Modeling and Analysis of Distributed Reconfigurable Hardware*

Christian Haubelt and Jürgen Teich
Hardware-Software-Co-Design
University of Erlangen-Nuremberg
{haubelt, teich}@cs.fau.de

The ability to migrate hardware processes in a network of hardware reconfigurable nodes improves the fault tolerance of these networks. The degree of fault tolerance is inherent to such networked systems and can be optimized during design time. Therefore, an efficient way of calculating the degree of fault tolerance is needed. This paper presents an approach based on satisfiability testing which regards the question: How many resources may fail in a distributed reconfigurable system without losing any functionality?

1 Introduction

Networked and hardware reconfigurable platforms [4] are becoming more and more important in many areas, e.g., automotive control systems, body area networks, ambient intelligence, etc. An outstanding property of these systems is the ability of hardware process migration. In terms of system synthesis, this means that the binding of processes to resources is not static. In a network of connected FPGAs, many processes must be assigned to nodes in the network and it becomes possible to migrate hardware processes from one node to another during run-time. Thus, resource faults can be compensated by *rebinding* processes. The task of rebinding is also called *online partitioning*. A network of reconfigurable nodes that implements online partitioning will be termed *ReCoNet* in the following [7]. By using hardware reconfiguration, we have the opportunity to efficiently optimize several objectives, like power consumption, latency, etc., simultaneously and during run-time. This is not possible when using a software only approach. Figure 1(a) shows an implementation of a ReCoNet. Other first approaches for networked reconfigurable architectures are PACT [1] and Chameleon [3].

The ability to migrate processes from one reconfigurable node to another improves the fault tolerance of a ReCoNet, i.e., node failures can be compensated online. The degree of fault tolerance is inherent to the system and can be optimized during run-time. In order to evaluate the degree of fault tolerance, we define a new objective called *k-bindability*. A system is called *k-bindable* iff any set of k resources is redundant. Note, it may be possible that more than k resources are redundant but the k-bindability determines that k such that any set of k arbitrary resources can be removed from the system without losing any functionality.

This paper presents an approach based on satisfiability testing (SAT) which regards the question: How many resources may fail in a ReCoNet without losing any functionality? In Section 2

*Supported in part by the German Science Foundation (DFG), SPP 1148 (Rekonfigurierbare Rechensysteme)

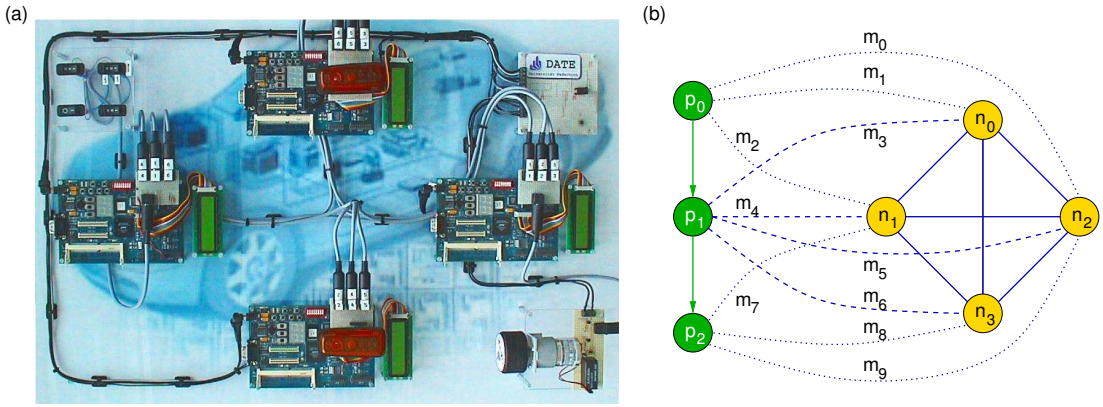

Figure 1: (a) Implementation of a ReCoNet consisting of four nodes. (b) Graph-theoretic model of the ReCoNet consisting of the four nodes $\mathsf{n}_0, \mathsf{n}_1, \mathsf{n}_2$, and n_3 and three processes $\mathsf{p}_0, \mathsf{p}_1$, and p_2. The additional mapping edges m_i model possible bindings of processes onto nodes.

we introduce the formal specification model of distributed reconfigurable systems used in this paper. The following section shows how to reduce the binding problem to the satisfiability problem of boolean formulas and a SAT-based approach to determine the k-bindability of a distributed reconfigurable systems that supports online partitioning is proposed. Finally, we will show by experiment (Section 4) that problem instances of reasonable size are easily solved.

2 Modeling Distributed Reconfigurable Systems

In order to allow a formal analysis of a ReCoNet, we need an appropriate mathematical model: The behavior of the system is modeled by a *process graph* $g_\mathrm{p} = (P, D)$ consisting of communicating processes $p \in P$. The data dependencies are modeled by edges $d \in D \subseteq P \times P$. The structure of the ReCoNet is given by a so-called *architecture graph* $g_\mathrm{a} = (N, C)$, where N is the finite set of nodes and $C \subseteq N \times N$ is the finite set of connections in the network. In order to relate the behavior and the structure, we use so-called *mapping edges* $m \in M \subseteq P \times N$ that relate processes $p \in P$ to nodes $n \in N$.

Figure 1(b) shows a model of a ReCoNet. The set of processes P and data dependencies D are given by $P = \{\mathrm{p}_0, \mathrm{p}_1, \mathrm{p}_2\}$ and $D = \{(\mathrm{p}_0, \mathrm{p}_1), (\mathrm{p}_1, \mathrm{p}_2)\}$, respectively. The architecture graph consists of the nodes $N = \{\mathrm{n}_0, \mathrm{n}_1, \mathrm{n}_2, \mathrm{n}_3\}$. The mapping edges $M = \{\mathrm{m}_0, \ldots, \mathrm{m}_9\}$ indicate that the sample process p_0 may be executed on any reconfigurable node n_0 to n_2 and the driver process p_2 may be performed on any of the reconfigurable nodes n_1 to n_3. The controller process p_1 could be bound to any of the reconfigurable nodes in the architecture graph.

3 Determining the Degree of Fault Tolerance

With the model introduced previously, the task of system synthesis is: “Find a feasible *binding* of the processes $p \in P$ to reconfigurable nodes $n \in N$, i.e., a subset of mapping edges.” Here, a binding is said to be feasible if:

1. each process $p \in P$ is bound to exactly one reconfigurable node $n \in N$ and

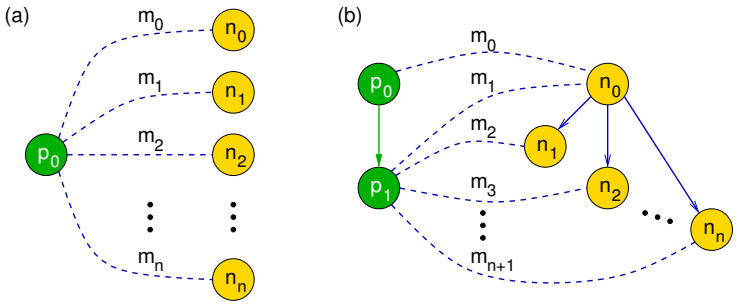

Figure 2: (a) For each process $\mathrm{p} \in \mathrm{P}$ exactly one outgoing mapping edge has to be activated. (b) In order to establish the required communication ($\mathrm{d} = (\mathrm{p}_0, \mathrm{p}_1)$), we have to execute the processes p_0 and p_1 on the same reconfigurable node n_0 or on adjacent reconfigurable nodes.

2. required communications given by the data dependencies $d \in D$ can be handled by the given architecture graph, i.e., if there is a directed edge $d = (p_i, p_j)$ then either p_i and p_j have to be performed on the same reconfigurable node n (intra-node communication) or on reconfigurable nodes n_i and n_j which are directly connected via an edge $c = (n_i, n_j)$ (inter-node communication).

Blickle et al. [2] have reduced the problem of finding a feasible binding to the boolean satisfiability problem which is NP-complete. In this paper, we show how to derive boolean functions from specifications given by a process graph, an architecture graph, and the mapping edges such that the boolean function is satisfiable iff the specified ReCoNet has a feasible binding. Later, we extend this idea in order to analyze aspects such as whether a ReCoNet is fault tolerant and to what degree.

First, we consider the problem of checking the feasibility of a given binding. Therefore, we introduce some notations: Let m_i be a boolean variable, indicating if the mapping edge m_i is part of the binding ($m_i = 1$), or not ($m_i = 0$). The assignment of all variables m_i is denoted by (m). The set of all possible assignments of (m) is denoted by (M). We check the feasibility of a given binding represented by the coding (m) by solving a boolean equation. Therefore, we test both criteria of the feasibility as given above. First, we test if there is exactly one outgoing mapping edge per process $p \in P$. As an example see Figure 2(a). For each process $p \in P$ we have to establish a formula which guarantees the required property. The logical product results in a boolean function $b_1 : (M) \rightarrow \{0, 1\}$ with $b_1((m)) = 1$ iff (m) contains exactly one mapping edge per process. Hence, we obtain Equation (1) where $\prod$ denotes the boolean AND and $\sum$ denotes the boolean OR.

$$b_1((m)) = \prod_{p \in P} \left[\left(\sum_{\substack{m \in M: \\ m=(p,n)}} m \right) \cdot \left(\prod_{\substack{m_i, m_j \in M: \\ m_i=(p,n_x) \wedge m_j=(p,n_y) \wedge n_x \neq n_y}} (\overline{m}_i + \overline{m}_j) \right) \right] \tag{1}$$

The first clause ensures that at least one of the mapping edges is activated ($m_i = 1$). The remainder guarantees that at most one mapping edge is part of the binding. The conjunction of both parts results in the required property.

Next, we check the second property of feasible bindings. All data dependencies $d \in D$ must be provided by the architecture of the implementation. Therefore, let $d = (p_i, p_j)$. If p_i is executed on n_x then p_j has to be performed on n_x as well or on an adjacent reconfigurable node n_j, i.e., $(n_i, n_j) \in C$. An example is shown in Figure 2(b). A boolean function $b_2 : (M) \rightarrow \{0, 1\}$ to

check the second property of feasibility is:

$$b_2\left((m)\right) = \prod_{\substack{m=(p,n)\in M,\\ p_i\in P:(p,p_i)\in D}} \left[\overline{m} + \sum_{\substack{m_j\in M:m_j=(p_i,n_x)\wedge\\ (n=n_x\vee(n,n_x)\in C)}} m_j\right] \tag{2}$$

With Equation (1) and (2), we formulate a boolean function $b : (M) \rightarrow \{0,1\}$ to check the feasibility of a given binding coded by (m):

$$b\left((m)\right) = b_1\left((m)\right) \cdot b_2\left((m)\right) \tag{3}$$

b is given in cnf and $b\left((m)\right) = 1$ iff the system has a feasible binding. In order to check if there is at least one feasible binding for a given specification, a SAT solver may be used to solve the following problem: $\exists(m) : b\left((m)\right)$. In a ReCoNet, the availability of resources may change over time, e.g., due to resource faults. It may be possible to compensate resource faults by *rebinding* processes to fully functional resources. The task of rebinding at run-time is called *online partitioning*. A first implementation of this task is described in [7]. The three basic operations needed to implement a ReCoNet are:

1. Line detection: Is a link $c = (n_1, n_2)$ between two nodes available or not?
2. Network state distribution: If a connection c' between two nodes fails, all available nodes $n \in N$ in the network not incident to c' must be informed.
3. Routing of failing communication links $d \in D$.

Nevertheless, in this paper, we focus on how to model resource faults and how to measure the robustness of a given ReCoNet. Therefore, we define the so-called *k-bindability* which quantifies the number of redundant resources in such a system.

If a reconfigurable nodes fails, i.e., we cannot use this node for further process execution, the *allocation* of resources nodes may change. The allocation is the set of used resources in our implementation. As in the case of the binding, we use the term (n) as the coding of an allocation where $n_i = 1$ indicates that the reconfigurable node n_i is part of the allocation. Furthermore, the term (N) describes the set of all possible allocation codings. If a reconfigurable node n_i fails, we have to set the associated binary variable n_i to zero. All incident mapping edges m_j to n_i become meaningless, and should be excluded from the binding, i.e., $m_j = 0$. Again, we propose a boolean function to deactivate all mapping edges adjacent to a defect reconfigurable node. This boolean function $e : (M) \times (N) \rightarrow \{0,1\}$ is satisfiable iff no reconfigurable nodes fail or there exists a feasible binding not using any of the mapping edges to the defect node.

$$e\left((m),(n)\right) = \prod_{n\in N, m\in M:m=(p,n)} (p + \overline{m}) \tag{4}$$

With this formula, we can check if a given reconfigurable node is redundant. For example, to test if n_0 is redundant we solve the following SAT formula: $\exists(m),(n) : \overline{n}_0 \cdot e\left((m),(n)\right) \cdot b\left((m)\right)$. A frequent question is how many resources could fail in a ReCoNet without losing the desired functionality. Therefore, we define the number of nodes that may fail as *k-bindability*, i.e., k is the maximum number such that any set of k reconfigurable nodes is redundant. Note that we can remove any $n < k$ nodes of our distributed system without losing the specified functionality.

In order to check for k-bindability using SAT-based techniques, we formulate a boolean function which encodes all system errors with exactly k reconfigurable node defects: $f^{(k)} : (A) \times (N) \rightarrow \{0,1\}$. This function depends on the auxiliary variables a_i with $i = 0, \ldots, a_{|N|-1}$ where $|N|$ denotes the cardinality of N. If exactly k auxiliary variables are set to zero, we set the k

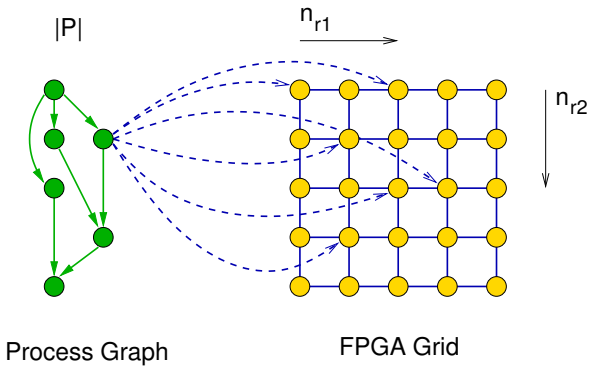

Figure 3: Example specification of an FPGA grid and an application consisting of $|P| = 6$ processes. The grid is composed of $n_{\mathrm{r1}} \times n_{\mathrm{r2}} = 5 \times 5$ FPGAs. The number of mapping edges per process is given by $n_{\mathrm{m}} = 6$.

corresponding allocation variables n_i to zero, otherwise all allocation variables may be set to one (for details see [5]). Now, that we know how to code resource defects in a boolean function, we formulate the general form of the QBF solving the k-bindability problem:

$$\forall(a)\exists(n),(m) : f^{(k)}\left((a),(n)\right) \cdot e\left((m),(n)\right) \cdot b\left((m)\right) \tag{5}$$

4 Results

The k-bindability as defined above specifies the degree of fault tolerance of a ReCoNet. This fault tolerance can be optimized during system design. In order to compare different implementations during design space exploration, we must evaluate these implementations. In this section, we present first results of our new approach by using QSOLVE [6]. For this purpose, we design a benchmark. Our goal is to evaluate the run-time in dependence of the problem size which could be easily solved and, hence, could be investigated during automated design space exploration.

In a first step, an array of $n_{\mathrm{r1}} \times n_{\mathrm{r2}}$ reconfigurable nodes is defined. Communications are established such that the array is a 2-dimensional grid. Here, we use a grid in order to construct scalable architectures of distributed reconfigurable systems. Furthermore, a grid is typical for reconfigurable architectures as FPGAs and coarse grain architectures like PACT [1], Chameleon [3], etc. Next, we define a weakly connected random process graph with $|P| = n_{\mathrm{p}}$ processes. The probability that there is a data dependency between process p_i and process p_j with $j > i$ is given by another parameter called pb. In a last step, n_{m} mapping edges are randomly drawn from each process $p \in P$ to reconfigurable nodes $n \in N$, i.e., there are $|M| = |P| \cdot n_{\mathrm{m}}$ mapping edges. Figure 3 shows an example of such a specification. The most meaningful results are presented in the following.

Table 1 shows the average results (100 samples each) obtained from solving the boolean functions with the QBF-solver QSOLVE. We have chosen a 4×4 grid of reconfigurable nodes. Different numbers n_{p} of processes are mapped onto this architecture. With parameters $n_{\mathrm{m}} = 13$ and $pb = 0.5$, we check the k-bindability for $k = 4, \ldots, 1$. Table 1 shows that systems with 50 processes are still solvable in a reasonable amount of time. As mentioned above, our approach is not limited to grids of reconfigurable nodes but supports arbitrary topologies. Thus, it is possible to optimize the architecture of a ReCoNet by using SAT-based techniques during design space exploration.

Table 1: Number of recursions recur, number of assignments assign, and computation time time to solve the k-bindability equation (satisfiable) Equation (5).

$\|P\|$		$k=1$	$k=2$	$k=3$	$k=4$	$\|P\|$		$k=1$	$k=2$	$k=3$	$k=4$
	$recur$	532	2759	10230	30106		$recur$	587	4972	19136	49858
25	$assign$	8569	43856	163972	487057	40	$assign$	10179	77695	293135	827385
	$time/s$	0.10	0.60	2.22	6.72		$time/s$	0.23	1.78	7.06	20.26
	$recur$	434	3235	11931	35222		$recur$	649	4524	17107	51307
30	$assign$	7474	54488	201263	594378	45	$assign$	12109	86569	318774	933571
	$time/s$	0.12	0.94	3.62	10.69		$time/s$	0.35	2.47	9.15	26.28
	$recur$	488	3593	13403	39787		$recur$	633	4893	17818	53272
35	$assign$	9031	65010	240039	705878	50	$assign$	12573	93306	342441	1007054
	$time/s$	0.19	1.45	5.89	16.63		$time/s$	0.37	2.74	10.02	29.90

5 Conclusions

Distributed reconfigurable systems possess an inherent fault tolerance which can be optimized during system design. In order to determine the degree of fault tolerance a system has, we have to answer the question: How many resources may fail in a distributed reconfigurable system that supports online partitioning without losing any functionality? In this paper, we proposed a fast method to answer this question based on quantified boolean formulas and applying the QBF solver QSOLVE in order to test the satisfiability of these formulas. Only if these formulas are satisfiable, the modeled system has the tested degree of fault tolerance. We have shown by experiment that our new approach can easily be applied to systems of reasonable size as we will find in the future in the field of body area networks and ambient intelligence.

References

[1] V. Baumgarte, F. May, A. Nückel, M. Vorbach, and M. Weinhardt. PACT XPP - A Self-Reconfigurable Data Processing Architecture. In *ERSA*, Las Vegas, Nevada, June 2001.

[2] T. Blickle, J. Teich, and L. Thiele. System-Level Synthesis Using Evolutionary Algorithms. In Rajesh Gupta, editor, *Design Automation for Embedded Systems*, 3, pages 23–62. Kluwer Academic Publishers, Boston, January 1998.

[3] Chameleon Systems. *CS2000 Reconfigurable Communications Processor*, 2000.

[4] R. Dick and N. Jha. CORDS: Hardware-Software Co-Synthesis of Reconfigurable Real-Time Distributed Embedded Systems. In *Proc. of ICCAD'98*, pages 62–68, 1998.

[5] R. Feldmann, C. Haubelt, B. Monien, and J. Teich. Fault Tolerance Analysis of Distributed Reconfigurable Systems Using SAT-Based Techniques. In *Proc. of Int. Conf. on FPL2003*, pages 478–487, Lisbon, Portugal, September 2003.

[6] R. Feldmann, B. Monien, and S. Schamberger. A Distributed Algorithm to Evaluate Quantified Boolean Formulas. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence*, pages 285–290, 2000.

[7] C. Haubelt, D. Koch, and J. Teich. ReCoNet: Modeling and Implementation of Fault Tolerant Distributed Reconfigurable Hardware. In *Proc. of SBCCI2003*, pages 343–348, São Paulo, Brazil, September 2003.