Low-Cost Sequential ATPG with Clock-Control DFT

Miron Abramovici
Agere Systems - Murray Hill, NJ
miron@agere.com

Xiaoming Yu and Elizabeth M. Rudnick
University of Illinois - Urbana, IL
{xiaoming,liz}@crhc.uiuc.edu

Abstract: We present a new clock-control DFT technique for sequential circuits, based on clock partitioning and selective clock freezing, and we use it to break the global feedback loops and to generate ***clock waves*** to test the resulting sequential circuit with self-loops. Clock waves allow us to significantly reduce the complexity of sequential ATPG. Unlike scan, our non-intrusive DFT technique does not introduce any delay penalty; the generated tests may be applied at speed, have shorter application time, and dissipate less power.

Categories and Subject Descriptors: B.8.1 [Performance and Reliability]: Reliability, Testing, and Fault-Tolerance

General Terms: Algorithms, Design, Reliability

1. Motivation

Automatic test-pattern generation (ATPG) for sequential circuits is an extremely expensive computational process, so that ATPG algorithms working on complex circuits can spend many hours of CPU time and still obtain poor results in terms of fault coverage. Because of the difficulty of the sequential ATPG problem, the electronics industry has given up the idea that complex circuits can be tested without intrusive design for testability (DFT) techniques, such as scan design. However, scan-type DFT introduces delay penalties resulting in performance degradation, and scan tests require long test application times, increase the power consumption, and are difficult to run at-speed. Because many defects create delay faults, at-speed testing has become essential in achieving good defect coverage.

2. Main Contributions

In this paper, we introduce a **new clock-control DFT technique** based on clock partitioning and selective clock freezing. In a sequential circuit, a feedback loop may be *local* or *global*. A local loop includes only one flip-flop (FF) and is also called a *self-loop*; any loop with two or more FFs is called global. A pipeline is a loop-free (acyclic) sequential circuit. First we use clock control to temporarily freeze a subset of FFs to break all global loops; this creates a *near-acyclic circuit* where every FF is either feedback-free or has a self-loop; we will refer to such a circuit as a ***loopy pipe***, since it has a pipeline structure if we ignore the self-loops. Because loopy pipes do not have global feedback, they are structurally simpler than sequential circuits with both local and global feedback. Many partial-scan design methods, starting with [6], scan FFs to break all global feedback, so that the resulting partial-scan circuit is a loopy pipe. However, sequential ATPG for a loopy pipe can still be difficult. In fact, counters, which are notoriously difficult for sequential ATPG, are often implemented by loopy pipes.

A major contribution of the paper is using clock control to generate **clock waves**. A clock wave is a novel clocking scheme that allows a loopy pipe to be tested as a pipeline. We describe **modeling techniques for loopy pipes** tested with clock waves to allow combinational ATPG techniques to be used. We present **a new sequential ATPG algorithm**, called *WAVEXPRESS,* that **detects most faults in a sequential circuit using combinational techniques**, which are at least one order of magnitude faster than sequential ones.

Our DFT technique, called *CLOCKWAVE,* **does not introduce any delay penalty, has small area overhead, and the generated tests are applied at speed.** Compared with scan tests, *WAVEXPRESS* vectors have **shorter test application time** and dissipate **less power**.

The remainder of this paper is organized as follows. Section 3 reviews prior work related to our new approach. Section 4 shows the use of clock freezing to cut global feedback. Section 5 introduces our new testing paradigm. Section 6 presents the *CLOCKWAVE* DFT. Section 7 describes the modeling techniques used in *WAVEXPRESS*. Section 8 discusses the algorithm, while Section 9 shows the results of our preliminary implementation. Section 10 presents conclusions.

3. Related Prior Work

Our approach uses *clock-freezing* [2], which temporarily suspends the sequential behavior of the circuit by freezing its clock, and applies several vectors without changing the current state. In this way, the current state is fully exploited before it is changed. We couple clock freezing with c*lock partitioning* [3][4][8][9][23], a DFT technique that divides the FFs sharing the same clock into several groups, so that in test mode, each group can be independently clocked. Note that clock partitioning implies *selective clock freezing,* as the clocks not activated can be considered temporarily frozen. Clock partitioning increases the testability of the sequential circuit, by reducing dependency among FF values and introducing many more state transitions in the state transition graph, thus making states that are illegal or difficult to reach, easier to reach in test mode. As a result, some faults that were impossible or difficult to detect in the original circuit become easier to test. Clock partitioning has been used for delay-fault testing [10], to cut global feedback [8], to keep global loops in separate

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

partitions [4], and to reduce the detrimental effects of sequential reconvergence on fault detectability [23]. A clock partitioning technique that allows a sequential circuit to be decomposed into a set of overlapping loopy pipes has been used to develop a new at-speed BIST method [26].

An important property of a pipeline is that all or most of its faults can be detected by combinational algorithms [12]. To test a fault in the pipeline of Figure 1a, we apply one input vector, then we clock twice to propagate the input vector throughout the entire circuit. Since the registers serve only to transmit the data, we can make them *transparent* for ATPG, and model the pipeline by a combinational circuit as shown in Figure 1b. Every combinational vector generated on this model is held constant while all registers are clocked; the number of required clocks is equal to the sequential depth of the pipeline. Note that keeping the same vector constant while clocking the circuit a number of times is also very useful for testing general sequential circuits [19].

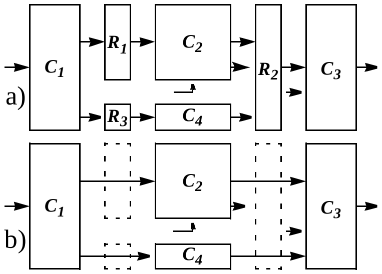

Figure 1. Pipeline and model

4. Breaking Global Feedback

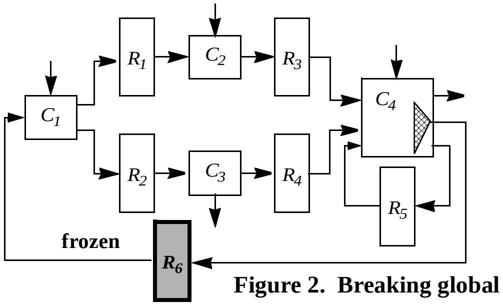

Figure 2. Breaking global loops

In Figure 2 [12], assume that every register R_i has an independent clock (the C_j blocks are combinational). Freezing R_6 is sufficient to cut the two (register-level) global loops of the circuit. The remaining circuit is a loopy pipe because of the self-loop involving R_5. Note that some of the primary inputs (PIs) of the loopy pipe (provided by the outputs of R_6) are frozen, and that a subcircuit of C_4 (indicated by a dashed triangle) remains "invisible" as long as the frozen register is not clocked. The selection of the registers to freeze is heuristic and tries to minimize the invisible regions.

5. The Clock-Wave Testing Paradigm

Figure 3 illustrates the structure of a loopy pipe (the logic driving POs is not shown). For every FF we build its data cone, consisting of all the logic feeding its *D* input. Tracing back to construct a cone, we stop at PIs and FF outputs. Since a loopy pipe has no global feedback, we can levelize the FFs if we ignore the self-loops. We start by assigning level 1 to every FF fed only by PIs; then to every FF fed by FFs whose maximum level is *i* we assign level *i*+1. The highest level *d* is the sequential depth of the loopy pipe. We assume that by clock partitioning, *all FFs at the same level i have an independent clock* $Clock_i$.

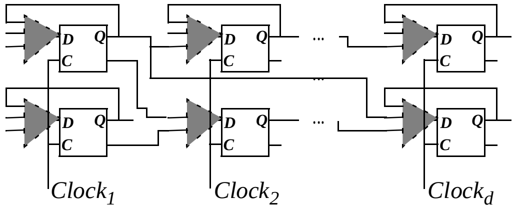

Figure 3. Loopy pipe with partitioned clocks

The basic step in our new paradigm for testing a loopy pipe consists of *keeping a PI vector constant while applying a* ***clock wave***, that is, a sequence that applies a pulse to $Clock_1$, then to $Clock_2$,..., and eventually to $Clock_d$. In this way the applied vector propagates throughout the circuit, with *every FF being clocked only once*. The input vector is kept constant for *d*+1 clock cycles; cycle *d*+1 is needed to allow the changes in the level-*d* FFs to propagate to POs.

6. *CLOCKWAVE* DFT

Figure 4a illustrates the clock control logic for a loopy pipe with *d* levels embedded in a circuit with a frozen register, which is treated as an additional level *d*+1. The role of the additional PIs EG_1 and EG_2 will be explained shortly; now assume that EG_1=EG_2=1. Every FF has a built-in clock-enabling AND gate. FFs at level *i* are enabled by the same EL_i (enable level *i*) signal. In normal mode *N_Mode*=1, so that all EL_i signals are active, and the main clock *MClock* propagates to all FFs. In test mode *N_Mode*=0, and the clock propagation is under the control of *d*+1 enabling signals EN_i, generated by a circular shift register with *d*+1 FFs. The initial

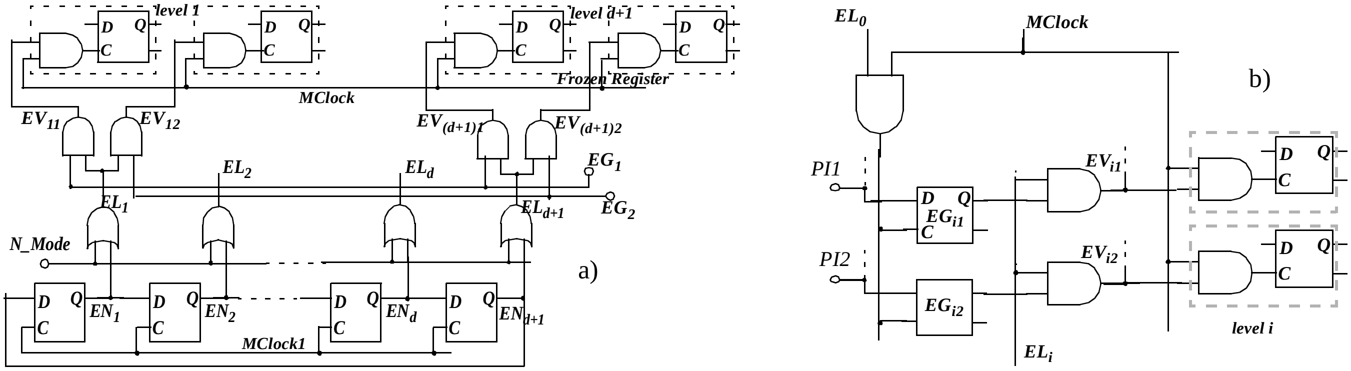

Figure 4. a) Clock control logic b) Storing enabling values

state in the shift register is 10...0, so that one level at a time will be enabled in the proper sequence. The shift register is clocked by a different phase of the main clock (*MClock1*).

The EG_j (enable group *j*) signals allow further clock partitioning of the FFs at the same level *i*. The enable value for group *j* at level *i* is EV_{ij} = *AND*(EL_i,EG_j). The timing of *MClock1* is determined so that the *EV* values will be stable before the arrival of the next *MClock* pulse. In Figure 4a we have two groups at every level (only one FF in each group is shown). While the normal PIs are kept constant for *d*+1 clock cycles, the *EG* signals may change before every clock pulse. They can be shared among all levels, because the levels are clocked one-by-one; hence in every cycle the *EG* signals determine the clock partition for the current level. In normal mode, all *EG* PIs are always set to 1.

The same mechanism is used to hold the state of the frozen register for several clock waves, by setting all the *EG* signals to 0 in the *d*+1 cycle. Note that even when the frozen state is not changed, cycle *d*+1 is used to observe POs.

We could save hardware by using only one AND gating *MClock* for all the FFs in the same group, but this is likely to cause clock skewing. Having clock gating within every FF, the clock-tree distribution path to FFs (routing of *MClock*) is not changed in test mode; hence *DFT cannot cause clock skewing*.

The scheme in Figure 4a requires *G* additional PIs, where *G* is the maximum number of groups needed at any level. For circuits where *G* spare PIs are not available, Figure 4b presents an alternative DFT scheme that needs no additional PIs, but requires additional FFs EG_{ij} to store the enable values for every group *j* at every level *i*. These FFs must be set before the wave starts, so they are treated as an additional level 0, obtained by adding another FF (EN_0) to the circular shift register. Data values for the *EG* FFs are supplied by “normal” PIs; this is possible because *MClock* does not reach any functional FF when level-0 FFs are enabled. This scheme can provide a large number of enabling signals without additional PIs. In normal mode, the *EG* FFs are always set to 1.

Figure 5 shows the timing diagram for a circuit having a frozen register and a loopy pipe with *d*=4, using the second DFT scheme. Each vector is held for cycles 1 to 5. *MClock* pulse 0 loads *EG* FFs, and *MClock* pulse 5 is for the frozen register. The POs are observed after *MClock* pulse 4.

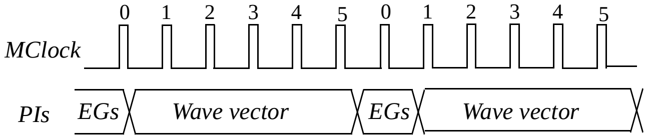

Figure 5. Timing for the second DFT scheme with *d*=4

For a circuit with *n* FFs, the main component of the area overhead is the *n* clock-enabling AND gates, which is less than the area overhead in scan design. In addition, we have *d*+1 FFs and *d*+1 OR gates for the generation of the clock wave. Let G_i be the number of clock groups at level *i*. For the first DFT scheme, we need *G*=max{G_i} PIs, while for the second scheme we need G_i FFs and G_i AND gates at every level *i*. But usually the total number of FFs for clock control is much smaller than *n*, so the total area overhead will still be less than that required for full-scan.

An important feature of *CLOCKWAVE* is that the operating frequency of the circuit is not affected by the clock control mechanism, since *no delays are introduced in data paths*. Unlike scan design, *CLOCKWAVE* is *fully compatible with at-speed testing.* Compared to scan-based tests, our tests have *shorter test application time* and dissipate *less power*, because scan operations are not performed, and only a subset of the FFs are concurrently clocked.

7. Modeling Techniques

Figure 6 shows the combinational model of a self-loop FF. The clock is also treated as a regular PI [1]: *CK*=1 denotes the application of a clock pulse that transfers the *D* input into the new state *Q*, while *CK*=0 means that a clock pulse is not applied, so the FF holds its previous state *QF*, represented by an additional PI. Note that *QF* may also influence *Q* through the data cone, reflecting the effect of the feedback loop.

QF is modeled as a *frozen* PI, so that our ATPG algorithm, *WAVEXPRESS*, will not be allowed to change its value. When we start with *QF=X* denoting an unknown initial state, if *WAVEXPRESS* needs *Q* set to a binary value, this model will force it to set *CK*=1 and to assign other inputs of the data cone to values that allow the FF to be initialized.

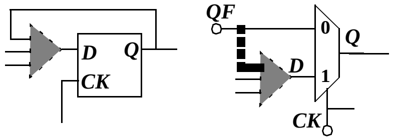

Figure 6. Model of a self-loop FF

The same model is used for all FFs. For frozen FFs we force *CK*=0, and for transparent FFs we set *CK*=1. Then we obtain a combinational circuit, which will be our model for one time-frame (Figure 7). We distinguish between the frozen PIs carrying the previous state of internal FFs (*QFIs*), and those that carry the state of the frozen register (*QFFs*). Clocks are modeled as independent PIs; the figure shows only one clock per level, but, in general, we will have several such clocks at every level. *This combinational model is possible because every internal FF is clocked at most once during a clock wave.* The combinational vector generated in this time-frame is kept constant while the clock wave is applied. Here “one time-frame” corresponds to *d*+1 conventional time-frames.

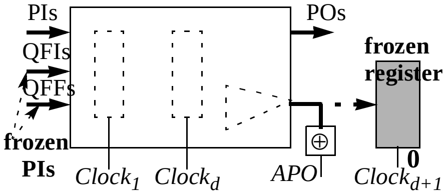

Figure 7. Model for one time-frame

To make the faults from the “invisible” logic also observable within one time-frame (without clocking the frozen register), we add an additional PO (*APO* in Figure 7) that observes all the inputs of the frozen register via an XOR tree. This technique [11][24], used to improve observability, is independent of clock control, has negligible impact on timing, and is compatible with at-speed testing.

8. The *WAVEXPRESS* Algorithm

WAVEXPRESS is an *opportunistic algorithm* that tries to detect all detectable faults with minimal computational effort. Unlike most conventional ATPG algorithms, which work on a selected target until it is detected or proven untestable, *WAVEXPRESS* abandons the current target if it cannot detect it quickly, and moves to other targets that may be easier to detect in the current state. Figure 8 depicts the basic flow of our algorithm, which starts with a single time-frame (Figure 7), and tries to detect as many faults as possible without changing the current frozen state. Before targeting faults, *WAVEXPRESS* analyzes the set of targets and eliminates many faults whose detection is not possible given the currently frozen PI values. After all potentially detectable faults have been targeted within a single time-frame, *WAVEXPRESS* switches to a model with *k* time-frames, starting with *k*=2. After the first fault is detected (by a sequence of *k* vectors), *WAVEXPRESS* reverts to the one-time-frame loop with a new state of the frozen register. The number of time-frames *k* is increased only when not even one fault could be detected with *k* time-frames.

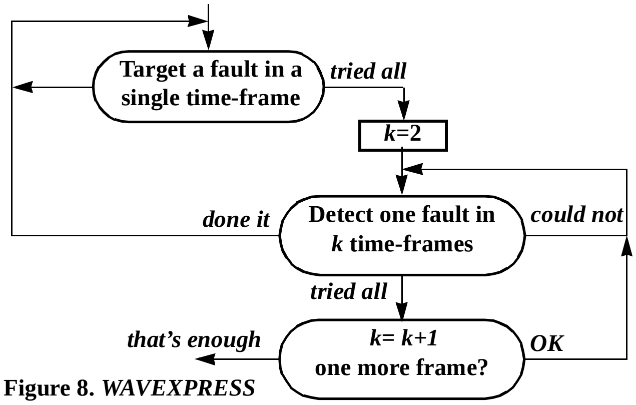

Figure 8. ***WAVEXPRESS***

All assignable PIs, including the non-frozen clocks, start with an unknown logic value. If *WAVEXPRESS* detects the selected target fault, every generated vector is expanded into a sequence of *d*+1 identical vectors and their associated clock wave. The values of the clock PIs are mapped into settings for the corresponding enabling values. Clocks left unassigned are not activated, to save power and (possibly) to preserve fault effects stored in internal FFs. Any unassigned PIs are set to random binary values. The resulting sequence is fault simulated using the PROOFS fault simulator [21]. Note that in the test generation model, the POs are observed only after the *d*-level FFs have been clocked, but in the fault simulation model they are observed after every clock. This is equivalent to applying several random vectors in between the generated vectors, and these vectors may detect additional faults. The results of PROOFS are ported back into *WAVEXPRESS* to enable fault dropping and reporting fault effects stored in FFs. The two programs communicate via *sockets*.

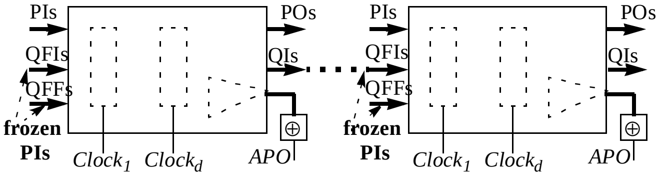

Figure 9. Sequence of separated time-frames

Figure 9 illustrates the operation of the first loop of *WAVEXPRESS*, showing two consecutive time-frames. After a vector is generated in the first one, the changed values of internal FFs are propagated to the *QFI* inputs of the next time-frame before a new fault is targeted; these values are frozen for ATPG. Except for this value transfer (denoted by the dotted line in the figure), consecutive time-frames are not connected, so *WAVEXPRESS* works separately with each time-frame. The values of frozen register *QFFs* are the same in every time-frame.

Figure 10 depicts a model with two time-frames, used after all targets have been tried with one time-frame. The previously frozen register is now allowed to change, and its clock ($Clock_{d+1}$) is now an assignable PI in the first time-frame. No inputs are frozen in the second time-frame, as every *QF* input is connected with its corresponding *Q* output from the first time-frame. All POs in the first time-frame are ignored, since none of the remaining targets could be detected at these POs. Similarly, in a model with *k* time-frames, all POs in the first *k*-1 time-frames are ignored. For *m* PIs, the size of the search space for *k* separated time-frames is $k2^m$, while for *k* connected time-frames is 2^{km}.

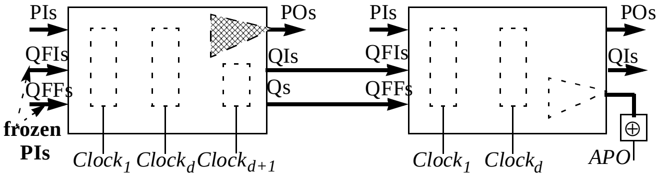

Figure 10. Sequence of connected time-frames

Like FASTEST [15], *WAVEXPRESS* backtraces objectives to PIs, but its logic value system avoids backtracing toward the PIs with frozen values. Backtracing can span multiple time-frames. Since decisions are done only as PI assignments, *WAVEXPRESS does not do explicit state justification, and it never has to justify illegal states*.

Selecting target faults: As preprocessing steps, we use: (1) the algorithms FIRES [14] and FUNI [17], running on the circuit with partitioned clocks, to identify combinationally and sequentially untestable faults; (2) the combinational test generator ATOM [13], running on a full-scan model of the circuit, to identify the rest of the combinationally untestable faults. To reduce the ATPG run-time, we remove the identified untestable faults from the set of target faults.

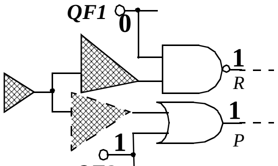

Figure 11.

Since the frozen PI values may preclude the activation or the observation of many still undetected faults, *WAVEXPRESS* removes these faults from the current set of target faults, so that only potentially detectable faults will be targeted. For example, in the circuit in Figure 11, *R* s-a-1 and *P* s-a-1 cannot be activated, and no fault in the shaded areas can be observed. Faults whose effects are stored in currently observable frozen or internal FFs are always included in the set of targets. The set of targets is incrementally updated after every clock wave.

WAVEXPRESS relies on additional heuristics based on testability measures to select a target most likely to be detected in the current state. We also enforce a limit on the number of times the same fault is going to be targeted.

9. Results

We compare a preliminary implementation of *WAVEXPRESS* with (1) HITEC - a deterministic test generator [20], (2) GATEST - a test generator based on genetic algorithms [25], (3) a commercially available sequential ATPG (S_ATPG), and (4) ATOM [13]. S_ATPG was run on a SPARC SUNW Ultra-Enterprise computer; the other programs were run on a Pentium III 700 computer. The two machines have comparable performance.

Table 1: Circuits data

Circuit	PIs	POs	FFs	SLs	Faults	Untst.	Frz.	*D*	*G*	*EG*
s208	11	2	8	8	215	0	0	8	1	8
s420	19	2	16	16	430	0	0	16	1	16
s838	35	2	32	32	857	0	0	32	1	32
ctr_12	2	1	12	12	636	13	0	12	1	12
ctr_16	2	1	16	16	940	13	0	16	1	16
div16	33	34	50	48	2147	176	2	19	3	35
pcont2	9	8	24	0	11272	3846	16	3	16	24
piir8o	9	8	56	0	19936	4648	8	5	8	8
write_s	15	13	18	16	446	3	2	11	2	11
tx_s	24	24	23	23	772	27	3	12	3	13
alpha_s	26	25	16	13	759	46	3	10	3	11

Table 1 provides the data for the circuits we have experimented with. The first three are ISCAS89 benchmarks; *ctr_12* and *ctr_16* are 12-bit and 16-bit up-down counters with a single PO off the most significant bit. *piir8o* is an optimized finite-impulse response filter [7], *div16* is a 16-bit divider [7], and *pcont2* is a controller circuit used in DSP applications [7]. *write_s*, *tx_s*, and *alpha_s* are telecommunication circuits from [5]. *SLs* is the number of self-loops, *Untst* is the number of untestable faults identified in preprocessing, *Frz* is the number of temporarily frozen FFs, *D* is the sequential depth of the circuit (*D*=*d*+1 if a register is frozen, otherwise *D*=*d*), *G* is the number of additional PIs needed in the first DFT scheme, and *EG* is the number of enable FFs used in the second scheme. The last two parameters were computed to create maximum clock partitioning, where each FF has a separate clock, but this is not needed in practice. Although the first five circuits have no global feedback, they are still highly sequential because the high values of *D*.

Table 2 summarizes the ATPG results. Note that ATOM uses full-scan, *WAVEXPRESS* uses *CLOCKWAVE* DFT, and the other programs handle the original circuit without DFT. S_ATPG was run only on four circuits, and GATEST could not run the counters. All run-times are in seconds. *FC* is the detectable fault coverage, computed after the subtracting *Untst* from the total fault count. *Vec* is the number of vectors. *Tst* is the number of tests generated by *WAVEXPRESS*, where each test corresponds to *D* constant vectors. *Conv* stands for "converted," being the number of faults found untestable by HITEC or GATEST, that were detected by *WAVEXPRESS*. *Msd* is the number of faults that were detected by HITEC or GATEST, but were missed by *WAVEXPRESS*.

Although *CLOCKWAVE* is a non-intrusive DFT, *WAVEXPRESS* obtains much higher fault coverage than the other sequential ATPG programs, and is much faster, often by one or two orders of magnitude. For most circuits, the *WAVEXPRESS* stuck-fault coverage was above 90%. In practice, such a test applied at-speed usually achieves higher defect coverage than a test with higher stuck-fault coverage that cannot run at-speed [18]. As expected, full-scan leads to the highest stuck-fault coverage and the fastest run-time for most circuits. However, there are additional costs for scan: area overhead, test application time, delay penalty, and power dissipation. For the largest circuits (*pcont2* and *piir8o*), the area overhead for scan is about twice the area overhead for *CLOCKWAVE*. For *pcont2*, the test application time for the *WAVEXPRESS* vectors is 300 clock cycles, while the full-scan vectors need 2925 cycles (assuming 2 scan chains). For *piir8o*, the corresponding numbers are 850 and 5115

Table 2: ATPG Results

Circuit	WAVEXPRESS					HITEC			S_ATPG			GATEST			ATOM		
	Time	FC%	Conv.	Msd.	Tst.	Time	FC%	Vec.	Time	FC%	Vec.	Time	FC%	Vec.	Time	FC%	Vec.
s208	1	91.6	78	0	63	1	63.7	147				4	62.3	100	0	100	65
s420	7	90.9	251	3	106	16	41.6	152				14	41.4	113	0	100	108
s838	26	91.2	530	2	227	4411	29.6	161				38	29.3	103	0	100	190
ctr_12	21	96.1	63	4	184	4866	86.7	4865	1403	65.0	1167	-	-	-	0	100	99
ctr_16	22	96.0	173	3	268	9865	77.7	7479	5200	56.0	905	-	-	-	0	100	90
div16	107	90.8	202	94	98	968	85.1	289				155	86.3	425	0.3	100	203
pcont2	169	98.9	4056	8	100	8930	44.3	7	7867	65.6	20	544	91.7	143	38	96.3	225
piir8o	1049	98.6	5135	11	170	11053	65.1	20	13948	54.2	17	2376	98.5	506	27	98.4	341
write_s	81	93.2	8	20	89	370	94.5	5664				20	60.0	180	0	100	93
tx_s	84	89.9	30	4	122	5794	60.8	2305				38	48.5	177	0	100	109
alpha_s	197	62.7	23	23	43	6522	28.0	1412				22	14.0	38	0	100	136

(assuming 4 scan chains). Five circuits have zero or negligible delay penalty, but for the other six the performance degradation caused by scan ranges from 2.5% to 14.3%. Compared to normal operation of the circuit, the average power dissipation during the application of *WAVEXPRESS* vectors is about *D* times smaller, because the clock waves reduce the average activity by a factor of *D* (assuming the logic is uniformly distributed among levels). Since usually the power dissipation during scan increases relative to the normal operation, it follows that the power dissipation during scan is at least *D* times larger than the power dissipated by the *WAVEXPRESS* vectors.

In most circuits, the number of faults missed by *WAVEXPRESS* is low. Since in practice, most faults not detected by a conventional sequential ATPG are untestable, we can conclude that *WAVEXPRESS* usually detects most of the sequentially detectable faults. However, *WAVEXPRESS* cannot determine whether the undetected faults are untestable.

For all but three circuits, *WAVEXPRESS* could detect every fault with no more than four time-frames. Five time-frames were needed only for several faults in *div16*, *tx_s*, and *alpha_s*. In most circuits, most faults are detected with one or two time-frames. This is remarkable since all the circuits are deeply sequential.

In the current *WAVEXPRESS* implementation, we did not use any of the advanced sequential ATPG techniques such as [16]; we expect such techniques to further increase the performance of our program.

10. Conclusions

Our *CLOCKWAVE* DFT technique significantly reduces the complexity of sequential ATPG, allowing most faults in deeply sequential circuits to be detected by a low-cost algorithm (*WAVEXPRESS*). The main factor contributing to the reduction in complexity is handling of *D* conventional time-frames as a single time-frame; for example, *D* is 32 for *s838* and 19 for *div16*. *CLOCKWAVE* is a non-intrusive technique that introduces no performance degradation or clock skews, and its area overhead is smaller than that needed for scan DFT. *WAVEXPRESS* vectors may be applied at-speed, and therefore can be used for delay-fault testing. Compared to scan-based tests, *WAVEXPRESS* vectors have shorter test application time and dissipate about *D* times less power.

For some circuits, additional non-scan DFT techniques, such as partial reset [22], may be used to complement *CLOCKWAVE*. *CLOCKWAVE* is also applicable to multiple clock domains, by separately partitioning each clock.

References

[1] M. Abramovici, J. J. Kulikowski, P. R. Menon, and D. T. Miller, "SMART and FAST: Test Generation for VLSI Scan-Design Circuits," *IEEE Design & Test of Computers*, August 1986.

[2] M. Abramovici, K. Rajan, and D. T. Miller, "FREEZE!: A New Approach for Testing Sequential Circuits," *Proc. 29th Design Automation Conf.*, June 1992.

[3] V. D. Agrawal, S. C. Seth, and J. S. Deogun, "Design for Testability and Test Generation with Two Clocks," *Proc. 4th Intn'l. Symp. on VLSI Design*, pp. 112-117, January 1991.

[4] S. Baeg and W.A. Rogers, "A Cost-Effective Design for Testability: Clock Line Control and Test Generation Using Selective Clocking," *IEEE Trans. on CAD*, vol. 18, no. 6, pp. 850-861, June 1999.

[5] F. Fummi, M. Boschini, X. Yu, and E. M. Rudnick, "Sequential Circuit Test Generation Using a Symbolic/Genetic Hybrid Approach," *Journal of Electronic Testing*, vol. 17, pp. 321-330, June 2001.

[6] K-T. Cheng and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. on Computers*, vol. 39, pp. 544-547, April 1990.

[7] V. Chickermane, J. Lee, and J.H. Patel, "A Comparative Study of Design for Testability Methods Using High-level and Gate-Level Descriptions," *Proc. Intn'l. Conf. on CAD*, pp. 620-624, November 1992.

[8] K. L. Einspahr, S. C. Seth, and V. D. Agrawal, "Clock Partitioning for Testability," *Proc. 3rd IEEE Great Lakes Symp. on VLSI*, pp.42-46, March 1993.

[9] K. L. Einspahr, S. C. Seth, and V. D. Agrawal, "Improving Circuit Testability by Clock Control," *Proc. 6th IEEE Great Lakes Symposium on VLSI*, pp. 288-293, March 1996.

[10] W-C. Fang and S.K. Gupta, "Clock Grouping: A Low Cost DFT Methodology for Delay Testing," *Proc. 31st Design Automation Conf.*, pp. 94-99, 1994.

[11] J. R. Fox, "Test-Point Condensation in the Diagnosis of Digital Circuits," *Proc. of IEE*, vol. 124, no. 2, pp. 89-94, February 1977.

[12] R. Gupta, R. Gupta, and M. A. Breuer, "The Ballast Methodology for Structured Partial Scan Design," *IEEE Trans. on Computers*, vol. 39, no. 4, pp. 538-544, April 1990.

[13] I. Hamzaoglu and J. H. Patel, "New Techniques for Deterministic Test Pattern Generation," *Proc. VLSI Test Symp*, pp. 446-452, April 1998.

[14] M. Iyer, D.E Long, and M. Abramovici, "Identifying Sequential Redundancies Without Search," *Proc. 33rd Design Automation Conf.*, June 1996.

[15] T.P. Kelsey, K.K. Saluja, and S.Y. Lee, "An Efficient Algorithm for Sequential Circuit Test Generation," *IEEE Trans. on Computers*, vol. 42, no 11, pp. 1361-1371, November 1993.

[16] X. Lin, I. Pomeranz, and S. M. Reddy, "Techniques for Improving the Efficiency of Sequential Circuit Test Generation," *Proc. Intn'l. Conf. on CAD*, November 1999.

[17] D.E Long, M. Iyer, and M. Abramovici, "FILL & FUNI: Algorithms To Identify Illegal States and Sequentially Untestable Faults," *ACM Trans. on Design Automation of Electronic Systems*, July 2000.

[18] P. Maxwell, R. Aitken, V. Johansen, and I. Chiang, "The Effect of Different Test Sets on Quality Level Prediction: When is 80% Better than 90%?," *Proc. Intn'l. Test Conf.*, pp. 358-364, 1991.

[19] L. Nachman, K. K. Saluja, S. Upadhyaya, and R. Reuse, "A Novel Approach to Random Pattern Testing of Sequential Circuits," *IEEE Trans. on Computers*, vol. C-47, no.1, pp. 129-134, January 1998.

[20] T. M. Niermann and J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits," *Proc. European Conf. on Design Automation*, pp. 214-218, 1991.

[21] T. M. Niermann, W. -T. Cheng, and J. H. Patel, "PROOFS: A Fast, Memory-Efficient Sequential Circuit Fault Simulator," *IEEE Trans. CAD*, vol. 11, no. 2, pp. 198-207, February 1992.

[22] P. Parikh, M. Abramovici, B. Mathew, and D. Saab, "On Selecting Flip-Flops for Partial Reset," *Proc. Intn'l. Test Conf.*, October, 1993

[23] K. B. Rajan, D. E. Long, and M. Abramovici, "Increasing Testability by Clock Transformation (Getting Rid of Those Darn States)," *Proc. VLSI Test Symp.*, April 1996.

[24] E. M. Rudnick, V. Chickermane, and J. H. Patel, "An Observability Enhancement Approach for Improved Testability and At-Speed Test," *IEEE. Trans. CAD*, vol. 13, no. 8, pp. 1051-1056, August 1994.

[25] E. M. Rudnick, J.H. Patel, G.S. Greenstein, and T.M. Niermann, "A Genetic Algorithm Framework for Test Generation," *IEEE Trans. CAD*, vol. 16, no. 9, pp. 1034-1044, Sept. 1997.

[26] J. Shin, X. Yu, E. M. Rudnick, and M. Abramovici, "At-Speed Logic BIST Using a Frozen Clock Testing Strategy," *Proc. Intn'l. Test Conf.*, pp. 64-71, October 2001.