Distributed Control in Testable High-Level Synthesis with Low Area and Power Overhead

Haidar Harmanani and Wissam Marrouche
Department of Computer Science
Lebanese American University
Byblos, 1401 2010, Lebanon

***Abstract*—This work presents a method for distributed test control in coordination with testable data path allocation in high-level synthesis. The method aims at creating distributed test controllers for synthesized data paths with the aim of minimizing area and power consumption. The distributed controllers are designed to be small and mounted next to their corresponding test kernels. Thus, every test kernel in the datapath has its own test controller in addition to a relatively small controller that synchronizes the distributed controllers. The system has been implemented using C++ and favorable results are reported.**

I. Introduction

The design and test of electronic circuits can be tackled at various levels of abstractions. It has been generally recognized that test incorporation earlier in the design cycle leads to an efficient exploration of the design space. High-level synthesis is the design and implementation of a digital circuit from a behavioral description. There are two main tasks in HLS: *operations scheduling* and *resources allocation*. There has been a lot of work on synthesis for test systems. Most of the work focused on optimizing the test time and overhead. However, a major concern when executing BIST in parallel on modules under test is power and noise dissipation [10]. A common way to reduce the circuit activity in synchronous systems is to stop the local clock of inactive subsystems. The local clock is gated with an activation signal [6].

Several techniques have been proposed for the reduction of power consumption in datapath synthesis [1] as well as for synthesis for test and controller optimization. For example, Crews et al. [8] and Hertwig et al. [9] discuss techniques for high-performance controller synthesis. Benini et al. [7] describe transformation for incompletely specified state machines in order to automatically synthesize low power finite state machines with gated clocks while Favalli et al. [6] methodologies that guarantee testability for gated-clock FSM. In [5], Nourani et al. describe the design of an interacting datapaths and controller. For self-testable designs based on BIST, research involving controllers focuses on test plan and test scheduling [2].

A. Problem Description

The problem we address in this paper is described as follows:

> Given a testable data path description in VHDL that was synthesized using a high-level synthesis with test system and a test schedule, generate a distributed test controller that meets the following constraints: 1) exercise the behavior during normal mode and test the circuit during test mode; 2) minimize the power consumption during normal and test execution.

The above problem is solved using a hierarchical controller approach where every test kernel in the datapath has its own test controller in addition to a relatively small controller that synchronizes the distributed controllers. This work is motivated by the following:

1) Distributed controllers are small and mounted next to their corresponding kernels. The bulk of the controller is distributed throughout the data path. The power dissipation is also distributed among the different sub-controllers.
2) By distributing the control, useless power dissipation can be eliminated by selectively stopping the clock portions of the controllers that are inactive at the specific clock cycle.

This paper is organized as follows. Section II discusses our synthesis for test system while section III described the controller synthesis approach in addition to the power minimization approach in our system. We conclude with results in section IV.

II. Testable Data Flow Synthesis

A datapath is represented at the behavioral level by a data flow graph (DFG) where nodes corresponds to 1) an operator that must be assigned to a functional unit during the control step in which it is scheduled; 2) a value that must be assigned to a register for the duration of its life time. The synthesis task is to map a DFG onto structure, that is onto a network of arithmetic logic units, multiplexers, registers and busses.

In this work we use a datapath allocation model based on the notion of *structural testability* [3]. The key element of the structural testability model is the *Testable Functional Block* (TFBs), shown in Figure 1(a), which are test kernels that do not have functional self-loops. In this model, behavioral operations are mapped onto the ALU of the TFB, while the behavioral variables are mapped to the register at the TFB output. The testable allocation algorithm is an iterative refinement procedure. The method constructs an initial datapath structure (IDP) that corresponds to an initial design point through the

0-7803-8322-2/04/$20.00 ©2004 IEEE.

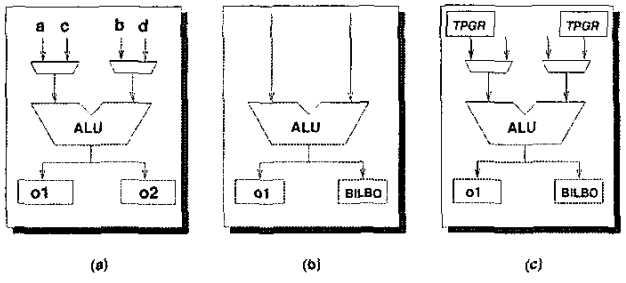

Fig. 1. a) Testable Functional Block, b) Kernel formed with ALU and one Test Register and (b) Kernel formed with registers are assigned to test kernel based on their function

mapping of DFG nodes onto individual TFBs. The initial design cost is incrementally improved through the merging of test kernels guided by the cost difference of the current and the intended data path configuration [3], [4]. Test points are selected concurrently with the synthesis process resulting with two solutions that present a good tradeoff between test length and test quality. The first is based on a BILBO style while in the second modules functionality is used to remove *unnecessary* test points.

Once the datapath has been allocated and the test points selected, it is important to organize tests for the datapath modules into test sessions that bring together the tests of compatible modules. Individual tests are conflicting because 1) they share common test resources such as a test bus or a test response compactors; 2) the power consumption during simultaneous testing exceeds the device power allowance. Power consumption is an important factor that may impact test parallelism. Test power dissipation is a function of time and depends on the switching activity resulting from the application of a new test vector to the system.

III. Controller Synthesis

From the data sequencing and the schedule, the synthesis system must derive the controller specification. We describe the controller structurally as a state machine that specifies the control steps in which various operations in the data flow graph are done using random logic. The controller controls the datapath execution by latching the necessary control signals that govern which path is selected through each multiplexer and which registers are loaded at each control step. Both controllers, the *central* as well as the *distributed*, will be optimized by the target synthesis tool.

A. Centralized Controller

In order to be able to assess the improvement in area, power consumption and speed in the distributed control scheme, a design scheme based on a centralized controller has also been implemented. The centralized controller will be used mainly as a comparison criterion with the distributed controller approach.

The controller is modeled using a Mealy Finite State Machine. Each state in the FSM corresponds to one clock cycle and specifies the next state in addition to a set of control signals. Control signals are represented using an activated device name, activated clock cycle, and activating condition values. For a multiplexer, an activated port number attribute is needed as well. Each control signal, c_i, controls a functional unit, a storage element, or an interconnect component in the data path. The control signals are determined during allocation.

The controller specification depends on whether there are conditional branches or not. The next state is determined based on the next contiguous time step, as implied by the FSMD, except in the case of loops and branches. If there are no conditionals, then the state diagram is a ring with the clock cycles continuous sequence of number for a scheduled control/data flow graph. For operations on conditional branches, an analysis of the control/data flow graph mutually exclusive conditions is necessary. These are done by the scheduler. The state memory of the controller holds the present state, and a combinational circuit decides the next state and output signals. Once a condition value is produced, it is maintained until no more operations depend on it. For a design with conditional branches, the condition values are remembered by the controller until no more conditional execution paths are dependent on those condition values. The reason is that a controller using status registers has a simpler state transition diagram than a controller without status registers. This is done through register life analysis on nodes with conditions values. Thus, a condition would be alive when it is first defined by the scheduler and assigned to a register and when the last clock cycle that the execution of operations are dependent on the condition. The number of status register required to implement the controller is the maximum number of condition values reserved for any cycle during the execution.

In addition to the above, the controller incorporates additional control signal lines that are used only during test mode. These signals are responsible for the control of the datapath *test registers* in addition to setting necessary test registers modes in a BILBO (Normal, MISR, TPGR). Furthermore, the controller must incorporate the necessary logic that will iterate through the necessary test kernels in the test sessions as indicated by the test schedule.

B. Distributed Controller

Distributed controllers tend to consume more power than centralized. However, the key is to ensure that distributed controllers are not active at the same time as total parallelism is not possible to achieve. Thus, the distributed controller is synthesized using hierarchical Finite State Machine (FSM) model where every test kernel has its own sub-controller. A relatively larger controller will coordinate the execution of the sub-controllers (Figure 2).

Each state in the distributed control corresponds to a clock cycle; however, the sub-controller does not necessarily have actions to execute in every clock cycle. The distributed controller scheme is based on a state machine model where the states include necessary signals needed for the proper execution of operations in the kernel *only* including the selection of the muxes inputs and the register modes in the kernel. The next

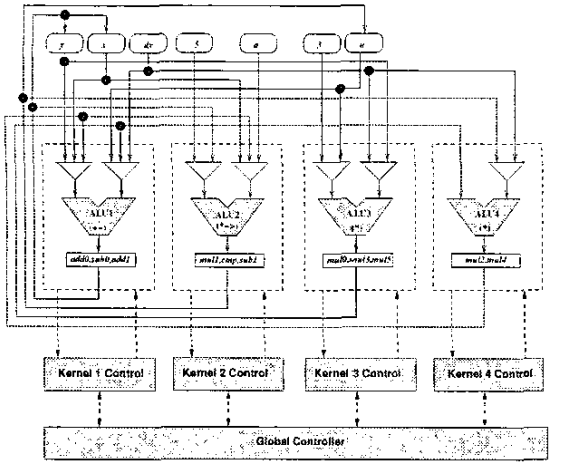

Fig. 2. Datapath with Distributed Control

```
ENTITY Gate IS
PORT (Clock: IN BIT;
      OpT: IN BIT_VECTOR(3 DOWNTO 0);
      Mode: IN BIT;
      Clk_Out: OUT BIT);

ARCHITECTURE Behavior OF Gate IS
      BEGIN
          PROCESS (Clock, Mode, OpT)
          VARIABLE Clk:BIT:=0;
          BEGIN
                IF Clock'Event and Clock = '1' THEN
                  IF Mode = '1' and OpT=B"000" THEN
                    Clk_Out <= 0;
                  ELSE
                    Clock_Out = Clk;
                    Clk := NOT(Clk);
                  END IF;
                END IF;
          END PROCESS;
      END;
```

Fig. 3. Gate VHDL Description

states are determined based on the next activation clock cycle in the kernel, as implied by the schedule. This may include loops and branches.

During distributed control execution, there are registers that maybe shared among one or more test kernel. These registers could be either *controllable* or *observable*. In order to better quantify the effect of power consumption in both models, we have implemented two test mechanism as follows:

1) *Option 1:* Kernels are formed of muxes, ALU's and output registers with all I/O belonging to the main controller.
2) *Option 2:* Registers are assigned to a kernel based on their test function.

The first option delegates the registers control to the global controller, while the second option, although results in a larger kernel size, reduces the size of the global controller, since fewer registers are controlled directly. The system will generate both types of kernels for comparison reasons.

The global controller controls the distributed sub-controllers via control signals. The sub-controllers act as decoders that are triggered by the main controller at the corresponding clock cycle. For each sub-controller, we construct a state or decoding table. An optimization step is done to remove duplicate control signals that target the same registers form and to different sub_controllers.

During test mode, the controller requires an additional external signal to distinguish between test and system mode. Each test session consists of three phases. 1) Initialization step, where the test pattern generators are initialized with a value different from zero and the compressors are set to a defined state; 2) test Execution, and 3) shifting out the corresponding signatures. A counter is next initialized with the corresponding test length before each test session. The counter's output is connected with the global controller. The controller also specifies various constants such as the number of test sessions, the scanpath length, the number of registers between a test pattern generator and a signature analyzer. In the scan path all registers are included, which act as a compressor in a test session. After each test session the signatures are shifted out.

C. Power Reduction

BIST circuits typically result in considerably higher circuit activity rate compared to normal mode operation. This is due mostly to the higher switching activity of a digital circuit and is, in general, the major source of power consumption [10]. Thus, the difference between BIST and normal modes is in the activity rate, which is, for instance, assumed 0.5 in the case of pseudo-random based BIST schemes, and often less for normal mode operation [6]. In order to obtain a realistic power dissipation for a random logic block, a predetermined percentage (ex. 10%) needs to be added to the block size N, corresponding to the average area overhead for the selected BIST scheme.

To solve the above problem, we use *RTL clock gating* at the level of each distributed controller. Thus, according to the test sequence, we can take advantage of the fact that not all test kernels are active during the same clock. This means that the non-active test kernels are shutdown using gating. Thus, by grounding their clock during test mode based on a specific signal (*opT* in Figure 3), where the signal has the information indicating when the kernel under test is active. Thus, yielding less switching activity and hence less power consumption.

RTL clock gating have many advantages. First, they are easy to implement and maintain, technology independent, and requires no RTL code change. Furthermore, they deliver significant power savings due to a reduced internal power consumption in gated register, reduced capacitance (switching power) on clock network, and minimizes area (muxes not needed).

It should be noted that one can always optimize the test schedule to further reduce the power consumption and in the extreme case would be to activate sequentially all the test kernels. However, this would increase the test time and reduces the parallelism in the distributed solution.

TABLE I

THE DYNAMIC POWER MEASURED USING SYNOPSYS SIMULATION FOR A 4-BIT BUS

Benchmark Example	*Mode*	*# Central*	*# Option1*	*# Option1G*	*# Option2*	*Option2G*
Elliptic	Norm	22.4693 uW	16.5983 uW	12.7921 uW	16.4339uW	8.2887 uW
BILBO	Test	8.7646 uW	25.9482 uW	7.9905 uW	80.3593 uW	7.1345uW
Elliptic	Norm	21.6776 uW	15.5717 uW	11.0283 uW	15.4314 uW	7.3950 uW
Selection	Test	8.5584 uW	24.5886 uW	7.6763 uW	79.7385 uW	6.8329uW
FIR	Norm	71.9638 uW	74.6128 uW	66.8155 uW	74.6128 uW	66.8155 uW
BILBO	Test	64.1252 uW	123.3939 uW	62.1604 uW	123.3939 uW	62.1604 uW
FIR	Norm	41.4004 uW	35.1743 uW	30.0457 uW	35.1743 uW	30.0456 uW
Selection	Test	28.9279 uW	63.7957 uW	26.7423 uW	63.7957 uW	27.7330 uW
Wavelet	Norm	72.1938 uW	39.4201 uW	16.6070 uW	39.3697 uW	16.1519 uW
BILBO	Test	21.2511 uW	107.0060 uW	11.0410 uW	106.9427 uW	10.6488 uW
Wavelet	Norm	42.3162 uW	38.9226 uW	12.4490 uW	39.1628 uW	16.1519 uW
Selection	Test	8.7174 uW	35.1173 uW	8.7757uW	106.9427uW	10.6488 uW
DCT4	Norm	28.5887 uW	28.7692 uW	19.6521 uW	27.9099 uW	14.6900 uW
BILBO	Test	18.4989 uW	89.1389 uW	9.3732 uW	88.9121 uW	8.8147uW
IIR	Norm	93.4799 uW	83.5396 uW	18.2383 uW	83.3994 uW	16.6751 uW
BILBO	Test	19.1457 uW	85.0118 uW	10.7404 uW	81.2435 uW	8.0898 uW

IV. RESULTS

We have implemented our system using C++ based on the MFC library. The tool takes a structural datapath description in VHDL that includes the test registers attribute, the data path width, and the test schedule and automatically generates a VHDL description for the controller and the datapath.

We measured the performance of our system for four data flow graphs which are *the 6^{th} order FIR filter (finite impulse response), the 3^{rd} order IIR filter (infinite impulse response), the 4-point DCT (Discrete Cosine Transform) circuit*, and *the 6-tap wavelet filter*. Our system successfully synthesized two different style data paths, a design style that is based on BILBO and a second style is based on a concurrent selection of test points [4]. The implementation details by our synthesis system are shown in Table I and in Figure 4. For every example, we show the circuit area and power dissipation based on Synopsys.

As shown in Table I, *Option 2* in the distributed control mechanism results in the best improvement in power dissipation with this option providing the most parallel solution. The improvement in power consumption is on the average 54.22% in normal mode while, as expected, the average power reduction decreases to 28.56% in the test mode due to the high switching activity. In specific, the improvement ranges from 7.15% in normal mode in the case of FIR BILBO to 82.16% in the case of the IIR FILBO. During testing mode, the improvement ranges from 3.5% in the case FIR BILBO to FIR BILBO to 57.74%. The benchmarks design area, based on Synopsys, are shown in Figure 4.

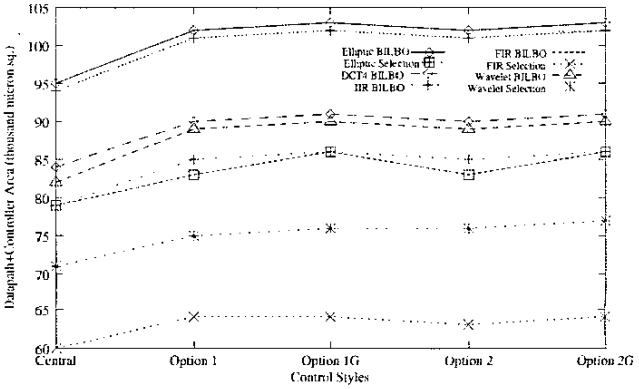

Fig. 4. Datapaths Area with Distributed Control

REFERENCES

[1] D. Singh, J.Rabaey, et al.,"Power Consious CAD Tools and Methodologies: A Perspective," *in Proc. DAC*, pp. 242-247, June 1995.

[2] G. Craig, C. Kime, and K. Saluja, "Test Scheduling and Control for VLSI Built-In Self-Test," *IEEE Trans. On Computers*, Vol. C-37, pp. 1099-1109, 1988.

[3] H. Harmanani, C. Papachristou, "An Improved Method for RTL Synthesis with Testability Trade-Offs," *in Proc. ICCAD*, pp. 30-37, 1993.

[4] H. Harmanani, M. Kodeih, "Concurrent BIST Cost Estimation During Data Path Allocation," *in Proc. of the First NEWCAS*, pp. 57-60, 2003.

[5] M. Nourani, J. Carletta, C. Papachristou, "Synthesis for Testability of Controller-Datapath Pairs that Use Gated Clocks," *in Proc. DAC*, 2000.

[6] M. Favalli, L. Benini, G. De Micheli, "Design for Testability of Gated-Clock FSMs," *in Proc. EDTC*, pp. 589-596, 1996.

[7] L. Benini, G. De Micheli, "Automatic Synthesis of Low-Power Gated-Clock FSM," *Trans. on CAD*, Vol. 15, Number 6, pp. 630-643, 1996.

[8] A. Crews, F. Brewer, "Controller Optimization for Protocol Intensive Applications," *in Proc. Euro-DAC*, 1996.

[9] A. Hertwig, H. Wunderlich, "Fast Controllers for Data Dominated Applications," *in Proc. ITC*, 1997.

[10] Y. Zorian, "A Distributed BIST Control Scheme for Complex VLSI Devices," *in Proc. IEEE VLSI Test Symp.* pp. 4-9, 1993.