Scan Pattern Retargeting and Merging with Reduced Access Time

Baranowski, Rafal; Kochte, Michael A.; Wunderlich, Hans-Joachim

Proceedings of the IEEE European Test Symposium (ETS'13) Avignon, France, 27-30 May 2013

doi: http://dx.doi.org/10.1109/ETS.2013.6569354

Abstract: Efficient access to on-chip instrumentation is a key enabler for post-silicon validation, debug, bringup or diagnosis. Re- configurable scan networks, as proposed by e.g. the IEEE Std. P1687, emerge as an effective and affordable means to cope with the increasing complexity of on-chip infrastructure. To access an element in a reconfigurable scan network, a scan- in bit sequence must be generated according to the current state and structure of the network. Due to sequential and combinational dependencies, the scan pattern generation process (pattern retargeting) poses a complex decision and optimization problem. This work presents a method for scan pattern generation with reduced access time. We map the access time reduction to a pseudo- Boolean optimization problem, which enables the use of efficient solvers to exhaustively explore the search space of valid scan-in sequences. This is the first automated method for efficient pattern retargeting in complex reconfigurable scan architectures such as P1687-based networks. It supports the concurrent access to multiple target scan registers (access merging) and generates reduced (short) scan-in sequences, considering all sequential and combinational dependencies. The proposed method achieves an access time reduction by up to 88x or 2.4x in average w.r.t. unoptimized satisfying solutions.

[1] **IEEE COPYRIGHT NOTICE**
©2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Scan Pattern Retargeting and Merging with Reduced Access Time

Rafal Baranowski, Michael A. Kochte, Hans-Joachim Wunderlich

ITI, University of Stuttgart, Pfaffenwaldring 47, D-70569, Stuttgart, Germany
Email: {baranowski, kochte}@iti.uni-stuttgart.de, wu@informatik.uni-stuttgart.de

***Abstract*—Efficient access to on-chip instrumentation is a key enabler for post-silicon validation, debug, bringup or diagnosis. Reconfigurable scan networks, as proposed by e.g. the IEEE Std. P1687, emerge as an effective and affordable means to cope with the increasing complexity of on-chip infrastructure.**

To access an element in a reconfigurable scan network, a scan-in bit sequence must be generated according to the current state and structure of the network. Due to sequential and combinational dependencies, the scan pattern generation process (*pattern retargeting*) poses a complex decision and optimization problem.

This work presents a method for scan pattern generation with reduced access time. We map the access time reduction to a pseudo-Boolean optimization problem, which enables the use of efficient solvers to exhaustively explore the search space of valid scan-in sequences. This is the first automated method for efficient pattern retargeting in complex reconfigurable scan architectures such as P1687-based networks. It supports the concurrent access to multiple target scan registers (*access merging*) and generates reduced (short) scan-in sequences, considering all sequential and combinational dependencies. The proposed method achieves an access time reduction by up to 88x or 2.4x in average w.r.t. unoptimized satisfying solutions.

***Keywords*-Design for debug & diagnosis, optimal pattern retargeting, scan pattern generation, reconfigurable scan network, IJTAG, P1687**

I. Introduction

As the complexity of designs increases, more and more instruments are integrated into on-chip infrastructure. Examples include structures for post-silicon validation, debug, bringup, or diagnosis. Embedded instrumentation is also used during operation in the field for power-up initialization, monitoring, error management, fault tolerance or repair [1], [2], [3]. Such tasks require flexible, fast and cost-effective access to the on-chip instrumentation. Reconfigurable Scan Networks (RSNs) emerge as a viable and scalable option, offering highly flexible infrastructures with distributed configuration in various topologies.

An example of a simple RSN is given in Fig. 1. The one-bit scan registers S1 and S3 control the access to two multi-bit registers S2 and S4, respectively. The scan-in data is shifted through registers S2 and S4 only if the previous access assured that S1 = S3 = 1.

The ongoing effort IEEE Std. P1687, also known as IJTAG, aims to standardize the design and access to RSNs, extending the widely adopted IEEE Std. 1149.1 (JTAG) [1]. Recently, the optimal construction of scan hierarchies [4] and access scheduling [5] was proposed for regular scan architectures compliant with P1687. These and similar contributions are based on scan networks with reconfigurability limited to scan bypasses (called Segment Insertion Bit, or SIB). In this particular subclass of RSNs, scan pattern generation is a simple decision problem. To access a scan register, the required scan-in sequences (scan patterns) are inferred directly from the structure of the network.

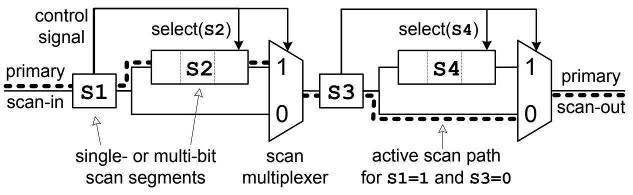

Figure 1. Example of a reconfigurable scan network and its terminology

In contrast, for general RSNs, scan pattern generation–called pattern retargeting in P1687–poses a much more difficult problem. Control signals for scan registers may depend on other scan registers in the same or different hierarchy levels. Not all scan registers may be accessible at a time and multiple access operations are typically required to reconfigure the network and access the target register. Due to sequential dependencies, scan pattern generation is an NP-hard decision problem which is similar to sequential stuck-at fault Automatic Test Pattern Generation (ATPG) [6].

Due to reconfigurability, an access to a scan register can be realized by many scan-in sequences. Possible solutions greatly differ in the number of bits that need to be shifted in. Table I shows an example of scan pattern generation for the RSN from Fig. 1. In the initial state, register S4 is inaccessible since $S3 = 0$. To access the target S4, register S3 is set to 1 in operation no. 1, and S4 is accessed in operation no. 2. The total access time of operations 1 and 2, which is the number of bits that are shifted in, amounts to $4 + 2 \cdot |\,S2\,| + |\,S4\,|$. This is a valid solution which could be obtained with the approach in [6], but the access time and the corresponding scan data volume is *not optimal*. It can be reduced if S2 is bypassed: Operations 1* and 2* perform an optimized access with minimal access time. If such minimization of total access time for multiple accesses (access merging) is targeted, the size of the search space increases drastically.

Table I: Pattern generation example for the RSN from Fig. 1

	S1	S2	S3	S4	access time	
initial state	1	X	0	X		
operation 1	W1	ACCESS	W1	BYPASS	2+|S2|	unopt.
operation 2	W1	ACCESS	W0	ACCESS	2+|S2|+|S4|	
initial state	1	X	0	X		
operation 1*	W0	ACCESS	W1	BYPASS	2+|S2|	opt.
operation 2*	W1	BYPASS	W0	ACCESS	2+|S4|	

This paper presents for the first time a method for access time reduction in reconfigurable scan networks. Our method enables efficient access to embedded instruments, e.g. for post-silicon validation, debug, bringup, diagnosis, or monitoring. Note that this method does not replace scan test pattern generation or test scheduling.

We formalize RSNs and access mechanisms with a compact model which is transformed into a set of clauses and a cost function. We propose an affordable, parallelized pattern generation procedure based on pseudo-Boolean optimization. For a given bound on the number of access operations, the proposed method guarantees that the generated patterns have the *minimal* length. Our experimental results show that such an optimization method is crucial for complex RSNs, reducing the access time by a factor of up to 88x. Consequently, our method is robust and more general than algorithms that are developed for only specific reconfigurable architectures, such as [7] for scan bypasses.

The following section introduces the terminology used across the paper. Section III describes the modeling of the network. Section IV presents the scan pattern generation procedure, followed by experimental results in Section V.

II. Terminology

Reconfigurable scan networks are usually accessed through a JTAG-compliant Test Access Port (TAP). An RSN can be viewed as a reconfigurable Test Data Register (TDR in IEEE Std. 1149.1/JTAG) with *variable length.* The logic state of the RSN determines which scan registers in the network are currently accessible. The RSN state may be changed by rewriting the content of accessible registers.

RSNs can be decomposed into basic building components, such as scan registers, multiplexers, or combinational logic blocks. In this paper, we follow a general definition of RSNs, as described below. Fig. 1 shows the terminology at an exemplary RSN.

The basic building block of an RSN is a *scan segment* (Fig. 2a). In the simplest case, a scan segment is a shift register composed of one or more *scan flip flops* sharing the *select* control input. When the *select* signal of a segment is active (segment *is selected*) during a *capture* operation, the shift register is loaded with data from outside of the RSN, e.g. with output of an on-chip instrument or combinational logic. If the segment is selected during a *shift* operation, data is shifted from the segment's scan-input, through its register bits, to the scan-output of the segment.

Optionally, a scan segment may include a *shadow* latch which is stable during the shift operation (as in JTAG test data registers). The optional elements of a scan segment are dashed in Fig. 2a. When the scan segment is selected during an *update* operation, the shadow latch is loaded from the shift register. A scan segment with a shadow latch may be used for bidirectional communication with an on-chip instrument. Scan segments with shadow latches are also used to drive internal control signals, such as *select* inputs of other scan segments (e.g. S1 and S3 in Fig. 1).

An RSN may include *scan multiplexers*, i.e. multiplexers which control the path through which the data is shifted in the RSN. For instance, the two scan multiplexers in Fig. 1 allow to bypass scan segments S2 and S4. The control signal of a scan multiplexer is called *address* and specifies the selected scan input.

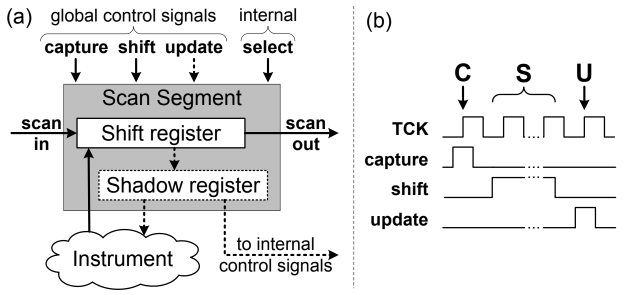

Figure 2. (a) Scan segment; (b) Capture, Shift, Update (CSU) operation

The state of both the *select* control inputs of scan segments and *address* inputs of scan multiplexers depends on the logic state of the RSN itself: These internal control inputs may be driven by arbitrary combinational logic blocks that take their input from shadow latches of scan segments (cf. Fig. 2a). For instance, all control signals in Fig. 1 are driven directly by the shadow latch of either scan segment S1 or S3.

A reconfigurable scan network has a *primary scan-input* and a *primary scan-output*, as well as global control inputs that enable the capture, shift, and update operations and are distributed to all scan segments. Optionally, an RSN may have external control inputs that drive internal control inputs (either directly or through combinational logic).

Two scan segments are *directly connected* if their scan-out and scan-in ports are connected either by a net or through a multiplexer. A *scan path* is a non-circular sequence of directly connected scan segments starting at a primary scan-in port and ending at a primary scan-out port. A scan path is *active* if and only if the select signals for all on-path scan segments are asserted and all on-paths multiplexers address the input that belongs to the active scan path. In Fig. 1, if $\mathtt{S1} = 1$ and $\mathtt{S3} = 0$, the active scan path goes through $\mathtt{S1}, \mathtt{S2}, \mathtt{S3}$ (S4 is bypassed).

A *scan configuration* is the logic state of all sequential elements and external control inputs. It is assumed that after reset (or power-up) all sequential elements are in a known state (either a '0' or a '1'). A scan configuration is *valid* if and only if: (i) at least one active scan path exists and (ii) scan segments that do not belong to the active scan path are deselected. This ensures that the shift-in data is delivered to the target scan segments, the captured data is shifted towards the primary scan-out, and all scan segments that do not take part in the access (i.e., do not belong to the active scan path) are stable.

The basic access to the scan network is an atomic (inseparable) operation that consists of three phases: *capture, shift, and update* (CSU), cf. Fig. 2b. During capture, the shift registers on the active scan path may latch new data. This data is shifted out during the shift phase, while new scan data is shifted in. Finally, during the update phase, the shifted-in data is latched in the shadow registers on the active scan path.

A read or write access to a scan register in the network requires that the accessed register is part of an active scan path (cf. Fig. 1). A *scan access* is a sequence of CSU operations required to reconfigure the scan network and access the target registers. *Access time* is the number of clock cycles that are required to perform the scan access, including the update and capture cycles of each CSU.

For the sake of brevity, we assume that the input *select* of a scan segment enables all the three phases of a CSU operation: Whenever a segment is selected, its state is both captured and updated by the CSU operation. An extension with *capture/update disable* control signals is straightforward [6].

III. CSU-ACCURATE RSN MODEL (CAM)

Our scan network model follows the modeling approach presented in [6]. It captures the structural and functional characteristics of RSNs in a formal way. The model can be easily derived from any structural description of an RSN: either a gate- or RT-level netlist or high-level representations, e.g. Instrument Connectivity Language (ICL) defined by P1687. In the following, we present a compact representation of this model.

Definition 3.1: The Capture-shift-update-Accurate Model (CAM) of an RSN is a tuple $M = \{S, I, C, \texttt{Active}\}$ that consists of a set of state elements S, a set of external control inputs I, a set of scan configurations C, and a predicate function $\texttt{Active}$. Each state element $s \in S$ corresponds uniquely to a 1-bit scan register in the network. A scan configuration $c \in C : C \subseteq (S \cup I) \times \{0, 1\}$ is the state of all elements in S and control inputs in I. Let $c(s) : S \mapsto \{0, 1\}$ be the function that denotes the state of $s \in S$ under configuration $c \in C$. The predicate function $\texttt{Active} : C \times S \mapsto \{0, 1\}$ assigns each state element $s \in S$ in scan configuration $c \in C$ a Boolean predicate $\texttt{Active}(c, s)$. Predicate $\texttt{Active}(c, s)$ is 1 iff the scan segment corresponding to s belongs to the active scan path in c and c is valid.

In the following, we describe the construction of predicates (Sec. III-A) and we define a characteristic function that captures the effect of a CSU operation (Sec. III-B).

A. Valid Scan Configuration

To construct the predicates, we need to distinguish valid scan configurations from invalid ones (cf. Sec. II). To this end, we construct a Boolean function $V : C \mapsto \{0, 1\}$ that evaluates to 1 iff the scan configuration is valid, i.e. when there exists a well formed scan path and all off-path scan segments are deselected. We construct function V piecewise as a conjunction of the form $V = \bigwedge_{s \in S} v(s)$, where $v(s)$ evaluates to true iff the local scan configuration of the scan segment corresponding to s is valid, as explained below.

For a scan segment s with a single predecessor $p \in \texttt{pred}(s)$ and a single successor $n \in \texttt{succ}(s)$ (cf. Fig. 3a), it is required that both p and n be selected if s is selected, such that scan data is not lost. Thus:

$$v(s) := select(s) \rightarrow [select(p) \wedge select(n)]$$

For a scan segment s with a single predecessor p and multiple successors (cf. Fig. 3b), a valid scan configuration requires that exactly one successor of s is selected if s is selected. This is captured by the following formula $v(s)$ which combines formulas (1) and (2) below:

$$v(s) := [select(s) \rightarrow select(p)] \wedge (1) \wedge (2)$$

$$select(s) \rightarrow \bigvee\nolimits_{n \in \texttt{succ}(s)} select(n) \qquad (1)$$

$$\forall_{n_k, n_l \in \texttt{succ}(s), n_k \neq n_l} : [select(n_k) \rightarrow \neg select(n_l)] \qquad (2)$$

This assures that in case of a branching scan path (fanout>1) only one branch is active, i.e. there are not multiple selected successors.

For a scan segment s with a single successor n and multiple predecessors selected by a multiplexer (cf. Fig. 3c), a valid scan configuration requires that exactly one predecessor of s is selected if s is selected. If a predecessor of s is selected, the *address* of the multiplexer must be correctly set:

$$v(s) := [select(s) \rightarrow select(n)] \wedge (3) \wedge (4)$$

$$select(s) \rightarrow \bigvee\nolimits_{p \in \texttt{pred}(s)} select(p) \qquad (3)$$

$$\forall_{p \in \texttt{pred}(s)} : [select(p) \rightarrow address = addr(p)] \qquad (4)$$

This assures that in case of a multiplexed scan path the active path is correctly routed.

In case of a node s with multiple predecessors and multiple successors, the following formula captures the condition for a valid scan configuration: $v(s) := (1) \wedge (2) \wedge (3) \wedge (4)$.

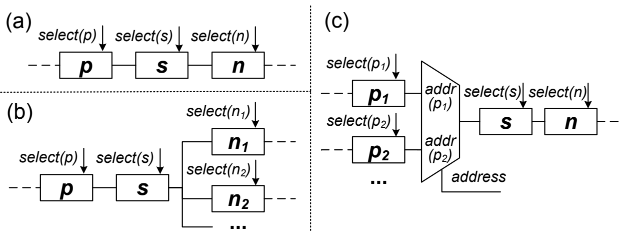

Figure 3. (a) Chained, (b) branching and (c) multiplexed scan substructures

With the function V, predicate $\texttt{Active}(s)$ is defined as:

$$\texttt{Active}(s) = select(s) \wedge V.$$

The Boolean $select$ functions are obtained by traversing the input cones of these control signals in the netlist.

B. Transition Relation of the CSU-Accurate Model (CAM)

The CAM transition relation models the effect of a CSU operation which we consider atomic. A CSU operation may arbitrarily change the state of all scan segments on the active scan path, since any data may be shifted into those segments from the primary scan input. We capture this behavior with a transition relation, as defined below.

Definition 3.2: The transition relation of a CAM $M = \{S, I, C, \texttt{Active}\}$ is defined as a set $T \subseteq C \times C$ with the following characteristic function:

$$T(c_1, c_2) := \bigwedge_{s \in S} [(c_1(s) \oplus c_2(s)) \rightarrow \texttt{Active}(c_1, s)]$$

where $c_1, c_2 \in C$. The transition relation T includes all pairs of scan configurations (c_1, c_2), such that c_2 can be reached from c_1 within one CSU operation.

The characteristic function of the transition relation defines the requirement for state changes: If the state of an element s differs in two consecutive scan configurations c_1 and c_2, the active scan path must be well formed and the corresponding scan segment must be selected in scan configuration c_1, which is when $\texttt{Active}(c_1, s) = 1$, i.e. when s is selected in a valid scan configuration c_1.

IV. Pattern Retargeting

An access to a scan segment may require several CSU operations to put the target scan segment on the active scan path. In the following, we formulate the problem of computing the minimal (shortest) scan-in sequence that implements an access. As the search for the global minimum may be prohibitively expensive, we propose an affordable pattern generation procedure which is based on pseudo-Boolean optimization. The proposed method is applicable to *access merging*, i.e. generation of efficient scan-in sequences that access multiple scan elements during one or multiple CSU operations.

A. Problem Formulation

We search for the scan pattern sequence that must be shifted into the RSN during one or multiple CSU operations to reach a certain target scan configuration with minimal access time. We specify a scan access by its initial scan configuration $c_0 \in C$ and target scan configuration $c_t \in C$. We denote the access by (c_0, c_t).

Given is the CAM of an RSN $M = \{S, I, C, \texttt{Active}\}$ with transition relation T, and a scan access (c_0, c_t). Scan pattern generation is the computation of a valid sequence of $n \in \mathbb{N}^+$ consecutive scan configurations $c_1, c_2, \ldots, c_n$ such that the following conditions hold:

$$c_n = c_t \wedge \forall_{i=1\ldots n} \left((c_{i-1}, c_i) \in T \right); \qquad (5)$$

and the solution minimizes the scan access time (number of required clock cycles) expressed with the following pseudo-Boolean cost function:

$$\texttt{Cycles}(c_0, c_1, \ldots, c_n) := 2n + \sum_{i=0}^{n-1} \sum_{s \in S} \texttt{Active}(c_i, s). \qquad (6)$$

Note that n is the number of CSU operations required for the optimal solution, which is a priori unknown.

Condition (5) is satisfied iff $c_0, c_1, \ldots, c_n$ is a valid sequence of consecutive scan configurations, such that the last configuration equals the target scan configuration. Scan access time $\texttt{Cycles}(c_0, c_1, \ldots, c_n = c_t)$ given by formula (6) amounts to the number of capture and update cycles ($2n$) plus the number of required shift cycles in each scan configuration, except for the target scan configuration c_t. The number of required shift cycles, i.e. the scan-in pattern length, equals the number of predicates that evaluate to 1, since each predicate corresponds to a 1-bit scan register, and the predicate is true iff the corresponding scan register is part of the active scan path.

The search for the minimal access sequence with the globally minimal access time is a hard problem. The search space has two dimensions: the number of required CSU operations n, and the Boolean state of all scan segments in $n+1$ scan configurations. Note that the global minimum is not necessarily found for the minimal number $n_{\texttt{min}}$ of CSU operations required to perform the access, i.e. to satisfy formula (5). Often, the access time can be reduced by allowing *additional* CSU operations (see Fig. 4).

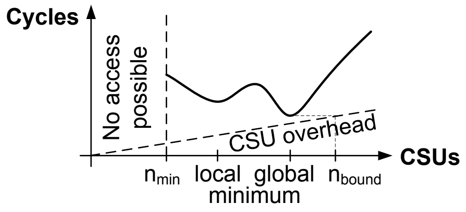

Figure 4. Example of a minimal access time curve

In principle, the global minimum can be found with an iterative procedure: Let $\texttt{Cycles}_n$ be the minimal access time with n CSU operations. Compute the shortest access sequences for $n = 1, 2, \ldots, n_{\texttt{bound}}$ CSU operations. Note that a CSU operation always incurs an access overhead of 2 cycles (cf. formula (6)). Thus, a solution with n CSU operations is the global minimum if the access time of all solutions with up to $n_{\texttt{bound}}$ CSU operations is not less than $\texttt{Cycles}_n$, where:

$$n_{\texttt{bound}} < \lceil \texttt{Cycles}_n / 2 \rceil.$$

In this case, the solution with n CSU operations is guaranteed to be the global minimum since the access time of all solutions with at least $n_{\texttt{bound}}$ CSU operations is at least $\texttt{Cycles}_n$ due to the overhead of 2 cycles per CSU operation (cf. Fig. 4).

In practice, due to high computational effort, the search for the minimal solutions with up to $n_{\texttt{bound}}$ CSU operations may be too expensive. Our experiments show, however, that the access time is significantly reduced by allowing just a few additional CSU operations.

The following Sec. IV-B explains merging of concurrent read and write accesses to multiple scan segments. Sec. IV-C describes how we generate an access sequence with the minimal access time for a given (fixed) number of CSU operations. In Sec. IV-D we present an affordable pattern generation procedure.

B. Access Merging

The challenge of access merging is to find the optimal order of multiple accesses to scan segments that results in a minimal scan-in sequence. The target scan segments must have their target values in the final scan configuration c_t, but the order in which the merged accesses are performed is not specified. It is therefore sufficient to specify the concurrent access to multiple scan segments by its initial and target scan configurations (c_0, c_t)

Specifying read accesses in this way restricts them to the last CSU operation. To improve merging flexibility, a read access may be specified by ensuring that during n CSU operations the target segment $s \in S$ belongs to the active scan path in one of the *intermediate* scan configurations, i.e. $\bigvee_{i=0\ldots n-1} \texttt{Active}(c_i, s)$ is true. Condition (5) is extended with such disjunctions for read accesses.

C. Mapping to Pseudo-Boolean Optimization

A pseudo-Boolean optimization problem is to find an assignment to the Boolean variables $(x_1, x_2, \ldots, x_k)$ that leads to the minimal value of a pseudo-Boolean cost function A among all assignments that satisfy a Boolean formula Ψ. The pseudo-Boolean cost function has the form:

$$A(x_1, x_2, \ldots, x_k) = c_0 + \sum_{i=1}^{k} c_i \cdot x_i$$

where $c_0, c_1, \ldots, c_k \in \mathbb{Z}$ and $x_i \in \{0, 1\}$. Pseudo-Boolean optimization can be performed for instance by incremental SAT solving techniques with pseudo-Boolean constraints translation to SAT [8], or speculative model enumeration techniques [9].

According to condition (5), an access (c_0, c_t) is implemented by a sequence of scan configurations $c_0, c_1, \ldots, c_n$ iff the following Boolean formula is satisfied:

$$\texttt{Access}(c_0, c_t, n) := \bigwedge_{s \in S} (c_n(s) = c_t(s)) \wedge \bigwedge_{i=1\ldots n} T(c_{i-1}, c_i) \qquad (7)$$

This formula is transformed into a conjunctive normal form (CNF) or a set of clauses. $c_i(s)$ are considered free variables for $i = 1 \ldots n-1$ and $s \in S$. If the formula is satisfiable, there exists a sequence of scan configurations $c_0, c_1, \ldots, c_n$ that describes a valid scan access, such that $c_n = c_t$. Otherwise, if the formula is unsatisfiable, no scan access with n CSU operations exists.

The formula $\texttt{Access}(c_0, c_t, n)$, given by (7), is subject to pseudo-Boolean optimization with the cost function $\texttt{Cycles}(c_0, c_1, \ldots, c_n)$, given by (6). The *satisfying assignment* (optimization solution) provides the state of all scan segments in scan configurations $c_1 \ldots c_{n-1}$. The scan-in sequence that implements the scan access is derived from the satisfying assignment: The i-th CSU operation is fully specified by a pair of scan configurations c_{i-1} and c_i. Configuration c_{i-1} specifies the active scan path. An element $s \in S$ belongs to the active scan path iff $\texttt{Active}(c_{i-1}, s) = 1$. Configuration c_i specifies the content of scan segments and so provides the scan-in sequence for the i-th CSU operation. The scan-in sequence is guaranteed to have the minimal access time in number of clock cycles for the given number of n CSU operations.

D. Pattern Generation Procedure

Our pattern generation procedure is based on a heuristic that finds a local access time minimum (cf. Fig. 4): We search for access sequences with increasing number of CSU operations as long as allowing more CSU operations provides a reduction of access time.

Let $\texttt{Cycles}_n$ be the value of the cost function (6) after optimization with n CSU operations. Potentially, a solution with lower access time can be found if more CSU operations are allowed. The SAT instance is extended to $n+1$ CSU operations to find the value of the cost function $\texttt{Cycles}_{n+1}$. If the cost of the new solution is higher than the previous one, i.e. when $\texttt{Cycles}_{n+1} > \texttt{Cycles}_n$, the pattern generation procedure terminates. Otherwise, the number of CSU operations is increased and the procedure is repeated until the user specified bound $n_{\texttt{max}}$ is reached.

Let n_t be the number of CSU operations at which the pattern generation procedure terminates. The procedure guarantees that the final solution has the minimal access time among all solutions with $n \leq n_t + 1$ CSU operations. There may exist a global minimum with lower access time that requires $n_{\texttt{opt}} > n_t + 1$ CSU operations. However, experimental results show that increasing the number of CSU operations beyond $n_t + 1$ rarely provides better results and leads to high solve times.

E. Implementation

The pattern generation procedure is implemented using the *clasp* toolkit [10], which includes a Boolean SAT solver and a pseudo-Boolean optimization engine. As the SAT solver is generally faster than the pseudo-Boolean optimizer, we use it to initially find the minimal number of CSU operations $n_{\texttt{min}}$ that is required to implement the access. After $n_{\texttt{min}}$ is found, pseudo-Boolean optimization is performed for increasing number of CSU operations, as described in Sec. IV-D. Incremental solving techniques are employed: The Boolean formula (*SAT instance*) generated for n CSU operations is extended with additional clauses for the characteristic function of the transition relation and reused in iteration $n + 1$.

Our framework exploits parallelism in the pattern generation procedure: After the minimal number of CSU operations is found, the optimization of SAT instances with increasing number of CSU operations is processed in parallel. Each optimization job is performed in a child process. Fig. 5 illustrates the parallel execution of the pattern generation procedure.

V. Evaluation

The proposed method is evaluated on several RSN designs based on ITC'02 benchmarks [11]. We analyze the access time obtained with the pattern generation procedure w.r.t. a solution with the minimal number of CSU operations, as in [6]. The experiments are run on an Intel Xeon CPU with 12 cores operating at 3.33 GHz.

A. Benchmark Circuits

We evaluate our approach on two RSN architectures, as described in [6]: hierarchical structures implemented with multiplexers and Segment Insertion Bits (SIBs).

The *MUX-based* architecture supports two access modes: configuration access and data access. Configuration access allows to reconfigure the scan chain by attaching or detaching internal scan segments or submodules. Fig. 6 shows the MUX-based architecture for the top-level part of the p34392 benchmark. The scan chain of each module starts with a 1-bit configuration register AM that sets the configuration mode ($AM = 0$), in which only the configuration registers (C) can be accessed, or data access mode ($AM = 1$). Once configured, this architecture is faster compared to the SIB-based scheme, as less control registers are present on the active scan path in the data access mode.

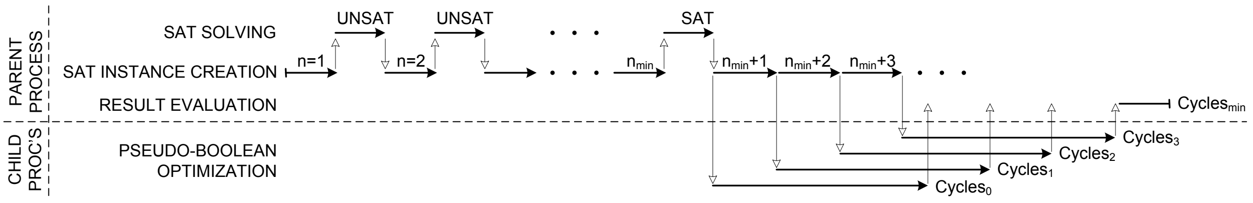

Figure 5. Parallel execution of the pattern generation procedure

The *SIB-based* scan architecture implements hierarchical scan bypasses with SIBs. A SIB consists of a 1-bit configuration register and a scan multiplexer that either bypasses or connects the lower-level scan segment (or a scan network) to the higher-level scan chain, depending on the content of the configuration register. The scan chain encompassing a single module is composed of several SIBs, as proposed in [4]. The SIBs provide configurable access to the scan segments of the core, its submodules, as well as its inputs and outputs. Fig. 7 shows such a scan architecture for the top-level part of the p34392 benchmark.

Table II presents the properties of MUX-based architectures: the number of multiplexers in the second column, the total number of scan segments (including configuration segments) in the third column, and the total number of scan register bits in the fourth column. The characteristics of the SIB-based architectures are listed in the first four columns of Table III.

B. Validation of Results

The results presented in the following section are validated by cycle-accurate simulation in a commercial logic simulator. For this purpose, the RSN models are automatically translated to hardware Verilog models. The generated patterns are used as stimuli for the primary scan input of a network. During simulation, assertions verify that the scan access is performed correctly.

C. Results

For each design, we perform 1000 pattern generation experiments. In each experiment, we search for the shortest scan-in sequence that *merges* read or write accesses to 10 randomly chosen scan segments. Optimization is performed with up to 6 additional CSU operations, executed in 6 parallel jobs (cf. Fig. 5).

Column “No optimization” in Table II and III presents the results of pattern generation without optimization. A SAT solver is used to iteratively check the satisfiability of instances with increasing number of CSU operations until a solution is found, as in [6]. For the 1000 experiments, column $n_{\mathtt{min}}$ gives the average and maximal number of CSU operations that are required to implement an access. Column $t_{\mathtt{avg}}$ gives the average pattern generation time per access. The average access time of the unoptimized patterns is given in column `cycles` in clock cycles.

The proposed method is evaluated in two series of experiments, limiting the maximal effort of the pattern generation procedure to 2 and 20 s per access. Table II and III give the average and maximal *access time reduction* (column `reduction`) w.r.t. to the unoptimized solution. The average and maximal number of additional CSU operations that are required to obtain the best solution is given in column $n_t - n_{\mathtt{min}}$.

The proposed method significantly reduces the access time for the MUX-based architectures (Table II): For almost all circuits, a maximal access time reduction of over 10x is achieved. For the t512505 benchmark, the access time is reduced by up to 88x. Compared to our previous results obtained with the MiniSat SAT solver [6], the proposed method achieves up to 230x access time reduction (not presented in the table). This shows that access optimization is crucial to prevent solutions with prohibitive access time. The proposed method also reduces unnecessary access overhead: For most of the benchmarks, the average access time over the 1000 experiments is nearly halved within 2 s of computational time.

For SIB-based architectures, efficient scheduling techniques exist for access time minimization [5]. Such architectures do not require a formal approach since optimal pattern generation reduces to a simple decision problem. Nevertheless, our method can also be applied for such regular structures. The results show that our pattern generation procedure reduces the access time for SIB-based architectures by a factor of up to 1.8x (Table III). In contrast to MUX-based architectures, the local minimum is always found for the minimal number of CSU operations that is required to implement the access ($n_{\mathtt{min}}$). The local minimum is usually found within 2 s of optimization. Extending the effort to 20 s achieves only a marginal access time reduction for larger benchmarks (italic in Table III).

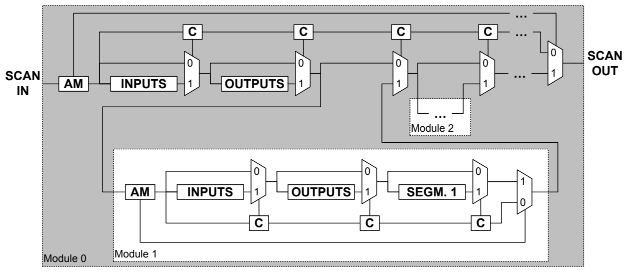

Figure 6. MUX-Based scan architecture for the p34392 benchmark

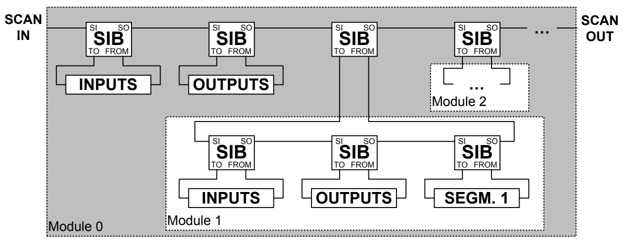

Figure 7. SIB-based scan architecture for the p34392 benchmark

Table II: ACCESS TIME REDUCTION (`reduction`) FOR THE MUX-BASED SCAN ARCHITECTURE W.R.T. UNOPTIMIZED SOLUTION (`cycles`)

Design	Num. MUX	Total scan segm.	Total scan bits	No optimization			Optimization effort $2s$		Optimization effort $20s$	
				$n_{\mathtt{min}}$ avg / max	$t_{\mathtt{avg}}$ [s]	`cycles` [cycles]	$n_t - n_{\mathtt{min}}$ avg / max	`reduction` avg / max	$n_t - n_{\mathtt{min}}$ avg / max	`reduction` avg / max
u226	59	99	1 475	5.6 / 7	0.03	705	0.4 / 3	1.54 / 6.8x	0.4 / 2	1.54 / 6.8x
d281	67	117	3 880	5.7 / 7	0.03	1718	0.8 / 2	1.90 / 13.4x	0.8 / 3	1.91 / 13.7x
d695	178	335	8 407	6.0 / 7	0.09	3569	0.5 / 4	1.78 / 11.2x	0.7 / 4	1.84 / 11.2x
h953	63	109	5 649	5.7 / 7	0.03	2776	0.9 / 3	1.91 / 16.1x	0.9 / 3	1.91 / 16.1x
g1023	94	159	5 400	5.9 / 7	0.04	2482	0.5 / 2	1.89 / 10.7x	0.6 / 2	1.93 / 10.7x
f2126	45	81	15 834	5.6 / 7	0.02	9327	0.8 / 3	1.78 / 12.1x	0.9 / 3	1.79 / 12.1x
q12710	30	51	26 188	5.7 / 7	0.01	17769	0.8 / 3	1.78 / 12.3x	0.8 / 3	1.78 / 12.3x
p22810	311	565	30 139	6.0 / 10	0.17	12335	0.5 / 4	1.65 / 33.3x	0.5 / 3	1.75 / 33.7x
p34392	142	245	23 261	6.9 / 10	0.09	14633	0.7 / 3	2.02 / 49.2x	0.8 / 4	2.16 / 49.2x
p93791	653	1241	98 637	6.0 / 9	0.38	21073	0.8 / 4	1.84 / 28.2x	0.9 / 4	1.99 / 28.2x
t512505	191	319	77 037	5.7 / 7	0.09	22146	0.5 / 3	2.31 / 87.8x	0.5 / 3	2.39 / 87.8x
a586710	47	79	41 682	6.3 / 10	0.02	36417	1.2 / 6	2.19 / 74.1x	1.3 / 5	2.26 / 74.1x

Table III: ACCESS TIME REDUCTION (`reduction`) FOR THE SIB-BASED SCAN ARCHITECTURE W.R.T. UNOPTIMIZED SOLUTION (`cycles`)

Design	Num. SIB	Total scan segm.	Total scan bits	No optimization			Optim. effort $2s$	Optim. effort $20s$
				$n_{\mathtt{min}}$ avg / max	$t_{\mathtt{avg}}$ [s]	`cycles` [cycles]	`reduction` avg / max	`reduction` avg / max
u226	50	90	1 466	2.6 / 3	0.01	879	1.09 / 1.81x	1.09 / 1.81x
d281	59	109	3 872	2.7 / 3	0.02	2039	1.13 / 1.81x	1.13 / 1.81x
d695	168	325	8 397	2.7 / 3	0.04	4294	1.14 / 1.61x	*1.15* / 1.61x
h953	55	101	5 641	2.7 / 3	0.01	3110	1.16 / 1.69x	1.16 / 1.69x
g1023	80	145	5 386	2.7 / 3	0.02	2507	1.17 / 1.62x	1.17 / 1.62x
f2126	41	77	15 830	2.5 / 3	0.01	9662	1.11 / 1.72x	1.11 / 1.72x
q12710	25	47	26 183	2.5 / 3	0.01	16550	1.09 / 1.68x	1.09 / 1.68x
p22810	283	537	30 111	2.8 / 4	0.08	12009	1.06 / 1.25x	*1.12 / 1.50*x
p34392	123	226	23 242	3.1 / 4	0.04	13122	1.16 / 1.45x	*1.17 / 1.66*x
p93791	621	1209	98 605	2.9 / 4	0.19	36278	1.09 / 1.24x	*1.14 / 1.48*x
t512505	160	288	77 006	2.7 / 3	0.04	35275	1.13 / 1.57x	*1.18 / 1.74*x
a586710	40	72	41 675	2.8 / 4	0.01	24618	1.13 / 1.85x	1.13 / 1.85x

The results presented in Table II and III are obtained with the pattern generation procedure of Section IV that terminates as soon as a local minimum is found. In additional experiments we computed shortest access sequences with 6 additional CSU operations over n_t. Despite the additional effort, the resulting access times are exactly the same as those obtained in the proposed algorithm. For all the examined circuits, the proposed algorithm provides the best achievable solution among all solutions with at most 6 additional CSU operations.

VI. CONCLUSION

Reconfigurable scan networks allow scalable access to on-chip infrastructure. The design complexity due to hierarchies and IP reuse requires novel EDA tools for optimal scan pattern generation. In this work, we present the mapping of this problem to a pseudo-Boolean optimization problem. This novel method is applicable to a wide range of reconfigurable architectures and to merging of multiple concurrent scan accesses. It provides the optimal access time for a given bound on the number of access operations. The experiments demonstrate that, contrary to common sense, it is often necessary to allow more access operations to reduce the overall access time. The results show that even for complex reconfigurable scan architectures the proposed method leads to significant reduction of access time by up to 88x with low computational effort.

ACKNOWLEDGEMENTS

Parts of this work were supported by the German Research Foundation (DFG) under grants WU 245/11-1 and WU 245/13-1.

REFERENCES

[1] N. Stollon, *On-Chip Instrumentation: Design and Debug for Systems on Chip*. Springer US, 2011.

[2] E. Larsson and K. Sibin, "Fault Management in an IEEE P1687 (IJTAG) Environment," in *IEEE Int'l Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2012, p. 7.

[3] J. Rearick and A. Volz, "A Case Study of Using IEEE P1687 (IJTAG) for High-Speed Serial I/O Characterization and Testing," in *Proc. IEEE International Test Conference (ITC)*, 2006, paper 10.2.

[4] F. Zadegan, U. Ingelsson *et al.*, "Design Automation for IEEE P1687," in *Proc. Design, Automation Test in Europe Conference (DATE)*, 2011, pp. 1412–1417.

[5] E. Larsson and F. Zadegan, "Accessing Embedded DfT Instruments with IEEE P1687," in *Proc. IEEE Asian Test Symposium (ATS)*, 2012, pp. 71–76.

[6] R. Baranowski, M. A. Kochte, and H.-J. Wunderlich, "Modeling, Verification and Pattern Generation for Reconfigurable Scan Networks," in *Proc. IEEE Int'l Test Conf. (ITC)*, 2012, paper 8.2.

[7] F. Ghani Zadegan, U. Ingelsson *et al.*, "Access Time Analysis for IEEE P1687," *IEEE Trans. Computers*, vol. 61, no. 10, pp. 1459–1472, October 2012.

[8] N. Eén and N. Sörensson, "Translating Pseudo-Boolean Constraints into SAT," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, pp. 1–26, 2006.

[9] M. Gebser, R. Kaminski *et al.*, "Multi-Criteria Optimization in Answer Set Programming," in *Technical Communications of the Int'l Conf. on Logic Programming (ICLP)*, ser. LIPIcs, vol. 11, 2011, pp. 1–10.

[10] M. Gebser, B. Kaufmann *et al.*, "clasp : A Conflict-Driven Answer Set Solver," in *Logic Programming and Nonmonotonic Reasoning*, ser. LNCS. Springer, 2007, vol. 4483, pp. 260–265.

[11] E. Marinissen, V. Iyengar, and K. Chakrabarty, "A Set of Benchmarks for Modular Testing of SOCs," in *Proc. IEEE International Test Conference (ITC)*, 2002, pp. 519–528.