An ILP Formulation to Optimize Test Access Mechanism in System-on-Chip Testing

Mehrdad Nourani
Center for Integrated Circuits & Systems
The Univ. of Texas at Dallas
Richardson, TX 75083-0688
nourani@utdallas.edu

Christos Papachristou
Department of EECS
Case Western Reserve Univ.
Cleveland, OH 44106-7071
cap@eecs.cwru.edu

Abstract

We present an optimization method that complies with IEEE P1500 draft standard and deals with modeling and design of the test access mechanism for the SoCs. The basic goal is to develop a global design for test methodology and optimization technique for testing a core-based SoC in its entirety. We propose an ILP formulation to minimize the hardware cost or the overall access time which also produces the test access schedule.

1 Introduction

1.1 Motivation

A fundamental change has taken place in the way digital systems are designed. It has become possible to design an entire system, containing million of transistors, on a single chip. In order to cope with the growing complexity of such modern systems, designers often use pre-designed, reusable megacells known as cores [ChPa96]. Typical cores include memories, microprocessors, ASICs etc. Core-based system-on-chip (SoC) design strategies help companies significantly reduce the time-to-market and design cost for their new products. The Semiconductor Industry Association's National Technology Roadmap [SIA97] shows the percentage of reused cores in SoCs rising from 40% in 1997 to 80% in 2006 while it predicts a 50% reduction in time-to-market over the same period. Unfortunately, core-based SoCs are difficult to test after fabrication. Some difficulties stem from the sheer complexity inherent in putting an entire system's functionality on a single chip. Specifically, flexible test access mechanisms are required to "control" and "observe" what is happening in the circuits embedded deeply inside the chip.

1.2 Background

A major difficulty concerns accessibility of embedded cores from the I/O terminals of the system. There are three main approaches to deliver test patterns to an individual core and observing core responses. One approach uses extra test circuitry around the core to *isolate* the core during test. For example, in [ImRa90] an extra multiplexor is placed at each core input so that the input is directly accessible from the system input. Alternatively, [ToPo97] inserts a partial isolation ring, similar but cheaper than a full scan chain, around the core inputs and outputs so that they can be accessed serially. The second approach uses a *transparency* [GhJD97] or *bypass* [NoPa99] mode for embedded cores to reduce the problem to one of finding paths from the system inputs to the core inputs and from the core outputs to the system outputs. A similar concept, called *transfer*, was used in [MaLo99] to transport test data at different levels of circuit hierarchy while a *test protocol* formally specified how this transportation is executed for a core under test. The third approach is based on test bus architectures by which the cores are isolated from each other in test mode using a dedicated bus [VaBh98][Chak00] or a flexible *TESTRAIL* [MABD98] around the cores to transfer test data. Test time reduction is the main objective in the majority of these methods. For example, [Chak00] used an integer linear programming (ILP) formulation to find the best test assignment and optimize the bandwidth distribution among various test buses to minimize time.

In the IEEE P1500 draft document (evolving standard) [P1500] [MZKT99] there are three key components: (1) *Core Test Wrapper* provides mechanisms for core test data access and core test isolation; (2) *Test Control Mechanism* whose purpose is to enable and control test modes, test operations and test signals, within the system chip; and (3) *Test Access Mechanism* to transfer test data between the primary I/O of the system chip and the core test wrapper. Test access mechanism is

0-7803-6546-1/00 $10.00

system chip specific and is defined by the designer during system chip integration.

Scan-based techniques to test SoCs show some difficulties [Whet97][MABD98]. The interconnection and interfaces are not tested thoroughly. More importantly, scan prevents at-speed testing of the core; when scan chains are used, for example, it takes several clocks to apply each test pattern. Additionally, in some applications, some cores are preferred to be kept as one group during test to allow for example high-speed interaction among the cores in that group. This coincides with our previous works in [NoCP97], [CaNP99] and [PaMN99] where we showed that multi-core partition testing (e.g. for datapath-controller pair) can be quite advantageous in terms of test overhead and detecting some faults that can easily escape detection.

1.3 Significance of Proposed Method

Designing industrial SoCs with testability in mind is an economical necessity. The rising level of complexity of chips makes it increasingly difficult to achieve an adequate system test using ad-hoc techniques currently practiced in industry. Many experts agree that unless structured methods for system level testability are developed, the next bottleneck in design will be related to testing the SoCs [P1500]. Given a set of cores, and their individual test information, to be integrated into a system-on-chip, our goal is to devise a test access mechanism whereby a complete test of the system can be accomplished quickly and inexpensively.

Most of the access mechanisms mentioned earlier ignore the existing core structural model and the interface architecture among them. We think in the majority of SoCs the existing core topology can facilitate testing them and so that insertion of separate access components (e.g. bypass, buffers) would be needed only in special circumstances. Exploring these features is the main contribution of this work. Specifically, we propose two techniques:

1. Taking advantage of all "access elements" (e.g. MUXes, controllable buffers, bypass routes, tri-state ports etc.), generally called *bridges* in this paper, by utilizing the existing ones or adding new ones whose cost is justified from the designer's point of view. Additionally, the notion of bridge element allows design and test access mechanism to be mixed during test synthesis approaches.
2. Formulating the SoC test access mechanism as an ILP problem to minimize the overall test time and/or the bridge overhead to access all cores in the SoCs. The size of this ILP formulation is very reasonable since it depends on bridges and the overall number of input/output ports of all cores in the SoC.

Based on these two techniques, we propose a systematic methodology to develop the test access mechanism (TAM) for designing easily testable core-based system. The TAM unit utilizes usage of the interconnect bandwidths and allows core access/testing in an overlapped fashion to shorten the overall test time. One flexibility of our formulation allows any selected cores to be controlled and observed for testing as a group. This has been shown to make it easier to achieve true at-speed testing by detecting faults or defects that escape from isolation-based core testing (e.g. cross-talk, overshoots among high-speed interacting cores) or other defects that degrade the performance but not necessarily the functionality, (e.g. causing excessive power dissipation) [NoPa99].

1.4 Paper Organization

The rest of the paper is organized as follows. The bridge components, the SoC model and the basic ILP formulation to minimize overall bridge overhead are explained in Section 2. Section 3 expands the ILP formulation to minimize test time and to produce the TAM test schedule. The experimental results are discussed in Section 4. Finally, the concluding remarks are in Section 5.

2 Modeling and Formulation

2.1 Bridges: The Access Components

We call any *controllable* component that allows two points to be connected in an SoC a *bridge*. The controllability is the key issue which allows flexibility in connecting (or electrically separating) two points. Figure 1 shows some examples of bridges. Figure 1(a) assumes that *Core A* and *Core C* have tri-state outputs controlled by its output enable (*oe*) signal. These bridges can be viewed as bridges internal to the cores. Figure 1(b) shows a bypass route controlled by the multiplexor select (*sel*) input. Some companies such as Philips [MABD98] [MZKT99] have already embedded the bypass route inside their core to be P1500 standard compliant. However, in our modeling the bypass can be internal or external. Figure 1(c) and (d) show tri-state buffers to connect core ports through shared bus or between two buses which can be controlled by their enable (*en*) inputs. A Without loss of generality, a bidirectional bridge (e.g. buffer) is modeled as two single-direction bridges.

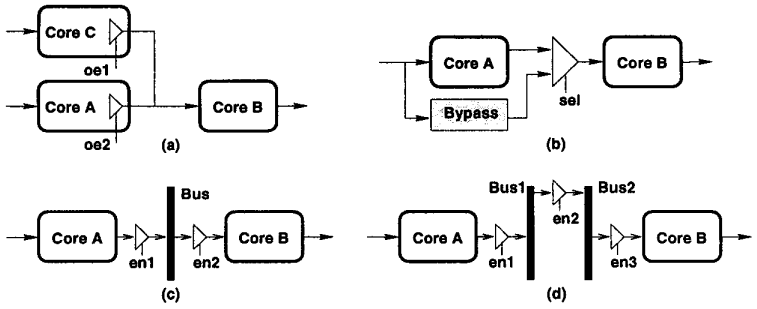

Figure 1: Typical bridges: (a) Tri-state outputs (b) bypass, (c) bus buffers, (d) bus-to-bus

Using the notion of bridges, TAM development becomes a bridge selection and bridge scheduling problem in general. Although such selection and scheduling is for test mode it will affect the overall cost and performance of the SoC. In general, our TAM optimization technique can be applied to the normal mode bridges (determined in the SoC design phase). Obviously, such bridges can be also used in test mode. However, when design and test coordination allows, it will be more beneficial to allow additional bridges to be used only in the test mode.

2.2 SoC Modeling

We first identify the core environment which comprises all the core input/output connections from/to chip primary input/output ports. Figure 2 shows such environment for a typical core. For simplicity, we just showed one input and one output test path.

In general there will be many ways of accessing input/outputs of a core all of which will go through a subset of bridges whose characteristics (in terms of cost, delay etc.) will affect the overall design and test features. Thus, selecting the best subset of bridges to provide all accessibility while minimizing overall test overhead or overall access time should be pursued. In [NoPa99] we presented a heuristic, limited to the bypass bridges, to identify the shortest path between core input/outputs and SoC input/outputs and overlap testing cores in a pipelined fashion. In this work, we use a more general and global approach. This selection needs to be done globally due to the fact that a local decision (e.g. to optimize overhead/time for one core) may significantly restrict accessing other cores. The necessity for such a global view in bridge selection motivated us to use the integer linear programming which globally solves the problem.

2.3 Preliminary ILP Formulation

This formulation finds two set of access paths for each core which eventually form the TAM; one from the SoC's primary inputs to the core inputs to be used for delivering test patterns, and one from the core outputs to the SoC primary outputs, to be used to observe core responses (see Figure 2). In this preliminary formulation all connections, buses, ports and bridges are assumed to have the same bitwidth (e.g. W) and the formulation does not produce the TAM test schedule to show when and how data is transferred to each core in the test mode. We will address these two issues in a more complete formulation in the next section.

Assume that the cores in SoC have totally n input/output ports ($P_1, P_2, \cdots, P_n$) that need to be controlled/observed. Also, assume that based on our topographical analysis of SoC there are m *bridges* (see Figure 1), each with a hardware cost C_i, to transfer test data. Finally, assume there are K_j possible paths (not necessarily disjoint) through which port P_j can be accessed. Other definitions are as follows for $1 \leq i \leq m$, $1 \leq j \leq n$ and $1 \leq k \leq K_j$:

$$x_i = \begin{cases} 1 & \text{if bridge } i \text{ is chosen to be in the TAM} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{j,k} = \begin{cases} 1 & \text{if path } k \text{ is used to access port } P_j \\ 0 & \text{otherwise} \end{cases}$$

$$a_{i,j,k} = \begin{cases} 1 & \text{if bridge } i \text{ is used in path } k \text{ to access port } P_j \\ 0 & \text{otherwise} \end{cases}$$

$a_{i,j,k}$ values are constant coefficients which are predetermined based on the topography of SoC, availability or designer choice. x_i and $y_{j,k}$ are the ILP variables. This version of the ILP formulation minimizes the test overhead (cost of bridges) while establishing access paths for all core input/output ports:

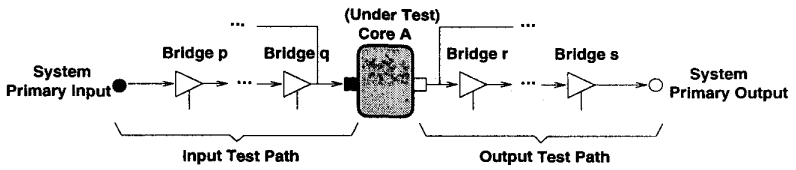

Figure 2: Core environment and the input/output test paths

Minimize $\sum_{i=1}^{m} C_i \cdot x_i$

subject to:

(i) $\sum_{k=1}^{K_j} y_{j,k} \geq 1$ for $1 \leq j \leq n$

(ii) $(\sum_{i=1}^{m} a_{i,j,k}) \cdot y_{j,k} \leq \sum_{i=1}^{m} (a_{i,j,k} \cdot x_i)$ for $1 \leq j \leq n\,;\, 1 \leq k \leq K_j$

(iii) $x_i, y_{j,k} \in \{0,1\}$ $\forall i,j,k$

The first constraint (n inequalities) guarantees that the selected bridges form at least one path to access port P_j. The second constraint ($\sum_{j=1}^{n} K_j$ inequalities) ties selection of bridges and paths together by making sure that the cost of all bridges are included once a specific path k is selected to access port P_j. In this constraint, $\sum_{i=1}^{m} a_{i,j,k}$ represents the total number of bridges in path $y_{j,k}$. So, if $y_{j,k} = 0$ then the corresponding path is not selected and thus the bridges in that path can be chosen ($x_i = 1$) or not ($x_i = 0$) to satisfy the constraint. On the other hand, if $y_{j,k} = 1$ then to correctly establish a path, all the bridges in that path must be selected ($x_i = 1$ for all bridges in the path). This is guaranteed by constraint (ii). The third constraint expresses that all variables can be only 0 or 1. Note that order of m, n and K_j makes this formulation of a manageable size such that almost any ILP software can solve it.

3 Generalization of the Test Methodology

3.1 Path Enumeration Using Depth First Search

As explained above, the first and second constraints use K_j, the number of paths that port P_j can be accessed and also the actual paths for traversal and search. In practice, due to the existence of many bridges and buses in a SOC the number of such paths may become large; thus we need a systematic approach to find them all. To do this, we construct a directed graph, called *bridge access graph* (BAG), corresponding to the SoC whose traversal provides all test paths. Each node in the BAG corresponds to a port (input or output) or to a bus (internal or I/O). Each edge in the BAG corresponds to a

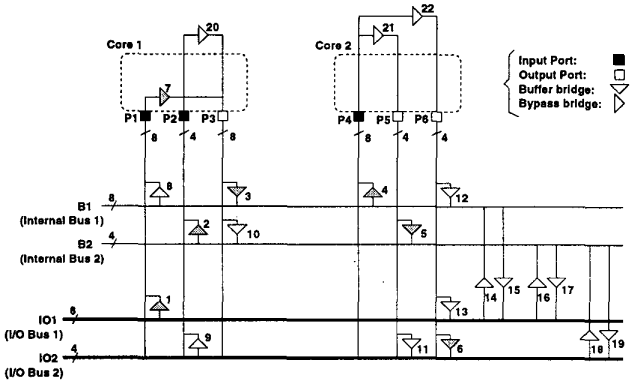

Figure 3: A small SoC example.

bridge (normal-mode or test-mode). The *normal-mode* bridges are the ones in the SoC to realize normal-mode behavior. Of course, the normal-mode bridges can be used in the normal or test mode. The *test-mode* bridges are added for the purpose of testing only and have no use in the normal mode. In our formulation, designer may add as many test-mode bridges as he/she perceive them as *potentially* useful to access cores in the test mode. Ultimately, the formulation selects a subset of them to optimize the objective function (e.g. cost or time).

A small SoC example and its corresponding BAG is shown in Figure 3 and Figure 4, respectively. The normal-mode and test-mode bridges are shown in Figure 3 as shaded and unshaded triangles. In this example $n = 6$ and $m = 22$. The index of bridges are written near each edge in Figure 4. Applying a depth first search (DFS) [Rich93] to the BAG can easily identify all paths to/from SoC ports. Specifically, starting at an output port and following edge directions, the DFS lists all paths from core outputs to the SoC's output ports. Similarly, starting at an input port and following the edges in opposite directions, the DFS lists all paths from the SoC's input ports to the core input ports.

We applied the classical DFS algorithm discussed in [Rich93]. However, an important modification is needed to guarantee a valid TAM solution. If in the search process, we revisit a node (port/bus) or an edge

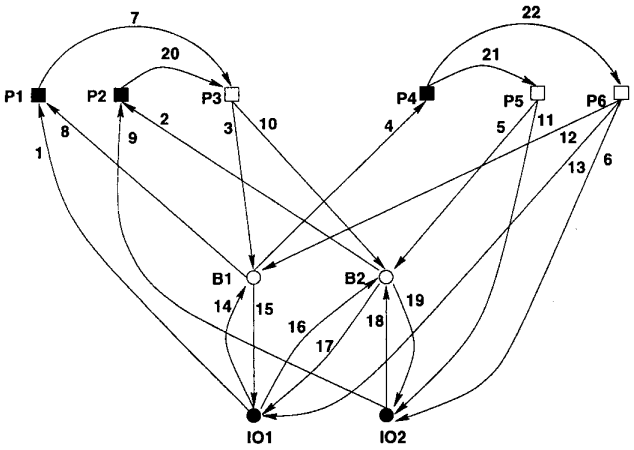

Figure 4: The Bridge Access Graph corresponding to the SoC example.

j	1	2	3	4	5	6
Port Pj	P1	P2	P3	P4	P5	P5
Kj	5	7	8	5	6	8

(a)

Path 4,1: P4 $\xleftarrow{4}$ B1 $\xleftarrow{14}$ IO1

Path 4,2: P4 $\xleftarrow{4}$ B1 $\xleftarrow{3}$ P3 $\xleftarrow{7}$ P1 $\xleftarrow{1}$ IO1

Path 4,3: P4 $\xleftarrow{4}$ B1 $\xleftarrow{3}$ P3 $\xleftarrow{20}$ P2 $\xleftarrow{9}$ IO2

Path 4,4: P4 $\xleftarrow{4}$ B1 $\xleftarrow{3}$ P3 $\xleftarrow{20}$ P2 $\xleftarrow{2}$ B2 $\xleftarrow{16}$ IO1

Path 4,5: P4 $\xleftarrow{4}$ B1 $\xleftarrow{3}$ P3 $\xleftarrow{20}$ P2 $\xleftarrow{2}$ B2 $\xleftarrow{18}$ IO2

(b)

Figure 5: Applying the DFS algorithm to the running example.

(bridge), we discard that path since it is an indication of a cycle. The running time of the DFS algorithm is $\Theta(||V||+||E||)$ and the storage requirement of $O(||V||)$, where $||\ ||$ denotes the set cardinality; and $||V||$ and $||E||$ are the number of nodes and edges in the graph, respectively [Rich93].

Ignoring the bitwidth of bridges and connections, the DFS reports 5 to 8 paths for the ports in this example as shown in Figure 5(a). For example, five paths found by the DFS to access P_4 are listed in Figure 5(b). The bridge indices used in the paths are shown above the arrows.

The identification of paths using BAG helps to build the constraints quickly. For example, constraints (i) and (ii) related to port P_4 are:

$$(i)\ \ y_{4,1} + y_{4,2} + y_{4,3} + y_{4,4} + y_{4,5} \geq 1$$

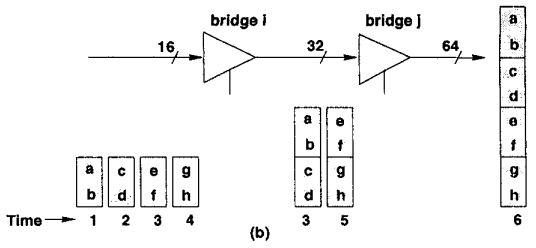

Figure 6: General strategy of multi-pass mechanism

$$(ii)\ \begin{cases} 2y_{4,1} \leq x_4 + x_{14} \\ 4y_{4,2} \leq x_4 + x_3 + x_7 + x_1 \\ 4y_{4,3} \leq x_4 + x_3 + x_{20} + x_9 \\ 5y_{4,4} \leq x_4 + x_3 + x_{20} + x_2 + x_{16} \\ 5y_{4,5} \leq x_4 + x_3 + x_{20} + x_2 + x_{18} \end{cases}$$

3.2 Bridge Cost

The hardware cost of a normal-mode bridge (if previously chosen) is assumed to be zero. The actual hardware cost of a test-mode bridge is proportional to the bitwidth of the connection and also it depends on the technology and the library cell. For simplicity, in this paper we assume that the bitwidth of a bridge represents its normalized cost. If p and q are the bitwidths of the two sides (e.g. buses B1 and B2), the bridge can allow only $min\{p, q\}$ bits and so the cost is assumed to be $min\{p, q\}$. The limitation that the bitwidths impose can be easily incorporated within the DFS algorithm. In the path search process for port P_j with bitwidth W_{P_j} the DFS visits some bridges (e.g. bridge i with bitwidth W_i). If at any time $W_i < W_{P_j}$ that path is discarded since the data transfer is not possible in a single-pass. In our running example, $W_{P_4} = 8$ and thus paths corresponding to $y_{4,3}$, $y_{4,4}$ and $y_{4,5}$ will be discarded since there is at least one 4-bit bridge (i.e. $W_{20} = 4$) in them.

A multi-pass strategy can be employed to allow passing $max\{p, q\}$ bit data through a $min\{p, q\}$ bridge. One possible strategy is pictured in Figure 6 which is based on the work discussed in [NoPa99]. As this figure shows the data is transfered in smaller packets, is accumulated after each bridge and is finally transferred when it becomes complete. Such method is costly since it requires a bit-match circuitry using additional memory elements, and more importantly it influences the working speed in the test mode. We will not peruse the multi-pass strategy in this paper.

3.3 Considering Time in the ILP Formulation

To consider time in the formulation, we need to add one more index (t) to variables x and y reflecting the time index (in clock cycle) in which the access route is established. This is done as follows:

$$x_{i,t} = \begin{cases} 1 & \text{if bridge } i \text{ is used at time step } t \\ 0 & \text{otherwise} \end{cases}$$

$$y_{j,k,t} = \begin{cases} 1 & \text{if path } k \text{ is used to access port } P_j \\ & \text{at time step } t \\ 0 & \text{otherwise} \end{cases}$$

Let t_j^S and t_j^L denote the earliest and the latest time (in clock cycle) that port P_j can be accessed. When a path is established between the SoC's primary input and a core's input port or between a core's output port and the SoC's primary output, only one clock cycle is needed to transfer data assuming the same bitwidth bus/interconnects. Thus, t_j^S and t_j^L correspond to as soon as possible (ASAP) and as late as possible (ALAP) time indices, respectively, in the test access scheduling plan. Let P_{in} and P_{out} denote the set of all SoC input and output ports, respectively. Then these bounds can be estimated as:

$$t_j^S = \begin{cases} 1 & \text{if } P_j \in P_{in} \\ 2 & \text{if } P_j \in P_{out} \end{cases}$$

$$t_j^L = \begin{cases} ||P_{in}|| + ||P_{out}|| - 1 & \text{if } P_j \in P_{in} \\ ||P_{in}|| + ||P_{out}|| & \text{if } P_j \in P_{out} \end{cases}$$

Note that the above bounds are too conservative because we ignored the execution time and the testing order of the cores to create the maximum competition to access ports. Additionally, we assumed that: a) all cores compete with each other at all test time to access the TAM; and b) each output port of a core requires only one input of the same core to receive data before generating a signature. For example if P_j is an output port, the earliest time step (t_j^S) that it may send a test result is 2, leaving time step 1 for at least one of the input ports to receive some test data. On the other hand, the latest time step (t_j^L) that it may send a test result will be $||P_{in}|| + ||P_{out}||$ ($||\ ||$ denotes the set cardinality) assuming that all ports are competing at the same time. In reality, much more information about the core execution time and core input-output behavior are available and thus much tighter bounds can be defined which may significantly reduce the number of constraints. For example, if the execution time of a core in the test mode is too long, then that busy core does not need to use TAM until it finishes its execution.

In addition to selecting bridges and paths, the new formulation should also produce the test schedule to access ports. Specifically, based on the above definitions the access time step of port P_j is:

$$t_j = \sum_{k=1}^{K_j} \sum_{t=t_j^S}^{t_j^L} t \cdot y_{j,k,t}$$

3.4 Objective Functions

Now that time has been incorporated within the formulation we can define the objective function to minimize overall test time, overall bridge cost or a mix of them.

• Test Time Minimization

For time minimization, the objective function is:

$$total\_time = MAX\{t_j\};\quad \forall j:\ P_j \in P_{out}$$

We don't need to enter the access time for the input ports since other constraints make sure that, for a specific core, its output ports are accessed after its input ports. To linearize this function, it can be written as:

Minimize $\quad total\_time$
subject to: $\quad t_j \leq total\_time \quad \forall j:\ P_j \in P_{out}$

• Bridge Cost Minimization

For overhead (bridge cost) minimization, the objective function is: $total\_cost = \sum_{i=1}^{m} C_i \cdot x_i$ where:

$$x_i = \begin{cases} 1 & \text{if bridge } i \text{ is used at least once} \\ 0 & \text{otherwise} \end{cases}$$

To linearize this function, it can be written as:

Minimize $\quad total\_cost = \sum_{i=1}^{m} C_i \cdot x_i$
subject to: $\quad \sum_{j=1}^{n} \sum_{k=1}^{K_j} \sum_{t=t_j^S}^{t_j^L} y_{j,k,t} \leq ||U_i|| \cdot x_i$
for $1 \leq i \leq m$

U_i is the set of all paths (regardless of time step) that use bridge i. Note carefully that if no path that bridge i is on them is selected, x_i is forced to be zero. Otherwise, x_i must be chosen to be 1 to satisfy the constraint.

Having the above constraints in our formulation, we can have a weighted objective function:

$$W_t \cdot total\_time + W_c \cdot total\_cost$$

$(W_t, W_c) = (1, 0)$ and $(W_t, W_c) = (0, 1)$ correspond to "time minimization" and "cost minimization" problems, respectively. Any other value for (W_t, W_c) such that $W_t + W_c = 1$ puts more emphasis on the larger factor.

3.5 Final ILP Formulation

After incorporating bitwidths and time the final ILP formulation will be as follows:

Minimize $W_t \cdot total\_time + W_c \cdot total\_cost$

subject to:

(i) $\sum_{k=1}^{K_j} \sum_{t=t_j^S}^{t_j^L} y_{j,k,t} \geq 1 \quad \text{for } 1 \leq j \leq n$

(ii) $(\sum_{i=1}^{m} a_{i,j,k}) \cdot y_{j,k,t} \leq \sum_{i=1}^{m} (a_{i,j,k} \cdot x_{i,t})$
$\text{for } 1 \leq j \leq n\,; 1 \leq k \leq K_j; t_j^S \leq t \leq t_j^L$

(iii) $\sum_{k=1}^{K_j} \sum_{t=t_j^S}^{t_j^L} t \cdot y_{j,k,t} \leq total\_time$
$\forall j : P_j \in P_{out}$

(iv) $total\_cost = \sum_{i=1}^{m} C_i \cdot x_i$

(v) $\sum_{j=1}^{n} \sum_{k=1}^{K_j} \sum_{t=t_j^S}^{t_j^L} y_{j,k,t} \leq ||U_i|| \cdot x_i$
$\text{for } 1 \leq i \leq m$

(vi) $\sum_{j=1}^{n} \sum_{k=1}^{K_j} y_{j,k,t} \leq 1$
$\text{for } y_{j,k} \in U_{P_j} \cup U_{B_j}$

(vii) $\sum_{k=1}^{K_j} \sum_{t=t_j^S}^{t_j^L} t \cdot y_{j,k,t} -$
$\sum_{k=1}^{K_r} \sum_{t=t_r^S}^{t_r^L} t \cdot y_{j,k,t} \geq d_v$
$\text{for } P_j, P_r \in Core\_v;\ P_j \in P_{in};\ P_r \in P_{out}$

(viii) $x_i, x_{i,t}, y_{j,k,t} \in \{0,1\} \qquad \forall i,j,k,t$
Other variables $\in$ Z

Constraints (i) and (ii) are very similar to the ones in the original formulation, except the variables have time index t. Constraints (iii), (iv) and (v) are those that we extracted to linearize the time and cost terms in the objective function. Constraints set (vi) shows which paths can not be used at the same time. In the BAG graph if two or more paths use the same node (i.e. port or bus) then they conflict and the corresponding paths can not be used to access ports simultaneously. More specifically, U_{B_j} (U_{P_j}) is the set of all paths, regardless of time, that pass through bus B_j (port P_j). Constraints set (vii) links the execution time of a core to the access time for input/output ports of that core. If port P_j and P_r are the input and output ports of $core\_v$ and such information is available then $t_j - t_r \geq d_v$ forms this set of constraints. The conservative value $d_v = 1$ can always be used to reflect the worst case in which all cores compete at all times with the expense of enlarging the size of the ILP formulation. Finally, constraint (viii) expresses that all of x and y variables can be only 0 or 1 and other variables are integers.

3.6 Effect of Technology on TAM

Depending on the technology and architecture used to implement the SoC the bridges and buses in TAM have various delays. In the synchronous SoCs these delays can be defined and traced using index t (representing the clock tick) through the constraints. Specifically, we have not considered bridge or bus delays in the ILP formulation directly. However, we believe incorporating these delays is quite straightforward. For example, if a bypass bridge or a low speed bus requires some time (e.g. few time steps due to its control mechanism) to function, their delays can be considered in constraint (vii) as a part of overall path delay. Or if two high-speed and low speed buses can not be connected through a bridge we simply avoid establishing paths through such bridges in constraints (i) and (ii).

4 Experimental Results

We used the *lp_solve_3.0* package from Eindhoven University of Technology [Berk00]. The running time for all experiments is less than 10 seconds wall clock time on a SUN Ultra 5 workstation. A C program translates the SoC specification to BAG, applies the DFS algorithm and generates the constraints in *lp_solve* format in a few seconds. The results of the running SoC example (see Figure 3) is shown in Table 1 for three choices of (W_t, W_c). For the first choice $(1,0)$, the ILP formulation finds the fastest access schedule (3 time steps) with no consideration of cost. For $(0,1)$, the formulation finds the cheapest TAM with no effort to minimize time. Finally, choice of $(0.5, 0.5)$ implies that both factors are equally important. The complete TAM solution (path and schedule), corresponding to $(W_t, W_c) = (0.5, 0.5)$ is shown in Figure 7. Although in this example the formulation managed to find the minimum-time and minimum-cost solution for $(W_t, W_c) = (0.5, 0.5)$ this may not be always possible due to the existing tradeoffs between the two. In any case, the formulation is quite flexible to explore this tradeoff.

To challenge with a larger SoC, we selected the Intel 8051 microcontroller. Figure 8 shows the block diagram of this microcontroller. Although this is still too small compared to the today's SoC systems it shows the effectiveness and practicality of our approach.

Table 1: Results of time-cost tradeoff for the running SoC example.

(W_t, W_c) Weights	*Additional* Bridge	*Total* Time	*Total* Cost
(1,0)	$\{b_8, b_9, b_{14}, b_{15}, b_{19}\}$	3	32
(0,1)	$\{b_9, b_{15}, b_{20}\}$	4	16
(0.5,0.5)	$\{b_{11}, b_{15}, b_{18}\}$	3	16

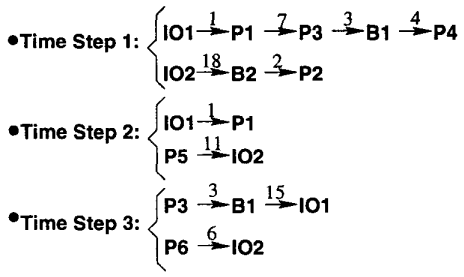

Figure 7: Test schedule of the running example.

Excluding the oscillator circuitry in test mode, the seven cores in 8051 microcontroller have overall 20 input ports (with total bitwidth of 114 bits) and 13 output ports (with total bitwidth of 82 bits). Six I/O terminals (overall bitwidth of 36) will be used in TAM to access cores. The DFS algorithm searches through over 800 paths. To create the highest level of competition among the cores, we did not make any assumption on the order of testing or on the number of patterns required to test each core. Briefly, we assumed that in the test mode all seven cores request to access TAM simultaneously and the formulation has to find the best way to deliver test patterns (to the core input ports) and responses (from the core output ports). The result of the such competition over TAM and the time-cost tradeoffs for testing 8051 are shown in Table 2 for three choices for (W_t, W_c). As expected the choice of $(1, 0)$ is the fastest but the most expensive one and $(0, 1)$ is the cheapest but the slowest test access scheme. Choice of $(0.5, 0.5)$ is located between the two extremes in terms of cost and time.

We also compared our approach with the parallel (multi-chain) scan in terms of cost and test time. Applying a scan-based test to this SoC that uses seven chains (one per core) requires 72 cycles. This is much slower than what our ILP formulation found (i.e. 8 to 19). In terms of cost, the cost of a CMOS tri-state buffer (used as bridges) is almost the same as cost of a CMOS 2×1 MUX (used to change a normal FF to a scanable FF). So, the cost of test overhead for both

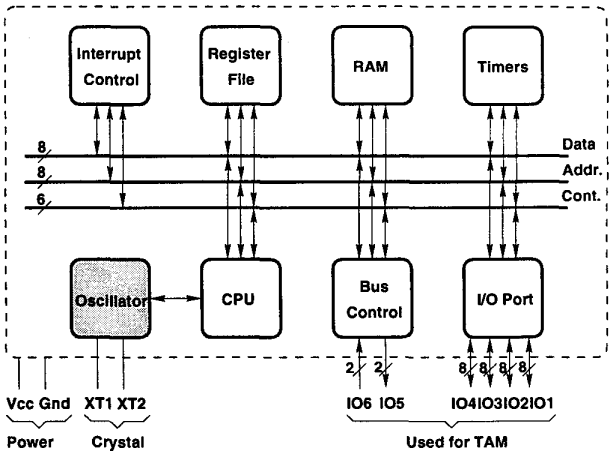

Figure 8: Microcontroller Intel 8051 block diagram.

Table 2: Results of time-cost tradeoff for the MCS 8051.

Our Method			*Scan*	
(W_t, W_c)	*Time*	*Cost*	*Time*	*Cost*
(1,0)	8	44	72	196
(0,1)	19	20		
(0.5,0.5)	10	28		

methods are proportional to the bitwidth of the components (bridge or MUX) as reported in Table 2. Even if we ignore cost of additional wires the multi-scan chain is still much more costly. The reason is that in our method we are reusing most of the existing (*normal-mode*) bridges and add only few (*test-mode*) bridges to optimize TAM while each scan chain uses dedicated scan cells. The cost of the test controller for both approaches are comparable and none of them showed any advantage in this respect.

5 Conclusion

We proposed an ILP formulation to optimize the test access mechanism (TAM) for SoC testing. The formulation is very flexible and bridge overhead or overall access time can be globally minimized. The formulation also produces the exact access schedule to achieve the minimum objective (cost or time). Our empirical results show that coordination between SoC design and test teams provides more flexibility to the formulation in terms of additional (test-mode) bridges. A wide range of tradeoff between cost and time can be performed to finalize the TAM architecture.

REFERENCES

[Aitk95] R. C. Aitken, "Finding Defects with Fault Models," *Proc. of the International Test Conference*, pp. 498–505, October 1995.

[Bako90] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison Wesley, 1990.

[Berk00] M. Berkelaar, *lp_solve, version 3.0*, Eindhoven University of Technology (The Netherlands), www.ics.ele.tue.nl, 2000.

[CaNP99] J. Carletta, M. Nourani and C. Papachristou, "Synthesis of Controllers for Full Testability of Integrated Datapath-Controller Pairs," in Proceedings of the Design Automation and Test in Europe Conference (DATE), (Munich, Germany), pp. 278-282, March 1999.

[Chak00] K. Chakrabarty, "Design of System-on-a-chip Test Access Architectures Using Integer Linear Programming", in *Proceedings of the VLSI Test Symposium (VTS)*, 2000.

[ChPa96] R. Chandramouli, S. Pateras, "Testing Systems on a Chip," *IEEE Spectrum*, Nov. 1996, pp. 42-47.

[GhJD97] J. Ghosh, N. Jha and S. Dey, "A Low Overhead Design for Testability and Test Generation Technique for Core-Based Systems," Proc. of *Intern. Test Conf. (ITC-97)*, pp. 50-59, Oct. 1997.

[ImRa90] V. Immaneni and S. Raman, "Direct Access Test Scheme- Design of Block and Core Cells for Embedded ASICs," in Proc. of *Intern. Test Conf. (ITC-90)*, pp. 488-492, Oct. 1990.

[MABD98] E. Marinissen, R. Arendsen, G. Bos, H. Dingemanse, M. Lousberg and C. Wouters, "A Structured and Scalable Mechanism for Test Access to Embedded Reusable Cores," Proceedings of *Intern. Test Conf. (ITC-98)*, pp. 284-293, 1998.

[MaLo99] E. Marinissen, M. Lousberg, "The Role of Test Protocols in Testing Embedded-Core-Based System ICs," Proceedings of *European Test Workshop. (ETW-99)*, 1999.

[MZKT99] E. Marinissen, Y. Zorian, R. Kapur, T. Taylor and L. Whetsel, "Towards a Standard for Embedded Core Test: An Example," Proceedings of *Intern. Test Conf. (ITC-99)*, pp. 616-627, 1999.

[NoCP97] M. Nourani, J. Carletta and C. Papachristou, "A Scheme for Integrated Controller-Datapath Fault Testing," Proceedings of the 34th Design Automation Conference, pp. 546-551, June 1997.

[NoPa99] M. Nourani and C. Papachristou, "Structural Fault Testing of Embedded Cores Using Pipelining," in the *Journal of Electronic Testing: Theory and Applications (JETTA)*, pp. 129-144, Oct. 1999.

[P1500] P1500 Scalable Architecture Task Force, "Preliminary Outline of the IEEE P1500 Scalable Architecture for Testing Embedded Cores," in Proceedings of *VLSI Test Symposium (VTS-99)*, May 1999.

[PaMN99] C. Papachristou, F. Martin and M. Nourani, "Microprocessor Based Testing for Core-Based System-on-Chip," in Proceedings of the 35th Design Automation Conference, (New Orleans, Louisiana), June 1999.

[RaTy93] J. Rajski and J. Tyszer, "Test Responses Compaction in Accumulators with Rotate Carry Adders," *IEEE Trans. on CAD*, Vol. 12, pp. 531-539, April 1993.

[Rich93] E. Rich, *Artificial Intelligence*, McGraw-Hill Inc., 1993.

[SIA97] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors*, Sematech, Inc. 1997, pp. 24.

[SLCR95] S. Sparmann, D. Luxenburger, K. Cheng and S. Reddy, "Fast Identification of Robust Dependent Path Delay Faults," Proceedings of the 32th Design Automation Conference, pp. 119-125, (San Francisco, California), June 1995.

[ToPo97] N. Touba, B. Pouya, "Testing Embedded Cores Using Partial Isolation Rings," in Proc. of *VLSI Test Symposium (VTS-97)*, pp. 1016, May 1997.

[VaBh98] P. Varma and S. Bhatia, "A Structured Test Reuse Methodology for Core Based System LSIs and a Testing Time Minimization Problem," in Proc. of *Intern. Test Conf. (ITC-98)*, Oct. 1998.

[Whet97] Lee Whetsel, "An IEEE 1149.1 Based Test Access Architecture for ICs With Embedded Cores," in Proc. of *Intern. Test Conf. (ITC-97)*, Oct. 1997.