# Reconfigurable Interconnect Synthesis via Quantified Boolean Satisfiability

Subramanyan Siva[1] Ranga Vemuri[1] Willard Vanfleet[2] John Franco[1]
John Schlipf[1]
[1] Department of ECECS, University of Cincinnati, Cincinnati, OH 45221-0030
[2] Department of Defense

**Abstract**

Many state-of-the-art reconfigurable computers consist of multiple field-programmable processors connected through a programmable interconnect fabric. Interconnect synthesis is the process of configuring the interconnection network to match the communication requirements of the designs implemented on the processors. We formulate interconnect synthesis as the problem of determining satisfiability of quantified Boolean formulae. For this class of quantified Boolean formulae, we propose an efficient method to eliminate the quantifiers so that satisfiability of the resulting quantifier-free formulae can be determined using ordinary satisfiability solvers.

## 1 Introduction

Reconfigurable Computers (RC) contain field-programmable processors and interconnect elements whose logic and communication structures can adapted to best suit the computation and communication requirements of an application [2]. Many commercial RCs today use Field-Programmable Gate Arrays (FPGA) as the processing elements and programmable interconnection networks based on simple configurable routing elements [3]. Figure 1 shows the general architecture of a reconfigurable computer.

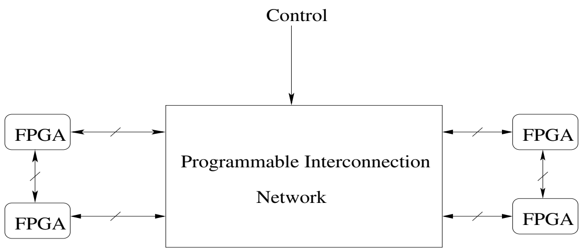


Figure 1: *Example of a Reconfigurable Computer*

Figure 2 shows the general architecture of the programmable interconnection network. The network routes signals among the various input and output pins of the FPGAs. The specific routing is determined by the control signals to each of the routing blocks. Typically, but not necessarily, the control signals define a permutation of the inputs of the block and the permuted signals are routed to the corresponding output pins of the block. Each control signal can be set to on, off or don't care. A control signal setting is said to be a *program* of the interconnection network. A program defines a routing of the signals through the interconnection network. A required routing may be realizable through one or more programs or not realizable at all depending upon the routing capabilities of the interconnection blocks and how they are connected. A *configuration* of an interconnection network refers to a set of routes realized by a program. Where as a program defines a configuration, it is not necessary that each configuration is realizable by a unique program. Computer-

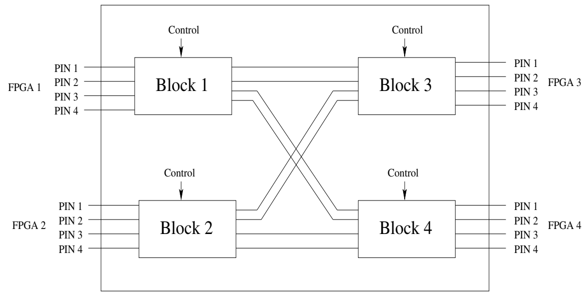


Figure 2: *A Programmable Interconnection Network*

Aided Design (CAD) tools for RCs must perform three interrelated tasks [12]: (1) Partition the computation into several fragments and map them to the various FPGAs; (2) Determine the control signal settings for the interconnection blocks so the signals among the FPGAs are appropriately routed; and, (3) Assign the interface signals of design fragments inside each FPGA to its pins so that the pin assignment is compatible with the configuration of the interconnection network. These are refereed to as the problems of partitioning, interconnect synthesis and pin-assignment respectively. This paper addresses the latter two problems.

## 2 Related Research

Devadas [1] proposed the idea of using Boolean satisfiability methods to solve layout synthesis problems including routing in VLSI chips. Wood and Rutenbar [6] proposed a Boolean satisfiability based approach to determine routability in a certain class of FPGAs referred to as island-style FPGAs. The problem is solved by representing the Boolean functions as Binary Decision Diagrams (BDD) [7].

Srinivasan et al. [4, 5] addressed the problem of interconnect synthesis for reconfigurable architectures. They modeled the interconnect synthesis problem as a Boolean satisfiability problem and proposed a solution using the ARC (Analyzer for Reconfigurable Computers) to specify RC interconnection architectures and to extract the Boolean expressions representing the behavior of the interconnection network. They have used BDD based techniques to solve the resulting satisfiability problem. We follow essentially the same general approach but instead of BDD techniques we use SAT techniques.

## 3 Proposed Approach

We specify a class of interconnection architectures using a modeling called PDL+ [10, 13]. ARC is an environment to elaborate and evaluate PDL+ models [11]. Among various tools provided in ARC is a symbolic evaluator which can be used to symbolically evaluate elaborated PDL+ models. This evaluator is used to generate Boolean expressions representing the routing behavior of a specific member of the family of interconnection networks modeled in PDL+. The process is described in detail in [5, 12]. Figure 3 shows the integration of the various components that constitute the interconnect synthesis system. We illustrate this process by an example:

Figure 4 shows a simple switch-box component. When the control input s is false, *S* transmits a to x and b to y. When

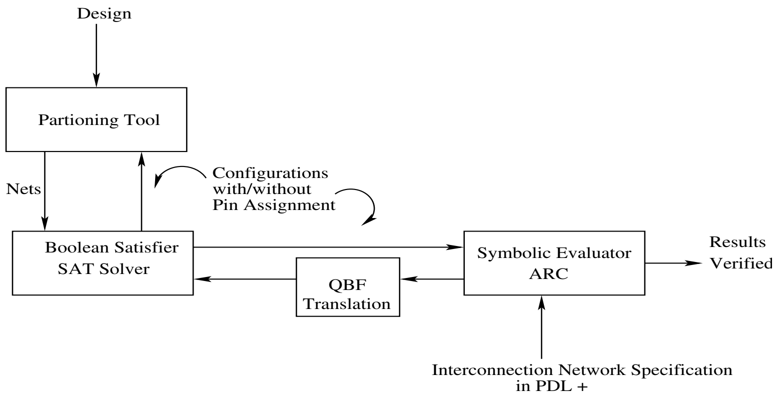


Figure 3: *Interconnect Synthesis Environment*

the control input s is true, *S* transmits a to y and b to x. Figure 5 shows the interconnection network that instantiates three switch boxes.

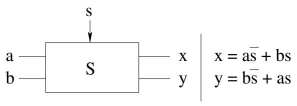


Figure 4: *A Simple Switch-Box Component*

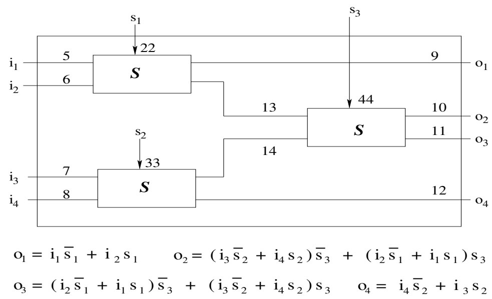


Figure 5: *A Network of Switch-boxes*

The switch-box and the network are modeled in PDL+. Listing 1 shows the result of symbolic evaluation [10] for the interconnect network shown in Figure 5.

The interconnect synthesis problem is represented as Quantified Boolean satisfiability (QBF-SAT) problem shown in Listing 2. Listing 3 shows the QBF-SAT formulation for some desired interconnections through the network shown in Figure 5.

Some QBF-SAT problems can be reduced into quantifier-free SAT problems In our approach, the quantified formulae representing the interconnect synthesis problem are translated into quantifier-free formulae using an approach named the *impulse response* method discussed below.

Given a Boolean vector $\{x_1, x_2, \cdots x_n\}$ an *impulse* is defined to be an assignment of *true* to one of the variables $x_i$ and *false* to all the other variables. Clearly, there are $n$ possible impulses for a vector of $n$ variables. Each interconnect building block simply routes input signals to outputs such that in any configuration, each output is directly connected to exactly one input.

| Result of Symbolic Evaluator | 1 |
|---|---|

```
board.o1'val = S1.x'val = ((board.i1'val and not(board.s1'val)) or (board.i2'val and board.s1'val));
board.o2'val = S3.x'val = ((S2.x'val and not(board.s3'val)) or (S1.y'val and board.s3'val));
board.o3'val = S3.y'val = ((S1.y'val and not(board.s3'val)) or (S2.x'val and board.s3'val));
board.o4'val = S2.y'val = ((board.i4'val and not(board.s2'val)) or (board.i3'val and board.s2'val));
S1.y'val     = S3.b'val = ((board.i2'val and not(board.s1'val)) or (board.i1'val and board.s1'val));
S2.x'val     = S3.a'val = ((board.i3'val and not(board.s2'val)) or (board.i4'val and board.s2'val));
```

| QBF Formulation | 2 |
|---|---|

```
∀(Primary Inputs) ∃(Primary Outputs and Intermediate Outputs)
   such that Model{(Interconnect Specification) and (Desired Interconnect Connection)}
   is satisfied.
```

| QBF for Interconnect Layout (Fig.5) | 3 |
|---|---|

```
( all-quantify
 ( exist-quantify
  ( and
   ( equ 9   ( ite 22 6 5))    --|
   ( equ 10  ( ite 44 13 14)) --|
   ( equ 11  ( ite 44 14 13)) --|Interconnect
   ( equ 12  ( ite 33 7 8))    --|Specifications
   ( equ 13  ( ite 22 5 6))    --|
   ( equ 14  ( ite 33 8 7))    --|
   ( equ 5 9)    -- |
   ( equ 6 11)   -- |Desired Interconnect
   ( equ 7 10)   -- |Connections
   ( equ 8 12)) -- |
 ( and 9 10 11 12 13 14)) -- Quantify away Internal Nets
( and 5 6 7 8))   -- Universal Quantify away Inputs
```

| QBF Translation | 4 |
|---|---|

```
For each input in the desired connection
{
   Set that input to false
   Set other inputs to true
   Create unique variables for intermediate outputs and primary outputs
}
```

Listing 4 shows the translation process and Listing 5 gives the propositional formulae after QBF translation. This method increases the number of constraints considerably but makes the problem simple as values can be inferenced immediately.

| QBF to SAT Translation for Interconnect Layout (Fig.5): Impulse Response Logic | 5 |
| --- | --- |

```
( and
    ( and
            ( equ 90   ( ite 22 1 0))
            ( equ 100  ( ite 44 130 140))
            ( equ 110  ( ite 44 140 130))
            ( equ 120  ( ite 33 1 1))
            ( equ 130  ( ite 22 0 1))
            ( equ 140  ( ite 33 1 1))
            ( equ 0 90)    -- |
            ( equ 1 110)   -- |Impulse applied on I1(5)
            ( equ 1 100)   -- |
            ( equ 1 120)   -- |
    )
                        :
                        :
```

# 4 Switch Box Modeling

**Definition:** Let the input set *i* be defined as a set of inputs $i_1, i_2 \ldots i_n$ and the output set *o* be defined as a set of outputs $o_1, o_2 \ldots o_m$. A route *r* establishes a connection between the elements of two sets. The output set is partitioned into *m* subsets $p_1, p_2 \ldots p_m$, such that an element in set *o* lies in no two partitions. A subset contains all outputs originating from the same input as shown in Figure 6. If $o_1 \in p_1$, then by definition $o_1 \notin p_2$ as a partition is formed whose elements (outputs) in the partition are connected to the same input.

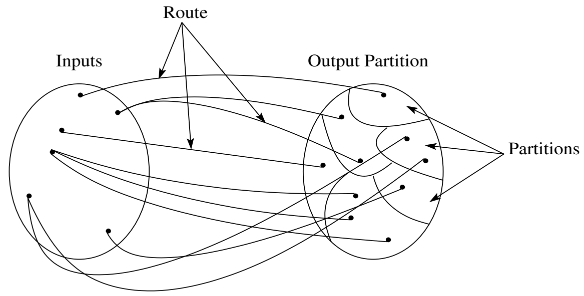


Figure 6: *Routes through Switch Boxes*

Hence $p_1 \cap p_2 \cap \ldots \cap p_m = \emptyset$ and $p_1 \cup p_2 \cup \ldots \cup p_m = o$

**Lemma:** If a set $S_1$ is partitioned into $S_1 = \{S_{11}, S_{12} \ldots S_{1n}\}$, and a set $S_2$ is partitioned into $S_2 = \{S_{21}, S_{22} \ldots S_{2n}\}$ and if $S_{11} = S_{21}$, $S_{12} = S_{22}$, $\ldots$, $S_{1n} = S_{2n}$ then set $S_1 = S_2$.

**Proof:** $S_{11}$ implies that there exists some element(s) in subset $p_1$ of set $S_1$. If $S_{11} = S_{21}$ then there are exactly same number of elements in subsets $S_{11}$ and $S_{21}$. Also, the outputs for both subsets originate from the same input. We know from definition each element belongs to one subset only. Hence if $S_{11} = S_{21}$, $S_{12} = S_{22}$, $\ldots$, $S_{1n} = S_{2n}$, elements in set $S_1$ are same as the elements in set $S_2$ and their outputs originate from the same input. Hence $S_1 = S_2$.

Given a switch box B with inputs I = $\{i_1, i_2 \ldots i_n\}$ and outputs O = $\{o_1, o_2 \ldots o_m\}$, a route r connects the outputs to the inputs. We shall denote route as a funcion r : O $\rightarrow$ I. Let R denote a set of all possible routes. Let C = $\{c_1, c_2 \ldots c_n\}$ be the set of configurations of the switch box.

We shall view the switch box as a function B : C $\rightarrow$ R. Let the desired connections net list be represented by the function as d : O $\rightarrow$ I

B projection r denoted as $B\mid_r$ is a set of configurations such that for the particular set of configuration, the route exists.

$B\mid_r$ = {c $\in$ C | B(c) = r}

r projection i (impulse) denoted as $r\mid_i$, is a set of outputs o such that r(o) = i

$r\mid_i$ = {o $\in$ O | r(o) = i}

B projection d on i denoted as $B\mid_{d|_i}$, is a set of configuration stream c such that B(c) = r and $r_i = d_i$

$$B\mid_{d|_i} = \{c \in C \mid B(c) = r \ and \ r_i = d_i\} \tag{1}$$

**Theorem:** Given a box B and the desired connections d,

$$B\mid_d = \bigcap_{i \in I} B\mid_{d|_i}$$

**Proof:**

$$\bigcap_{i \in I} B\mid_{d|_i} = B\mid_{d|_{i1}} \bigcap B\mid_{d|_{i2}} \ldots B\mid_{d|_{in}}$$

$$= \{c\mid_{B(c)|_{i_1} = d|_{i_1}}\} \bigcap \{c\mid_{B(c)|_{i_2} = d|_{i_2}}\} \cdots \bigcap \{c\mid_{B(c)|_{i_n} = d|_{i_n}}\} \qquad -- \ From \ \ Equation \ 1$$

$$= \{c\mid_{B(c)|_{i_1} = d|_{i_1} \ and \ B(c)|_{i_1} = d|_{i_1} \ldots \ and \ B(c)|_{i_n} = d|_{i_n}}\}$$

$$= \{c\mid_{B(c) = d}\} \qquad -- \ From \ Lemma$$

Therefore for a switch box, given the set of desired connections netlist, the *impulse response* logic guarantess routing if one exists.

# 5 Pin Assignment

In order to provide a mechanism to assign the logic signals to the appropriate FPGA pins, Srinivasan et al. [4] proposed a virtual interconnection network method. Figure 7 shows the virtual interconnection network.

# 6 Experimental Results

Interconnect synthesis is performed for varying number of desired interconnections for Wildforce [8] and CoReL [9] RC boards. Column 2 furnishes the time taken to convert the interconnect QBF to propositional formulae denoted as $T_{QBF \rightarrow prop}$. As the number of nets increases, the time taken to perform the translation increases for both boards. Column 3, 4 and 5 shows the time taken for generating the appropriate interconnection configuration stream that needs

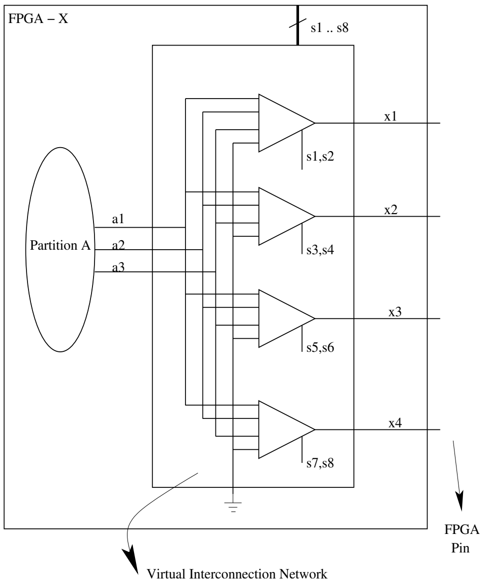


Figure 7: *Virtual Interconnection Network to Model Pin Assignment*

to be programmed to realize the connectivity using Grasp [15], Sato [16] and Chaff [14] solvers respectively. All experiments were conducted on Sun Ultra Sparc 2 workstation at 296 MHz with 384 MB of RAM.

| No. of Nets | $T_{QBF \rightarrow prop}$ (mts) | Grasp (secs) | Sato (secs) | Chaff (secs) |
|---|---|---|---|---|
| 4 | 0.444 | 50.67 | 2.62 | 0.93 |
| 8 | 0.564 | 103.22 | 5.81 | 2.62 |
| 12 | 0.681 | 150.84 | 10.52 | 5.23 |
| 16 | 0.798 | 226.2 | 17.28 | 5.12 |
| 20 | 0.922 | 309.49* | 25.10 | 8.88 |
| 24 | 1.039 | 372.21* | 34.15 | 10.51 |
| 28 | 1.164 | 440.57* | 41.43 | 13.1 |
| 32 | 1.285 | 504.16* | 52.37 | 17.19 |
| 36 | 1.409 | 599.85* | 63.78 | 19.30 |

* Experimentation on Sun Sparc 2 at 336 Mhz with 1344 MB of RAM

Table 1: *Results of Wildforce*

## References


[1] S. DEVADAS. **Optimal Layout via Boolean Satisfiability**. In *Proc. of ACM/IEEE ICCAD*, pages 294-297, 1989.

[2] B. L. HUTCHINGS AND M. J. WIRTHLIN. **Implementation Approaches for Reconfigurable Logic Applications**. In *International Worlshop on Field-Programmable Logic and Applications*, FPL, pages 419-428, Springer, 1995.

[3] PAK K. CHAN. **Digital Design Using Field Programmable Gate Arrays**, Prentice Hall, 1994.

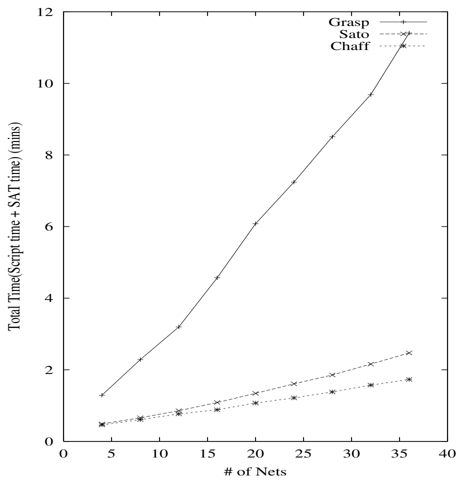


Figure 8: *Wildforce Plot*

| No. of Nets | $T_{QBF \rightarrow prop}$ (mts) | Grasp (secs) | Sato (secs) | Chaff (secs) |
|---|---|---|---|---|
| 8 | 0.563 | 6.48 | 1.06 | 0.18 |
| 16 | 0.912 | 12.24 | 133.20 | 0.38 |
| 24 | 1.26 | 19.17 | 5.93 | 0.62 |
| 32 | 1.679 | 26.77 | 10.92 | 0.82 |
| 40 | 1.965 | 34.43 | 15.25 | 1.04 |
| 48 | 2.311 | 42.46 | 20.40 | 1.23 |
| 56 | 2.618 | 49.67 | 27.09 | 1.47 |
| 64 | 3.024 | 60.47 | 36.34 | 1.79 |

Table 2: *Results of CoReL*


[4] VINOO SRINIVASAN, SHANKAR RADHAKRISHNAN, RANGA VEMURI AND JEFF WALRATH. **Interconnect Synthesis for Reconfigurable Multi-FPGA Architectures**. In *6th Reconfigurable Architectures Workshop*, pages 588-596, Springer-Verlag, April 1999.

[5] SHANKAR RADHAKRISHNAN. **Interconnect Synthesis for Reconfigurable Computers**. MS Thesis, University of Cincinnati, April 1999.

[6] R. G. WOOD AND R. RUTENBAR. **FPGA Routing and Routability Estimation via Boolean Satisfiability**. In *IEEE Trans. on VLSI*, vol. 6, No. 2, pages 222-2313, 1998.

[7] R. E. BRYANT. **Graph Based Algorithms for Boolean Function Manipulation**. In *IEEE Trans. on Computers*, C-35(8), pages 677-691, August 1986.

[8] WILDFORCE. **WILDFORCE Reference Manual, Document #11849-0000**, Annapolis Micro Systems, Inc.

[9] http://www-core.kaist.ac.kr

[10] RANGA VEMURI AND JEFF WALRATH. **Abstract Models of Reconfigurable Computer Architectures**. In *Configurable Computing: Technology and Applications, SPIE Symposium on Voice, Video and Data Communications, International Society for Optical Engineering*, Nov. 1998.

[11] JEFFREY WALRATH AND RANGA VEMURI. **A Performance Modeling and Analysis Environment for Reconfigurable Computers**. In *Proc. of Reconfigurable Architectures Workshop*, pages 19-24, Springer, March 1998.

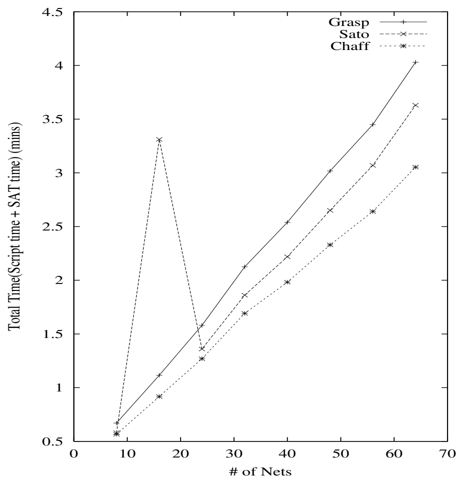


Figure 9: *Corel Plot*

[12] IYAD OUAISS, SRIRAM GOVINDARAJAN, VINOO SRINIVASAN, MEENAKSHI KAUL, AND RANGA VEMURI. **An Integrated Partitioning and Synthesis System for Dynamically Reconfigurable Multi-FPGA Architectures**. In *Proc. of Reconfigurable Architectures Workshop*, pages 31-36, Springer, March 1998.

[13] RANGA VEMURI, RAM MANDYAM AND VIJAY MEDURI. **Performance Modeling using PDL**. In *IEEE Trans. on Computers*, pages 44-53, 1996.

[14] M. MOSKEWICZ, C. MADIGAN, Y. ZHAO, L. ZHANG AND S. MALIK. **Chaff: Engineering an Efficient SAT solver**. In *38th ACM/IEEE Design Automation Conference*, pages 530-535, 2001.

[15] J. MARQUES-SILVA AND K. SAKALLAH. **Grasp: A Search Algorithm for Propositional Satisfiability**. *IEEE Transactions on Computers*, 48(5):506-521, May 1999.

[16] H. ZHANG. **Sato: An Efficient Propositional Prover**. In *International Conference on Automated Deduction*, pages 272-275, 1997.