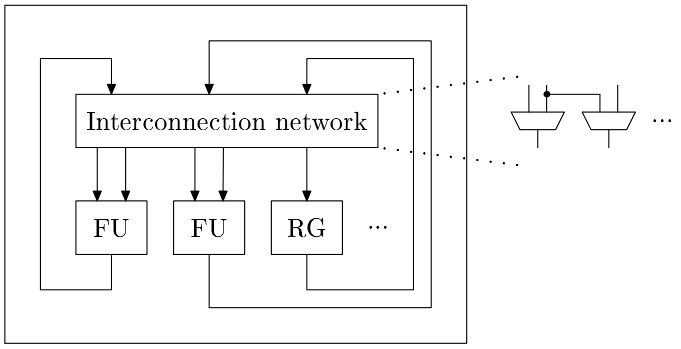

Figure 2: Architecture model

2 Architecture Model

The datapath architecture model used in this work consists of a set of functional units (FUs) and registers (RGs) organized around an interconnection network forming a programmable datapath, as shown in Figure 2. The interconnection network is based on a set of multiplexors (MUXes) that select the input data for functional units and registers.

As the computation progresses, the system reconfigures the datapath through the interconnection network, such that computational intensive pieces of the application are mapped onto it. Given the (coarse) granularity of the logic blocks (FUs and RGs), the number of bits required to encode them is much smaller than in the case of fine-grained architectures. As a result, fewer bits are needed to reconfigure the datapath, thus diminishing the size of the memory required to store the reconfiguration bits (the so called reconfiguration *context*). This is a central issue in SoC (System-on-a-Chip) designs where on-chip area is a premium asset. Moreover, the smaller the size of the context, the smaller the time overhead required for reconfiguration. Reconfiguration time is a critical feature in such systems, given that the final performance is determined by the sum of the computation time and the reconfiguration latency (if latency hidden techniques are not used).

3 The Control/Data-flow Graph Merge Problem

In this section we formulate the CDFG merge problem more formally. We want to merge several CDFGs (corresponding to application portion), in order to build a reconfigurable datapath which is capable of performing the computation of each portion, multiplexed in time, and has the minimum area cost of hardware blocks (functional units and registers) and interconnections. Each application portion i, $i = 1 \ldots n$ is modeled as a CDFG G_i, as defined below.

Definition 1 *A control/data-flow graph (CDFG) is a directed graph $G = (V, E)$, where:*

- *A vertex $v \in V$ represents an operation or a variable. Each vertex v has a set of input ports $ip = 1 \ldots n\_inports_v$ and attributes specifying its type and width (in bits).*

- *An arc $e = (u, v, ip) \in E$ indicates a data transfer from vertex u to the input port ip of vertex v.*

Given a vertex of a CDFG, there may be (in the component library) several hardware blocks where it can be executed.

Definition 2 *The set of hardware blocks $HB(v)$ of a CDFG vertex v contains the hardware blocks from the component library which can perform the computation represented by v.*

The resulting reconfigurable datapath is the merge of all CDFGs G_i, $i = 1 \ldots n$ and is modeled as a merged graph $\bar{G}$, as defined below. The merged graph $\bar{G}$ is the overlapping of all G_i, such that only vertices which can be implemented by the same hardware block can be overlapped.

Definition 3 *A merged graph, corresponding to CDFGs G_i, $i = 1 \ldots n$, is a directed graph $\bar{G} = (\bar{V}, \bar{E})$, where:*

- *A vertex $\bar{v} \in \bar{V}$ represents a mapping of $n\_map_{\bar{v}}$ vertices v_i, $1 \leq n\_map_{\bar{v}} \leq n$, each one from a different V_i, such that $\bigcap_{\forall v_i} HB(v_i) \neq \emptyset$.*

- *An arc $\bar{e} = (\bar{u}, \bar{v}, \bar{ip}) \in \bar{E}$ represents a mapping of $n\_map_{\bar{e}}$ arcs $e_i = (u_i, v_i, ip_i)$, $1 \leq n\_map_{\bar{e}} \leq n$, each one from a different E_i, such that all u_i have been mapped on $\bar{u}$, all v_i have been mapped on $\bar{v}$, and all ip_i match*[2].

The reconfigurable datapath will have one hardware block for each vertex $\bar{v}$ in $\bar{V}$. This hardware block is capable of performing the computation represented by all vertices v_i mapped on $\bar{v}$. Also, for each arc $\bar{e} = (\bar{u}, \bar{v}, \bar{ip})$ in $\bar{E}$, there will be in the reconfigurable datapath a "path" connecting the two hardware blocks corresponding to $\bar{u}$ and $\bar{v}$, more specifically, going from the output of the former to the input port $\bar{ip}$ of the latter. Moreover, for each input port $\bar{ip}$ of each vertex $\bar{v}$ which has more than one incoming arc $(*, \bar{v}, \bar{ip})$, the reconfigurable datapath will have a MUX selecting the input operand.

We describe the vertex mapping represented by a vertex $\bar{v} \in \bar{V}$ with a n-tuple $(v_1/v_2/\ldots/v_n)$ enumerating the vertices $v_i \in V_i$ mapped on $\bar{v}$. If there is no vertex from a given V_i mapped on $\bar{v}$, the corresponding element in the n-tuple is empty, as, for instance, in $(-/v_2/\ldots/v_n)$. Similarly, the arc mapping represented by an arc $\bar{e} \in \bar{E}$ is described by a n-tuple $(e_1/e_2/\ldots/e_n)$ enumerating the arcs $e_i \in E_i$ mapped on $\bar{e}$, with empty elements whenever necessary.

Given a set of n input CDFGs G_i, it is possible to build several different merged graphs $\bar{G}$ corresponding to them. The optimal solution for $\bar{G}$ is the one which produces the reconfigurable datapath with minimum area cost, considering both hardware blocks and interconnections.

The area cost of the reconfigurable datapath generated from a merged graph $\bar{G}$ is the sum of hardware block area cost and interconnection area cost. The hardware block area cost is the sum of the area cost of all hardware blocks of the datapath, in the component library.

[2]The meaning of matching input ports will be further elaborated in Subsection 3.1.

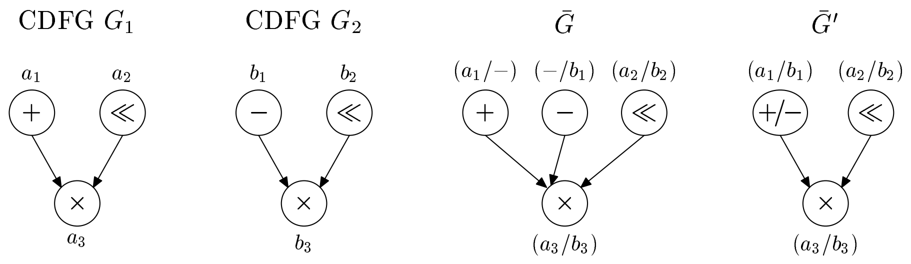

Figure 3: CDFGs G_1 and G_2 and two different merged graphs $\bar{G}$ and $\bar{G}'$

Each hardware block (corresponding to a vertex $\bar{v}$ of $\bar{V}$) is selected from the component library accordingly to the vertices v_i mapped on $\bar{v}$. For example, given the CDFGs G_1 and G_2 in Figure 3, we can build two different merged graphs $\bar{G}$ and $\bar{G}'$. In $\bar{G}$, vertices a_1 from G_1 and b_1 from G_2 are not mapped, so the reconfigurable datapath corresponding to $\bar{G}$ would have four hardware blocks (the functional units adder, subtractor, shifter, and multiplier). In $\bar{G}'$, those vertices are mapped, resulting in a reconfigurable datapath with three hardware blocks (an adder/subtractor, a shifter, and a multiplier).

Since in our architecture model the interconnection network is based on MUXes, the interconnection area cost is proportional to the number of MUX inputs. For each arc $\bar{e} = (\bar{u}, \bar{v}, \bar{ip}) \in \bar{E}$ which is a mapping of $n\_map_{\bar{e}}$ arcs e_i from CDFGs G_i, the MUX (if exists) in the input port $\bar{ip}$ of hardware block $\bar{v}$ has $n\_map_{\bar{e}} - 1$ fewer inputs than it would have if no arcs were overlapped. In Figure 3, the reconfigurable datapath corresponding to the merge graph $\bar{G}$ would have a MUX in the first input port of the multiplier, selecting the input operand from the adder or subtractor results. In $\bar{G}'$, since the vertices addition and subtraction are mapped to the same hardware block, it also became possible to map (a_1, a_3) onto (b_1, b_3) from CDFGs G_1 and G_2 respectively, thus eliminating the need for the MUX. Regarding each MUX as a tree of 2-input MUXes, there is a linear dependency between the number of 2-input MUXes and the number of wires, so the interconnection area cost can be expressed in terms of the number of wires.

The area cost of the reconfigurable datapath is defined below.

Definition 4 *Given a merged graph $\bar{G} = (\bar{V}, \bar{E})$, let $A_{hb}(\bar{v})$ be the area cost of the hardware block allocated to $\bar{v} \in \bar{V}$, and let A_{mux} be the area cost equivalent to one MUX input of the suitable width, both determined by the component library. The total area cost $A(\bar{G})$ of the reconfigurable datapath corresponding to $\bar{G}$ is:*

$$A(\bar{G}) = A_{hb}(\bar{G}) + A_{ic}(\bar{G})$$

where $A_{hb}(\bar{G}) = \sum_{\forall \bar{v} \in \bar{V}} A_{hb}(\bar{v})$ and $A_{ic}(\bar{G}) = |\bar{E}| \times A_{mux}$ are the hardware block and interconnection area cost, respectively, of the reconfigurable datapath.

We can now define the CDFG merge problem, as follows.

Definition 5 *Given n input CDFGs G_i, $i = 1 \dots n$, and a component library, find the corresponding merged graph $\bar{G}$, such that $A(\bar{G})$ is minimum.*

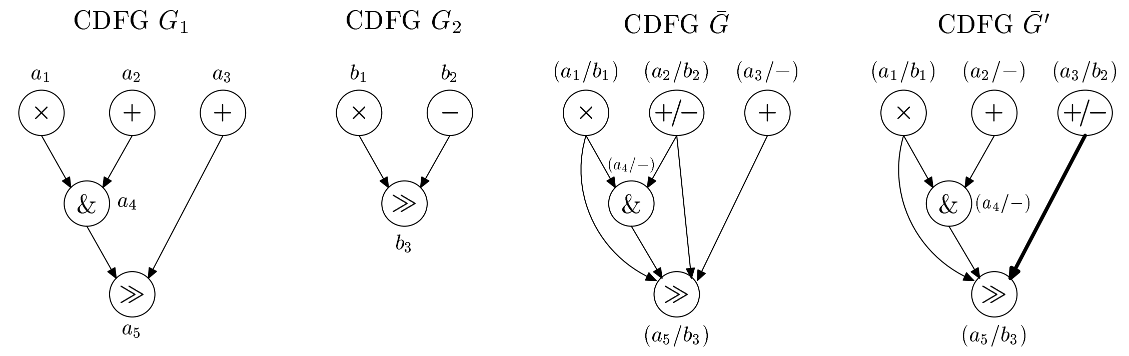

Figure 4: CDFGs G_1 and G_2 and two different merged graphs $\bar{G}$ and $\bar{G}'$: $A_{hb}(\bar{G}) = A_{hb}(\bar{G}')$, but $A_{ic}(\bar{G}) > A_{ic}(\bar{G}')$

Finding a mapping of the vertices from the CDFGs so as to minimize the hardware block area cost is not a difficult task. On the other hand, mapping the arcs from the CDFGs so as to minimize the interconnection area is a hard problem because the mapping of arcs depends on the mapping of their adjacent vertices. That is, two arcs from two CDFGs can only be mapped if their source vertices are mapped as well as their destination vertices. So, if we map vertices without considering the interconnection costs or using only estimates, we may get a solution where the interconnection area cost is not minimized and consequently, the total area cost is also non-optimal. For example, Figure 4 shows two different merged graphs $\bar{G}$ and $\bar{G}'$ obtained from CDFGs G_1 and G_2. $\bar{G}$ and $\bar{G}'$ represent different vertex mappings. In $\bar{G}$ vertex b_2 of G_2 is mapped onto vertex a_2 of G_1, while it is mapped on a_3 in $\bar{G}'$. The vertex mappings represented by $\bar{G}$ and $\bar{G}'$ may appear equivalent and, as a matter of fact, $A_{hb}(\bar{G})$ is equal to $A_{hb}(\bar{G}')$. But they allow for different arc mappings. In $\bar{G}$, no arcs are overlapped, so two MUXes are needed at the two input ports of vertex a_5/b_3. In $\bar{G}'$, the arcs (a_3, a_5) and (b_2, b_3) are mapped (highlighted in the figure), thus eliminating the need for one of the MUXes. As a result, $A_{ic}(\bar{G})$ is larger than $A_{ic}(\bar{G}')$ and consequently, $A(\bar{G})$ is also larger than $A(\bar{G}')$.

In order to compute the optimal solution for $\bar{G}$ we have to find out which vertex mapping, among several possibilities, gives the best arc mapping, i.e., which mapping minimizes the total area cost.

3.1 Input Ports and Commutativity

Several two-input operations are commutative, so properly exchanging the sources of these operations can enable arc mappings that would not exist otherwise, thus eliminating wires and MUXes and reducing the interconnection area cost of the reconfigurable datapath. Each operation has a set of input ports which represent the input operands it expects, for instance an addition has two input ports ip_1 and ip_2. From Definition 3, two arcs $(u_i, v_i, ip_i) \in G_i$ and $(u_j, v_j, ip_j) \in G_j$ can be mapped if u_i and u_j can be mapped, as well as v_i and v_j, and ip_i and ip_j match. The input ports ip_i and ip_j match if: (a) they are equal; or (b) v_i and/or v_j represent two-input commutative operations.

4 Proof of NP-Completeness

In this section we prove that the CDFG merge problem is NP-complete. We use the decision version of the problem in the proof, which is defined below.

Definition 6 *Given n input CDFGs $G_i = (V_i, E_i)$, $i = 1 \ldots n$, an integer k, and a component library with integer area costs, the CDFG merge decision problem (MERGE) consists in determining if there is a corresponding merged graph $\bar{G} = (\bar{V}, \bar{E})$, such that $A(\bar{G}) \leq k$.*

In order to prove that MERGE is NP-complete, we must show that MERGE belongs to NP and that it is NP-hard. We prove that MERGE is NP-hard by reducing the subgraph isomorphism problem (applied to directed graphs), which is NP-complete [2, GT48], to it. The subgraph isomorphism problem is defined below.

Definition 7 *Given the directed graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the subgraph isomorphism problem (ISO) consists in determining if G_1 contains a directed subgraph $H = (V_H, E_H)$ isomorphic to G_2, i.e., there exists a bijective function $f : V_2 \to V_H$, such that $(u, v) \in E_2 \Leftrightarrow (f(u), f(v)) \in E_H$.*

We now present the following theorem and its proof.

Theorem 1 *The MERGE problem is NP-complete.*

Proof: MERGE belongs to NP since we can construct, non-deterministically, and verify the merged graph $\bar{G} = (\bar{V}, \bar{E})$ in polynomial-time. We need to guess, for each $v \in V_i$, the vertex $\bar{v} \in \bar{V}$ on which it is mapped, and, for each $e \in E_i$, the arc $\bar{e}$ on which it is mapped, for all $i = 1, \ldots, n$. In the verification phase, we check, for each $\bar{v} \in \bar{V}$, if the vertex mapping it represents is valid, that is, if $\cap_{\forall v \text{ mapped on } \bar{v}} HB(v) \neq \emptyset$. Similarly, we check, for each $\bar{e} \in \bar{E}$, if the arc mapping it represents is valid, that is, if for all $e = (u, v, ip)$ mapped on $\bar{e}$, ip's match. Finally, we check if $A(\bar{G}) \leq k$.

To prove that MERGE is NP-hard, we transform an arbitrary instance of ISO, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, into an instance of MERGE. We establish $n = 2$, construct CDFGs $G'_1 = (V'_1, E'_1)$ and $G'_2 = (V'_2, E'_2)$ corresponding to G_1 and G_2, respectively, and determine k and the component library, such that:

- $\forall v \in V_i$, there is a vertex $v' \in V'_i$, $\forall i = 1, 2$, representing a commutative operation and with $n\_inports_{v'}$ equal to the in-degree of v;

- $\forall e = (u, v) \in E_i$, there is an arc $e' = (u', v', ip) \in E'_i$, $\forall i = 1, 2$, with ip's assigned sequentially from 1 to $n\_inports_{v'}$ for all arcs of the form $(*, v) \in E_i$;

- The component library has only one hardware block with area cost 1, which can perform the computation represented by all $v' \in V'_i, \forall i = 1, 2$. Moreover, A_{mux} is 1.

- $k = A(G'_1) = |V_1| + |E_1|$.

We claim that G_1 has a subgraph isomorphic to G_2 if and only if there is a merged graph $\bar{G} = (\bar{V}, \bar{E})$ corresponding to G'_1 and G'_2 such that $A(\bar{G}) \leq k$. Suppose that $H = (V_H, E_H)$ is a subgraph of G_1 isomorphic to G_2, and that $f : V_2 \to V_H$ is the isomorphism bijective function. Let $\bar{V}$ be a set of vertices such that, for each $v_1 \in V_H$ and $v_2 \in V_2$ with $f(v_2) = v_1$, there is a vertex $\bar{v} = (v'_1/v'_2) \in \bar{V}$. Also, for each $v_1 \in V_1 - V_H$, there is a vertex $\bar{v} = (v'_1/-) \in \bar{V}$. Similarly, let $\bar{E}$ be a set of arcs such that, for each $e_1 = (u_1, v_1) \in E_H$ and $e_2 = (u_2, v_2) \in E_2$ with $f(u_2) = u_1$ and $f(v_2) = v_1$, there is an arc $\bar{e} = (e'_1/e'_2) \in \bar{E}$. Also, for each $e_1 \in V_1 - V_H$, there is an arc $\bar{e} = (e'_1/-) \in \bar{E}$. Since H is isomorphic to G_2 and all operations in G'_1 and G'_2 are commutative and implemented by the same unique hardware block of the component library, $\bar{V}$ and $\bar{E}$ define the merged graph $\bar{G} = (\bar{V}, \bar{E})$. Furthermore, $A(\bar{G}) = k$ because $|\bar{V}| = |V_1|$, $|\bar{E}| = |E_1|$, $A_{hb}(\bar{v}) = 1, \forall \bar{v} \in \bar{V}$, and $A_{mux} = 1$.

Conversely, suppose that $\bar{G} = (\bar{V}, \bar{E})$ is a merged graph corresponding to G'_1 and G'_2 such that $A(\bar{G}) \leq k$. Since $k = A(G'_1)$, $\bar{G}$ has all vertices and arcs from G'_2 mapped on vertices and arcs from G'_1. Let V_H be the set of vertices $v_1 \in V_1$ with $\bar{v} = (v'_1/v'_2) \in \bar{V}$, and let $f : V_2 \to V_H$ be a function such that $f(v_2) = v_1$. Similarly, let E_H be the set of arcs $e_1 \in E_1$ with $\bar{e} = (e'_1/e'_2) \in \bar{E}$. V_H and E_H define the subgraph $H = (V_H, E_H)$ of G_1 isomorphic to G_2, and f is the isomorphism function.

This reduction can be performed in polynomial time since it requires only the construction of G'_1 and G'_2 from G_1 and G_2, and the computation of k, in order to transform an instance of ISO into an instance of MERGE. Also, the result of MERGE is trivially transformed into the result of ISO. $\square$

5 Conclusion

This report presented the control/data-flow graph merge problem. Performance speedup can be achieved through architectures that map the most time-consuming application kernel modules and/or inner-loops to a reconfigurable datapath. We represent each such modules/loops as CDFGs and merge them together into a single reconfigurable datapath, minimizing its area cost. Using a polynomial-time reduction from the subgraph isomorphism problem, we proved that CDFG-merge problem is NP-complete.

References

[1] K. Compton and S. Hauck. Reconfigurable computing: A survey of systems and software. *ACM Computing Surveys*, 34(2):171–210, June 2002.

[2] M.R. Garey and D. S. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness*. Freeman and CO., 1979.

[3] H. Schmit et al. PipeRench: A virtualized programmable datapath in 0.18 micron technology. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 63–66, 2002.

[4] H. Singh et al. MorphoSys: Case study of a reconfigurable computing system targeting multimedia applications. In *Proceedings of the Design Automation Conference*, pages 573–578, June 2000.

[5] W. Wolf. *Computers as Components – Principles of Embedded Computing System Design.* Morgan Kaufmann Publishers, 2001.