Test Cost Reduction for the AMD^{TM} Athlon Processor using Test Partitioning[1]

Anuja Sehgal[†] Jeff Fitzgerald[†] Jeff Rearick[‡]

Advanced Micro Devices

{anuja.sehgal, jeff.fitzgerald, jeff.rearick}@amd.com

[†]One AMD Place
Sunnyvale, CA 94088

[‡]2950 E. Harmony Road
Ft. Collins, CO 80528

Abstract

The application of SOC-style test partitioning to a monolithic microprocessor design results in considerable benefits, including simpler and faster ATPG, reduced ECO impact, faster debug, and, most surprisingly, reduced test application time. These results challenge the orthodoxy that flat, top-level ATPG is the best method to produce an optimal pattern set. The granularity of the partitioning was the key factor in achieving the results: a 33-element partition of the AMD^{TM} Athlon CPU chip resulted in better than a ∼80% reduction in test time compared to a flat model of the entire chip. This paper describes the ATPG experiments and quantifies the design overhead required for implementing wrapper cells at partition boundaries.

1 Introduction

Shrinking process technologies and the increase in complexity of microprocessor designs have exacerbated the problem of test cost reduction for microprocessor chips. Many microprocessor chips today contain multiple identical CPU cores, large caches, embedded register files, memory controllers and I/O interfaces. There are enormous test costs associated with these architectures, as well as power constraints for test. Moreover, the product lifetime of these designs is typically long-lived, even across technology nodes, which drives the significant need for efficient test methodologies to reduce test costs.

The challenges in test for microprocessor chips include limited test access, excessively long ATPG run times, high test application times, and power constraints during concurrent test of multiple CPU cores in the chip. Additionally, each design has several variants and each variant typically sees several bug fixes and other design changes, each of which require regeneration of the test patterns close to the tapeout deadline. Also, with a new set of test patterns, additional effort is required on the test floor to qualify good patterns and debug any failing patterns. Modular test is being increasingly used to alleviate test problems in core-based SOCs [1, 2]. Advantages of modular test include reduced ATPG run-time, reduced machine capacity needs, greater test reuse, simplified verification, simplified scan chain failure debug, and reduced test time. It has been shown that modular test can help reduce the overall test cost for core-based SOCs significantly [3]. However, most prior research in modular test has focused almost exclusively on core based SOCs [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]. The advantages of modular test can be extended to non-core-based CPU chips; however, new challenges and tradeoffs are involved in partitioning a non-modular CPU design consisting of process-independent soft IP written in RTL.

In a microprocessor chip, the chip has CPU cores and memory controllers, which are treated as soft in-house cores. Although it is common practice to use modular test at the chip level to test the CPU cores individually [14], the test costs still remain very high since each core itself is a very large functional entity in itself. It is quite normal for each of the cores to be multi-million gate circuits that also offer all the test challenges of limited test access, long ATPG run times, high test application times and power constraints. Moreover, due to area and performance constraints the cores themselves are not physically or functionally partitioned into modules. Thus, it is not very easy to introduce test-modularity into the CPU core itself. Also, it is often very difficult to convince designers that the return on investment of introducing modular test in their design is significant.

In this paper, we show that test costs can be reduced significantly for non-modular designs by appropriately partitioning the design and using a modular test approach. We describe a test partitioning methodology to use modular test for a non-modular CPU core design. We take into account the various area and performance constraints. We show how this approach can alleviate the problem of reducing test power for multi-core microprocessor chips without increasing the test time. We present results for an AMD Athlon microprocessor CPU core to demonstrate the effectiveness of this partitioning approach, which can be applied to other types of designs as well.

The rest of the paper is organized as follows. Section 2 describes the partitioning methodology and a low-effort flow to

[1]AMD, AMD Athlon, and combinations thereof, are trademarks of Advanced Micro Devices, Inc. All other product names are for identification purposes only and may be trademarks of their respective companies.

1-4244-1128-9/07/$25.00

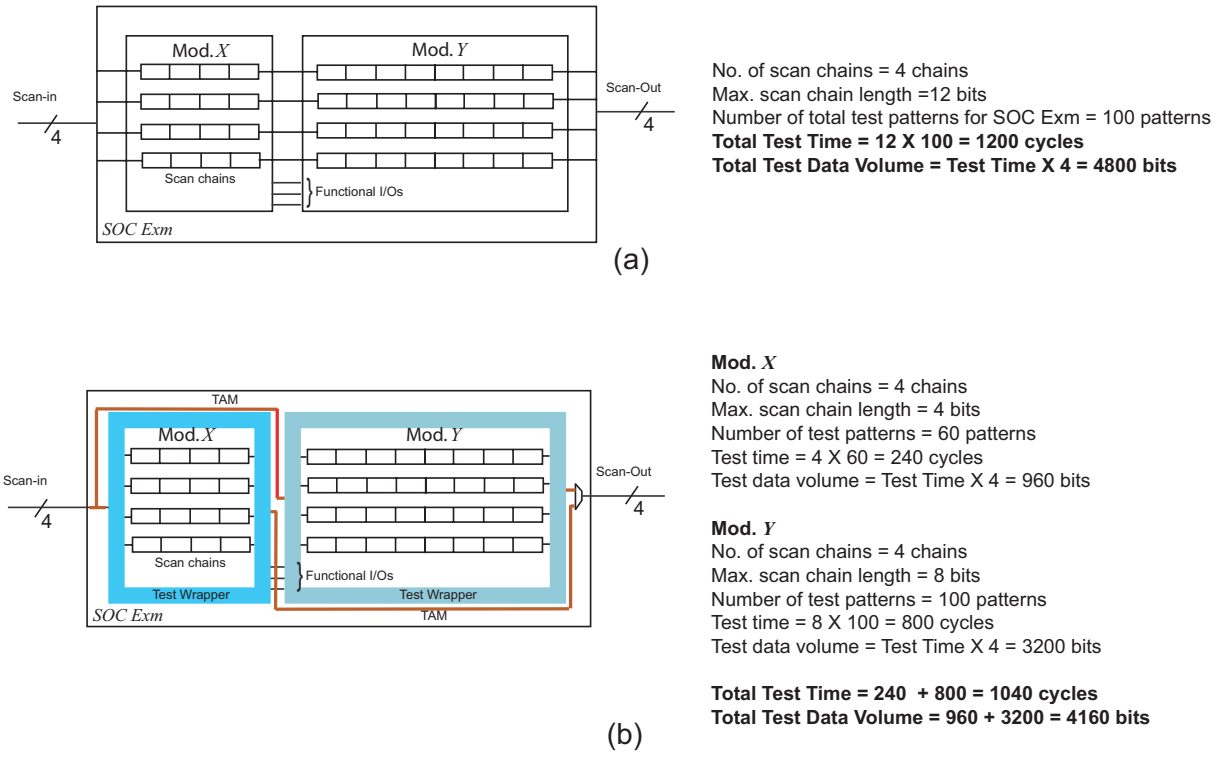


Figure 1. (a) Example of flattened test for design *Exm*; (b) Example of modular test for design *Exm*.

arrive at an effective partitioning. Section 3 presents the results of applying the partitioning method to the AMD Athlon core. Area and power tradeoffs are examined in Section 4. A discussion on hardware compression with the proposed approach is presented in Section 4 as well. Conclusions are drawn in Section 5.

2 Test Partitioning Methodology

Test partitioning decreases test time and test data volume for two reasons. First, scan chains through partitions are shorter than those through the entire chip, so shifting takes less time. Second, the target module is exposed to only those patterns required for it; it is not subjected to the (potentially large) number of fill patterns targeting other circuitry after its desired coverage has been reached. The reasons behind the effectiveness of test partitioning can be best explained by considering an example.

Design *Exm* shown in Figure 1 contains two modules *X* and *Y*. Figure 1(a) illustrates testing design *Exm* as a flattened architecture, while Figure 1(b) illustrates the two modules as separate test partitions, which are tested using a modular approach. The advantages of testing SOC *Exm* as shown in Figure 1(b) can be explained as follows.

Since Module *X* and Module *Y* are distinct modules, they each reach a desired test coverage number with a certain number of patterns. Assume that the number of patterns required to reach 100% coverage for Component *X* is 60 patterns, and the number of patterns required for Component *Y* is 100 patterns. In a flattened test approach the minimum number of patterns required is 100 patterns for the whole architecture since the scan chains are integrated. However, in a modular approach each component has core isolation and can be tested independently with only the required number of patterns, this reduces the overall test time and test data volume as shown in Figure 1.

The process of partitioning a monolithic design into test modules can be broken down into three major steps. The first step in partitioning addresses the disposition of the boundaries of the partitioned modules. This is typically performed by surrounding each test module with a core test wrapper. The IEEE 1500 Standard for Embedded Core Test standardizes a core test wrapper [15, 16]. The wrapper operates in the normal functional mode, inward-facing internal test (”intest”) mode, outward-facing external test (”extest”) mode, and the bypass mode.

Figure 2(a) illustrates an example of a mux-D wrapper boundary cell (WBC). The functional path is from the functional input WFI to the functional output WFO, and WTI and WTO are the test data input and output, respectively. Figure 2(b) shows the interface of two wrapped modules. In Figure 2(c) and (d), the WBC function during internal test and external test modes are shown. The input wrapper cell shifts and applies test stim-

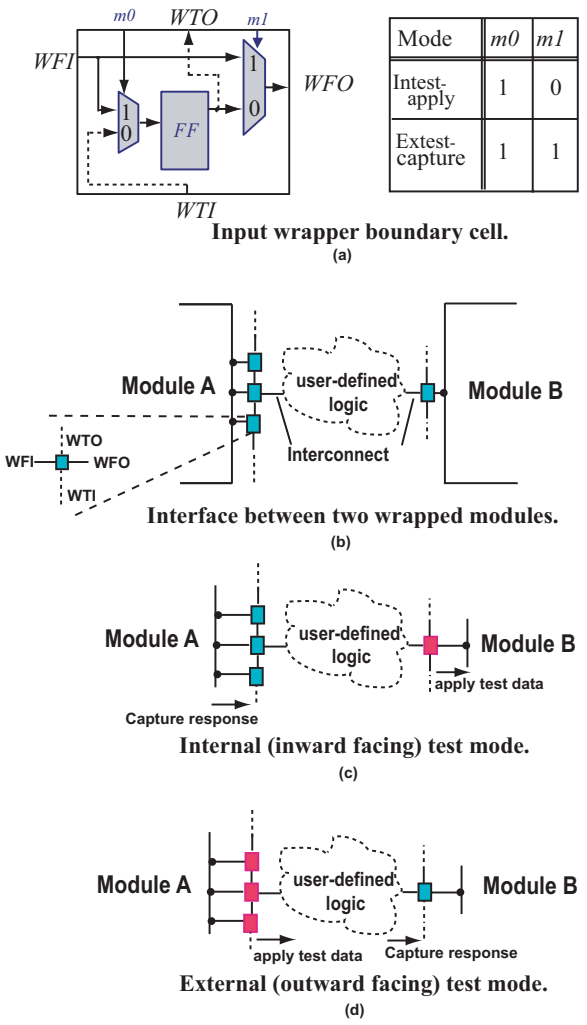


Figure 2. (a) Example of a mux-D input wrapper cell; (b) an example of an interface between two modules; (c) functionality of wrapper cells in internal test mode; (d) functionality of wrapper cell in external test mode.

uli in the internal test mode, and captures and shifts test responses in the external test mode. The operation of the output wrapper cells is the reverse of the input wrapper cells in the two modes.

During external testing, only the wrapper cells are scanned. In order to provide exclusive access to wrapper cells during external test, the wrapper cells are ordered such that they are confined to scan chains that consist of wrapper cells only. However, such a constraint may result in imbalanced scan chains, thus some wrapper cells may share a scan chain with non-wrapper scan cells to achieve better scan chain balancing. The non-wrapper scan cells in such chains can be bypassed.

The wrapper architecture can be customized to support the needs of the design. Most importantly, if the module I/Os are registered, a scan cell can serve as a wrapper cell and thus reduce the amount of test-only circuitry (i.e. a dedicated wrapper cell). It is desirable to register the I/Os of modules, not only from the DfT and ATPG points of view, but also to allow for design reuse and integration using an SOC design style.

The second step in the process of partitioning a monolithic design is to connect the partitioned modules to the test resources available to the entire device; this is known as providing a Test Access Mechanism (TAM). The TAM employed on the AMD Athlon core was simply 40 parallel scan chains in each module, all of which were balanced in length and shifted simultaneously, and whose 80 endpoints were connected to multiplexed chip functional pins. Thus our TAM architecture is similar to the single TAM daisy-chain architecture described in [2]. In general, the design of test access mechanisms must be co-optimized with the design of the test wrappers [17], and often results in complex mappings of a subset of scan chains to a subset of embedded modules. Our analysis of the AMD Athlon core found that there was only a 0.0001% difference in test time between the single TAM daisy-chain architecture that we chose versus a variety of multiple TAM architectures we examined. The use of a single TAM also eliminates the need for test resource partitioning.

It is clear that partitioning can help reduce the test time and subsequently the test data volume [18, 10, 17]; however, in order to partition a non-modular CPU architecture for test, test boundaries have to be appropriately selected. This is the third step in the process of partitioning a monolithic design. To start with, the number of test partitions must be selected, then the actual partition boundaries must be chosen. There are several criteria for partitioning the CPU core for the purpose of test, including 1) maximizing test coverage, 2) minimizing pattern count, 3) using the shortest possible scan chains, 4) minimizing routing overhead, 5) re-using existing scan registers at partition boundaries, and 6) allowing parallel module testing if desired. Solving a problem with so many different objectives can be difficult, particularly in a design environment that is driven very strongly by area and performance constraints, as well as stringent time lines for tapeouts. Also, since the partitions have to be chosen before the final netlists are available it is not possible to find the optimal partition without having the whole design available for analysis. Some initial analysis may be done using data from a previous variant of the chip, but inevitable differences may invalidate the results.

It is thus easier to use the design hierarchy of the core-level netlist to identify test boundaries. Every design has several levels of hierarchy. For example, a functional unit may instantiate smaller modules, which in turn may instantiate smaller circuit blocks, arrays, and so on. Any of these levels of design hierarchy may be used as a test boundary. The level of hierarchy that is used for drawing test boundaries directly impacts the area needed in wrappers and TAM routing, and ultimately determines the test time.

In order to determine the best granularity of partitioning for a design in terms of test time and area, ATPG estimates are

needed prior to test partitioning. The ATPG pattern count and the number of scan flops in each partition determine the test time of the partition. The overall test time of the core is the sum of the test time of all the partitions since we are using a single TAM architecture.

We propose a methodology based on low-effort synthesis to obtain initial ATPG estimates, which guide the test partitioning of the design. This flow is suitable for a design flow that uses synthesizable RTL. The flow of the methodology is described as follows.

1. Determine the number of Automatic Test Equipment (ATE) channels that are available for scan. In our design, the number of scan chains in each module is equal to the number of ATE scan channels.

2. Select a design hierarchy of the design that may be used to partition the design into test modules. A finer granularity of the partitions will most likely result in more wrapper cells, which will have a greater impact on area. Therefore, depending on the area budgets, the test hierarchy should be appropriately chosen.

3. Use low-effort synthesis (LES) to synthesize the RTL and do scan stitching. Low-effort synthesis is a quick process since it does not include optimizations. Note that custom array blocks would have to be black-boxed for synthesis. The purpose of using LES is to get an initial gate level netlist that can give us an initial ATPG estimate. The number of scan chains in the netlist is chosen to be equal to the number of available ATE channels for the CPU core. The scan chain configuration, at this point, is not very relevant since the idea is only to get an estimate of pattern count. The initial ATPG estimate has an error margin of less than 10% compared to the final ATPG [19].

4. Run ATPG on each module to estimate the pattern count.

5. Use the pattern count estimates, the number of I/Os (which represent the number of additional wrapper cells needed), and the number of flops to determine the test time (in terms of the number of shift clock cycles) of the modules and the overall test time of the core with Equation (1)[2]:

$$T(w) = sc(w) \times pt \quad (1)$$

where, pt in the number of patterns, $sc(w)$ is the maximum scan chain length for a module with w scan chains and a total flop count of ff_i; $sc_i(w) = \lceil \frac{ff_i}{w} \rceil$. The total flop count ff_i includes the number of input and output wrapper cells.

6. Repeat steps 2-5 with a different level of design hierarchy. Determine the area and test time of the partitioning as described above. Choose the appropriate test partitioning for the design.

7. Design test wrappers with appropriate wrapper boundary flops. Based on the type of the I/O signal different wrapper flops can be chosen [16].

8. Run a scan stitching script to stitch the flops and wrapper cells. The wrapper cells need to be ordered on chains such that they can be accessed during external test mode [20].

This flow was used to produce quantitative results for the AMD Athlon CPU core, which are presented in the next section.

3 Modular Test for the AMD Athlon Chip

We studied the impact of test partitioning on the AMD Athlon CPU core. We compared three cases: (i) testing the design as a non-modular flattened design, (ii) partitioning the design into 10 top-level modules for test, and (iii) partitioning the design into 33 second-level modules. We compared the three approaches in terms of test time, ATPG run time, and studied their impact on area.

Table 1 presents the information for the core divided into 10 top-level modules. These modules comprise the natural top-level design hierarchy. The information in Table 1 includes the number of scan flops, inputs, and outputs in each module. The number of wrapper cells that must be added to this partition of the design is equal to the number of inputs and outputs of each module. The maximum scan chain lengths are also listed for all the modules. The maximum scan chain length for the flattened approach was 4000.

Table 2 presents the test time and test data volume results for the flattened test approach (in which the CPU core is tested as a monolithic design under test), along with those for the modular test approach (in which the 10 design modules are

TLM	Scan flop count	No. I/Ps	No. O/Ps	Scan chain length	Pattern count (NC)	Pattern count (C)
m_a	19829	181	566	515	30674	4053
m_b	699	371	231	33	5222	799
m_c	22756	58	336	579	32250	5367
m_d	21240	148	159	539	16463	1428
m_e	33729	125	1361	881	52424	11796
m_f	0	900	828	921	73696	5191
m_e	11785	205	397	310	20417	1773
m_f	15466	250	815	414	25389	2667
m_g	18528	73	292	473	11684	2431
m_h	17951	296	510	469	18277	4993

TLM: Top-level module; NC: non-compacted; C: compacted.

Table 1. Module information for top level test partitioning.

[2]Equation (1) does not include the capture clock cycles. Due to the difference in scan and system frequencies, the system capture clock cycles are negligible compared to the shift cycles; also, the number differs based on the type of test applied (stuck-at v/s transition tests).

	Flattened approach (t_1)	Modular test approach (t_2) (10 modules)	Percentage difference $\Delta t^{\dagger}$
Total patterns	8607	40,498	$+370.5\%$
Total flops	161,983	170,085	$+5\%$
Test time (clock cycles)	$34,858 \times 10^3$	$21,742 \times 10^3$	-37.6%

$^{\dagger}\Delta t = \frac{t_2 - t_1}{t_1} \times 100.$

Table 2. Comparison of flattened approach with modular approach with 10 modules and compacted pattern sets for both approaches.

	Flattened approach (t_1)	Modular test approach (t_2) (10 modules)	Percentage difference $\Delta t^{\dagger}$
Total patterns	73,566	286,496	$+289\%$
Total flops	161,983	170,085	$+5\%$
Test time (clock cycles)	$297{,}911 \times 10^3$	$123{,}726 \times 10^3$	-58.4%

$^{\dagger}\Delta t = \frac{t_2 - t_1}{t_1} \times 100.$

Table 3. Comparison of flattened approach with modular approach with 10 modules and non-compacted pattern sets for both approaches).

wrapped and tested independently in a sequential manner). In these experiments, the compaction option for the ATPG tool was active (corresponding the the (C) column in Table 1). The TAM (as described in Figure1) provides each module with scan access via the full width of 40 parallel scan chains. Note that the ATPG compaction option is an algorithmic option in the ATPG engine [21], and does not require on-chip hardware.

The overall test time of the CPU core in the modular approach is the sum of the test time for each individual module. The resulting overall test time results is a reduction of $\sim$38% compared to the non-modular test approach. Also, since test data volume is directly proportional to test time (Test time $\times$ No. of scan channels), the test data volume is also reduced by $\sim$38%. The reduction in test time and volume is significant, although the number of flops increased by $\sim$5% (due to the additional wrapper cells) and the cumulative pattern count of the modules was $\sim$370% higher. This reduction can directly be attributed to significant reduction in the scan chain lengths in each module compared to the length of 4000 in the flattened approach. These results show that it is possible to achieve all the advantages of modular test together with a decrease in test time.

Similarly, Table 3 shows results with compaction inactive in the ATPG tool. The reduction in test time is $\sim$58% for this case. In the case with the compaction active, the reduction is lower because higher compaction can be reached for a larger design compared to smaller modules.

Next, we partition the design into test modules based on the next lower level of hierarchy in the design. We term this as

TLM	SLM	Scan-flop count	No. I/Ps	No. O/Ps	Scan chain length	NC	C
m_a	m_1	8601	271	269	229	13682	2312
	m_2	3747	251	196	105	7830	609
	m_3	3783	118	95	100	3528	765
	m_4	3698	7	37	94	5634	367
m_b	m_5	699	147	112	24	5222	799
m_c	m_6	3683	82	28	95	6719	1138
	m_7	5179	184	177	139	12514	950
	m_8	7965	138	111	206	8594	1270
	m_9	5929	34	29	150	4423	2009
m_d	m_{10}	6783	41	21	172	6783	324
	m_{11}	2144	100	33	57	3792	291
	m_{12}	12313	221	177	318	5180	813
m_e	m_{13}	6808	187	214	181	10392	2039
	m_{14}	1838	321	281	61	2278	673
	m_{15}	6210	374	259	171	16226	2551
	m_{16}	4088	366	478	124	11790	3620
	m_{17}	5706	308	276	158	7931	2551
	m_{18}	5292	265	178	144	3807	457
m_f	m_{19}	0	349	180	14	31921	2399
	m_{20}	0	174	120	8	16422	104
	m_{21}	0	142	120	7	101	961
	m_{22}	0	604	264	9	4438	1664
	m_{23}	0	350	173	14	20814	63
m_e	m_{24}	6077	267	305	167	7861	1211
	m_{25}	5708	242	94	152	7733	562
m_f	m_{26}	6189	392	148	169	4823	456
	m_{27}	4520	269	302	128	12968	1461
	m_{28}	4757	250	365	135	7598	750
m_g	m_{29}	2397	120	214	69	5681	1286
	m_{30}	16131	59	95	408	6003	1145
m_h	m_{31}	7947	289	206	212	6911	997
	m_{32}	7845	220	284	209	7175	1838
	m_{33}	2159	145	20	59	4191	2158

TLM: Top-level module; SLM: Secondary-level module.
NC: non-compacted pattern count; C: compacted pattern count.

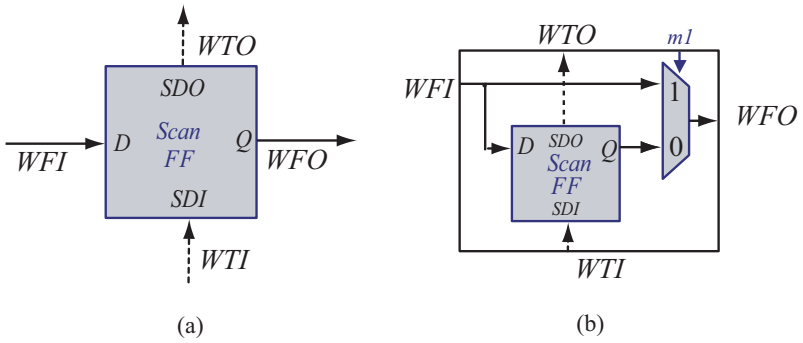
Table 4. Module information for secondary test partitioning.

	Flattened approach (t_1)	Modular approach (t_2) (33 modules)	Percentage difference $\Delta t^{\dagger}$
Total flops	161,983	172,582	$+6.5\%$
Test Time (clock cycles)	Compacted pattern set		
	$34,858 \times 10^3$	$5{,}670 \times 10^3$	-83.7%
	Non-compacted pattern set		
	$297{,}911 \times 10^3$	$34{,}305 \times 10^3$	-88.4%

$^{\dagger}\Delta t = \frac{t_2 - t_1}{t_1} \times 100.$

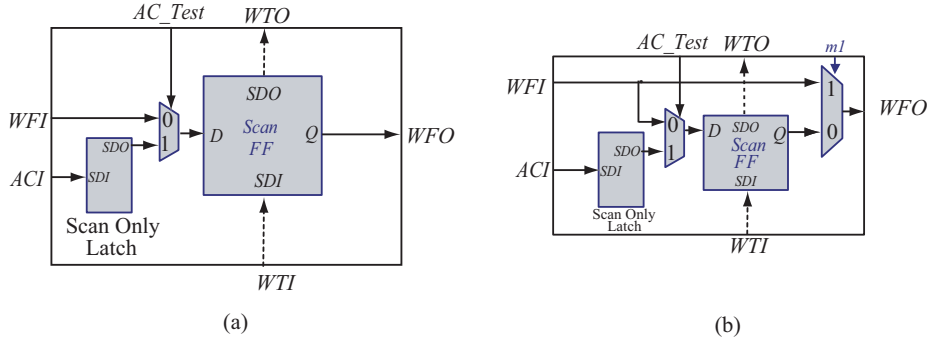
Table 5. Comparison of flattened approach and modular approach with 33 modules for a compacted and non-compacted test set.

secondary test partitioning. Table 4 presents the mapping of the secondary modules to their respective top-level modules. With an increase in the number of test partitions, there is an



Mode	*m1*
Apply stimuli	0
Capture response	1

Figure 3. Examples of wrapper cells for static testing; (a) WBC for I/O with flop at test boundary; (b) WBC for I/O without flop at test boundary; (c) Mux selects for apply and capture cycles.



Mode	*m1*	*AC_Test*
Apply stimuli	0	1
Capture response	1	0

Figure 4. Examples of wrapper cells for AC testing; (a) WBC for I/O with flop at test boundary; (b) WBC for I/O without flop at test boundary; (c) Mux selects for apply and capture cycles.

increase in the number of wrapped inputs and outputs in the secondary modules, but a corresponding decrease in the patterns per partition.

Table 5 presents the test time results for the flattened test and the modular test approach in which the 33 design modules are wrapped and tested independently in a sequential manner. Each module has 40 parallel scan chains. The results show a significant reduction of 84% and 88% for the compacted and non-compacted test set, respectively. There is a 6.5% increase in the number of flops due to the increase in the number of partitions. We analyze the impact on area due to the increased test partitioning and wrapper cells in the next section.

The ATPG time for the modular approaches were almost 5 times faster than the flattened approach. This was despite the use of multiple child processes in the runs for the flattened approach. There was also improvement is test coverage in each module compared to the overall coverage of the flattened approach.

4 Impact on Area, Power, and Hardware Compression

The impact of wrapper-based test partitions on area and performance are one of the biggest issues that need to be considered for a microprocessor chip. The area and additional gate delays for the additional wrapper cells need to be budgeted early in the design phase.

Figures 3 and 4 show a few examples of wrapper cells for static and AC scan test, respectively; the implementation of the scan cell depends on whether the design is mux-D or LSSD. Since we do only static external testing, output wrapper cells need not launch transitions, thus either one of the wrapper cells shown in Figure 3 can be used. The wrapper cells shown in Figure 3 are also suitable for WBCs in the case of static internal testing. Figure 4 shows examples of wrapper cells that can be used for transition testing. The *ACI* input is used to set up the second vector in a two-vector transition test. The *AC_Test* signal is asserted during transition testing, and deasserted during the capture cycle. Table 6 gives the relative area of wrapper cells for static and AC control with respect to the area of a non-scan cell. This area analysis is based on area of standard cell library elements for an undisclosed process technology within AMD. The scan cells have an LSSD architecture.

The performance overhead associated with wrapper cells is due to the additional delays added in the functional path. For static testing, if a non-scan flop exists at the boundary, it will be replaced with a scan flop (Figure 3(a)) as the wrapper. If a flop does not already exist at the boundary, adding a wrapper cell will result in an extra mux-delay (Figure 3(b)). In the case

Type of WBC	Original Cell	Original Cell Area	Wrapper Area
Static control	no cell	-	$2A_{\text{nsff}}$
	non-scan cell	A_{nsff}	$1.5A_{\text{nsff}}$
	scan cell	$1.5A_{\text{nsff}}$	$1.5A_{\text{nsff}}$
AC control	no cell	-	$2.1A_{\text{nsff}}$
	non-scan cell	A_{nsff}	$4.09A_{\text{nsff}}$
	scan cell	$2A_{\text{nsff}}$	$4.09A_{\text{nsff}}$

A_{nsff}: Area of a non-scan flip-flop.

Table 6. Area estimation for wrapper boundary cells (WBCs).

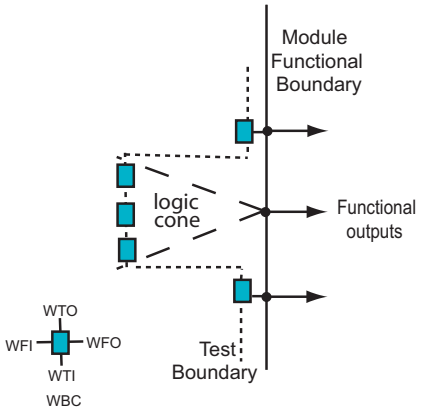


Figure 5. An example of a module with distinct test and functional boundaries.

of AC testing, there is an extra mux delay in both examples shown in Figure 4. Gate delays can be minimized by using flops with an integrated two-input multiplexer. The impact on area and performance due to wrapper cells also depends greatly on the wrapper boundary insertion strategy. There are three main choices for wrapper insertion:

1. Insert a wrapper at the boundary of the hierarchically partitioned module. This involves adding additional scan flops and a mux. Depending on whether the WBC is used for launching transition tests or static tests, the area can vary. Any of the wrapper cells shown in Figures 3 and 4 can be used in this case depending on the area and performance budgets.

2. Using a CAD script, trace the cone of logic from the boundary to the flops internal to the module, and replace the non-scan flops with scan flops. This may result in many more flops on the wrapper scan chain depending upon the cone of logic as illustrated in Figure 5. The logic between the test boundary and functional boundary in this case is tested during external test. Although, the area and performance impact of substituting a non-scan flop with a WBC may be lower, the number of such cells may be much greater; this trade-off needs to be evaluated for such an approach.

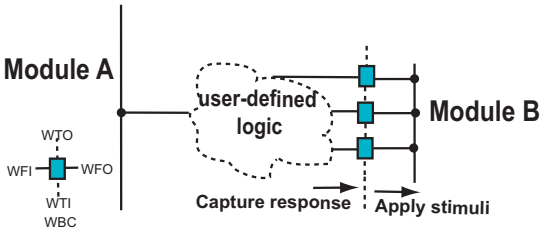


Figure 6. An example of a single layer of wrapper cells shared between two modules.

3. Re-use input wrapper cells of one module as the output wrapper cells of the next module [22] as shown in Figure 6. This is possible if the modules are tested sequentially, as is the case with the AMD Athlon core. Concurrent testing of modules is disallowed by this approach.

All three approaches can be suitable in different parts of the core. Table 7 presents results for the increase in area due to the conversion of scan and non-scan flops into wrapper cells for static and AC control. We compare the results for the case of 10 partitions and 33 partitions. Clearly, the case with 33 partitions results in a greater area-overhead, but this increase in area is accompanied with a large reduction in test time, and shorter ATPG run times, together with ease of debug and pattern reuse. Note that the increase in area refers to the increase in the area relative to the area of total number of scan flops (163711) in the design; it is not the total increase in the area of the core.

Also, if the input wrapper flops of one module serve as the output wrapper flops of a neighboring module there can be a significant reduction in the number of wrapper cells. In such a case, the test of the modules sharing wrapper cells has to be time multiplexed, which is the case in a sequential test schedule. For the case with 10 partitions, the number of internal output wrapper cells (4172) is greater than the number of input wrapper cells (1329). Hence, all the input wrapper cells can be shared as output wrapper cells, reducing the number of output wrapper cells needed to 2843, which is a reduction of 48% in the number of wrapper cells needed. Similarly, for the case with 33 partitions, the number of inputs (7844) is greater than the number of output wrapper cells (6511), hence all internal output wrapper cells can be replaced by shared input wrapper cells, which is a reduction of 45.3% in the number of wrapper cells.

Test Power Reduction

Quad-core microprocessor chips are common today, 8 core and 16 core microprocessor chips are also on the horizon for servers. A growing concern for such designs is the power consumption during concurrent test of the multiple CPU cores in the chips. Overheating of the chip during test can cause irreversible damage. Overheating may also cause patterns to fail and result in a yield loss. Modular test offers significant advantages in terms of reducing power without an increase in test time.

Test of WBC	Non-scan boundary cells	Increase in WBC Area (a_{10}) (10 modules)	$\Delta A_{10}^{\dagger}$ (%)	Increase in WBC Area (a_{33}) (33 modules)	$\Delta A_{33}^{\ddagger}$
Static Control	50%	$4051A_{\text{nsff}}$	1.23%	$6744A_{\text{nsff}}$	2.05%
	25%	$2025A_{\text{nsff}}$	0.61%	$3372A_{\text{nsff}}$	1.02%
	10%	$810A_{\text{nsff}}$	0.24%	$1348A_{\text{nsff}}$	0.41%
AC Control	50%	$20984.18A_{\text{nsff}}$	6.4%	$34933.92A_{\text{nsff}}$	10.6%
	25%	$18958.18A_{\text{nsff}}$	5.7%	$31561.92A_{\text{nsff}}$	9.63%
	10%	$17743.18A_{\text{nsff}}$	5.4%	$29537.92A_{\text{nsff}}$	9.02 %

A_{nsff}: Area of an LSSD non-scan flip-flop.
$^{\dagger}\Delta A_{10} = \frac{a_{10}}{163711 \times 2A} \times 100$. $^{\ddagger}\Delta A_{33} = \frac{a_{33}}{163711 \times 2A} \times 100$.

Table 7. Increase in scan flop area due to conversion of non-scan boundary cells to wrapper boundary cells (WBCs).

In a multicore microprocessor design with several identical CPU cores, the most time efficient test schedule is to test the cores concurrently by broadcasting the test stimuli to all cores and comparing the outputs of the cores[3]. Such a schedule, however, can result in excessive voltage droop. To avoid overheating of the core, it may become necessary to test the cores one at a time, which results in a significant increase in the test time of the chip. This problem can be alleviated by using the test partitioning method and a sequential test schedule. By scheduling the tests of the modules sequentially, all the cores can be tested concurrently in lower amount of time without as much power consumption. The total power consumed by testing a subset of all the cores is lower than testing all the full cores concurrently. We should note, however, that the local power rails in both, the flattened and the module approach, have to be able to support the test power consumption of the individual modules adequately.

In our design, a LSSD scan architecture was used, hence it is possible to use scan clock gating without introducing any delays in the functional clock tree. During test, only the scan clocks of module under test are operational, while there is no shift activity in the remaining modules. Since only one module is tested at any given time, the switching activity in the core is less compared to the case of testing all modules in a flattened test approach at any given time. Thus the average power is lower in the case of a modular test approach. Other power saving methods have been proposed for a modular mux-D architecture as well [23]. These methods include gating the scan enable and scan input to scan chains.

Integration with hardware compression

The use of hardware compression in large integrated circuits is common. Hardware compression can provide significant test time and test data volume reduction at the expense of additional area and routing. However, hardware compression lacks all the advantages of modular test such as reduced ATPG complexity and memory usage, easier test debug, greater test re-use, and simpler test vector verification. The proposed test partitioning approach can be used to enhance hardware compression, such that higher compression ratios can be reached with lower area and routing overhead. Moreover, all the advantages of a modular test approach can be enjoyed.

In a design with test partitioning, hardware compression can be done in a hierarchical manner, wherein each module has its own decompressor and compactor [24]. In doing so, the overall target compression ratio can be reached with lower local compression ratios in each module, thereby reducing the area and routing overhead. One may, however, argue that the care-bit density increases with increased test modularity in a design, which in turn may impact the achievable compression ratios. From our experiments, we found that the test patterns at the modular level are able to reach target compression ratios as high as the target compression ratio of the flattened approach. In one of the variants of our design we found that the flattened approach reached a 9.1X compression ratio for a 10X target compression scan chain partitioning. We simulated hardware compression on a SLM module by re-stitching the scan chains to achieve a compression ratio similar to one achieved for the flattened approach and reached a compression ratio of 9.9X.

Consider the AMD Athlon CPU core, we have shown that a modular test approach with 33 modules results in a 84% reduction in test time and volume compared to a flattened approach. If hardware compression with a compression ratio of 10X is used in the flattened approach, its test time will be 10 times lower. Also, if hardware compression is used in each of the modules, the test time of each of the modules will be 10 times lower. Since the modular approach is 84% better than the flattened approach to begin with, even with the hardware compression it will result in lower test time than the flattened approach. Moreover, the routing congestion is lower in the hierarchical approach since the scan chains have to be routed to a local decompressor and compactor instead of a global decompressor and compactor at the top-level. In [24] and [25], the hierarchical compression approach is described in detail and case studies have been shown for large ASIC SOCs. With test partitioning in the CPU core, the hierarchical compression approach can be easily used in the originally non-modular design.

[3] Details of the broadcast test access mechanism is beyond the scope of this paper.

5 Conclusions

In this paper, we have shown that the advantages of using a modular test approach are not limited to core-based SOCs, instead the advantages and test time reduction can be effectively extended to flattened microprocessor architectures. We have proposed a test partitioning methodology for flattened designs based on low-effort synthesis. We have presented data for the AMD Athlon CPU core, which demonstrates the effectiveness of the approach. A test time reduction of up to 84% can be achieved by appropriately partitioning the core into test modules. We have highlighted that the reduction in test time is accompanied by other advantages that are inherent to a modular test methodology.

We have also studied the impact on area and power. We have quantified the increase in area under different design scenarios for static and AC tests. We have also discussed the advantages of partitioning the design into test modules for test power reduction and easier integration of hardware compression.

Acknowledgment

We thank Ashok Mathur, Jan-Gudmund Ness, and Srinivas Reddy from AMD for the help they provided in collecting the ATPG results.

References

[1] Sandeep Kumar Goel, Kuoshu Chiu, Erik Jan Marinissen, Toan Nguyen, and Steven Oostdijk. Test Infrastructure Design for the Nexperia^{TM} Home Platform PNX8550 System Chip. In *Proceedings Design, Automation, and Test in Europe (DATE) Designer's Forum*, pages 108–113, Paris, February 2004.

[2] Tom Waayers, Richard Morren, and Roberto Grandi. Definition of a robust modular SOC test architecture; resurrection of the single TAM daisy-chain. In *Proceedings IEEE International Test Conference (ITC)*, page Digital Object Identifier 10.1109/TEST.2005.1584022, November 2005.

[3] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing Embedded-Core-Based System Chips. *IEEE Computer*, 32(6):52–60, June 1999.

[4] Joep Aerts and Erik Jan Marinissen. Scan Chain Design for Test Time Reduction in Core-Based ICs. In *Proceedings IEEE International Test Conference (ITC)*, pages 448–457, Washington, DC, October 1998.

[5] Prab Varma and Sandeep Bhatia. A Structured Test Re-Use Methodology for Core-Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 294–302, Washington, DC, October 1998.

[6] Erik Jan Marinissen et al. A Structured And Scalable Mechanism for Test Access to Embedded Reusable Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 284–293, Washington, DC, October 1998.

[7] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Integrated Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs. In *Proceedings ACM/IEEE Design Automation Conference (DAC)*, pages 685–690, New Orleans, LO, June 2002.

[8] Sandeep Koranne and Vikram Iyengar. On the Use of k-tuples for SoC Test Schedule Representation. In *Proceedings IEEE International Test Conference (ITC)*, pages 539–548, Baltimore, MD, October 2002.

[9] Erik Larsson and Zebo Peng. An Integrated Framework for the Design and Optimization of SOC Test Solutions. *Journal of Electronic Testing: Theory and Applications*, 18(4/5):385–400, August 2002.

[10] Sandeep Kumar Goel and Erik Jan Marinissen. Effective and Efficient Test Architecture Design for SOCs. In *Proceedings IEEE International Test Conference (ITC)*, pages 529–538, Baltimore, MD, October 2002.

[11] Sandeep Kumar Goel and Erik Jan Marinissen. Control-Aware Test Architecture Design for Modular SOC Testing. In *Proceedings IEEE European Test Workshop (ETW)*, pages 57–62, Maastricht, The Netherlands, May 2003.

[12] Anuja Sehgal, Sandeep Kumar Goel, Erik Jan Marinissen, and Krishnendu Chakrabarty. IEEE P1500-compliant Test Wrapper Design for Hierarchical Cores. In *Proceedings IEEE International Test Conference (ITC)*, pages 1203–1212, Charlotte, NC, October 2004.

[13] Tom Waayers. An Improved Test Control Architecture and Test Control Expansion for Core-Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 1145–1154, Charlotte, NC, September 2003.

[14] Mack Riley, Louis Bushard, Naoki Chelstrom Kiryu, and Steven Ferguson. Testability Features of the First-Generation CELL Processor. In *Proceedings IEEE International Test Conference (ITC)*, page Digital Object Identifier 10.1109/TEST.2005.1583967, November 2005.

[15] Francisco DaSilva, Yervant Zorian, Lee Whetsel, Karim Arabi, and Rohit Kapur. Overview of the IEEE P1500 Standard. In *Proceedings IEEE International Test Con-*

ference (ITC), pages 988–997, Charlotte, NC, September 2003.

[16] Erik Jan Marinissen et al. On IEEE P1500's Standard for Embedded Core Test. *Journal of Electronic Testing: Theory and Applications*, 18(4/5):365–383, August 2002.

[17] Vikram Iyengar, Krishnendu Chakrabarty, and Erik Jan Marinissen. Efficient Test Access Mechanism Optimization for System-on-Chip. *IEEE Transactions on Computer-Aided Design*, 22(5):635–643, May 2003.

[18] Yervant Zorian, Erik Jan Marinissen, and Sujit Dey. Testing Embedded-Core Based System Chips. In *Proceedings IEEE International Test Conference (ITC)*, pages 130–143, Washington, DC, October 1998.

[19] Ahmet Tokuz. private communication.

[20] Erik Jan Marinissen, Rohit Kapur, and Yervant Zorian. On Using IEEE P1500 SECT for Test Plug-n-Play. In *Proceedings IEEE International Test Conference (ITC)*, pages 770–777, Atlantic City, NJ, October 2000.

[21] Markus Seuring and Krishnendu Chakrabarty. Space Compaction of Test Responses for IP Cores using Orthogonal Transmission Functions. In *Proceedings IEEE VLSI Test Symposium (VTS)*, pages 213–219, Montreal, Canada, April 2000.

[22] Qiang Xu and Nicola Nicolici. Modular and rapid testing of SOCs with unwrapped logic blocks. *IEEE Transactions on VLSI Systems*, 13(11):1275–1285, November 2005.

[23] Ken Butler, Jayashree Saxena, Atul Jain, Tony Fryars, Jack Lewis, and Graham Hetherington. Minimizing power consumption in scan testing: pattern generation and DFT techniques. In *Proceedings IEEE International Test Conference (ITC)*, pages 355–364, 2004.

[24] Jefferey Remmers, Moe Villalba, and Richard Fisette. Hierarchical DFT Methodology - A Case Study. In *Proceedings IEEE International Test Conference (ITC)*, pages 847–856, November 2004.

[25] Jefferey Remmers, Darin Lee, and Richard Fisette. Hierarchical DFT with enhancements for AC scan, test scheduling and on-chip compression - a case study. In *Proceedings IEEE International Test Conference (ITC)*, page Digital Object Identifier 10.1109/TEST.2005.1584034, November 2005.