Channel-Based Behavioral Test Synthesis for Improved Module Reachability*

Yiorgos Makris and Alex Orailoğlu
Reliable Systems Synthesis Lab – CSE Department
University of California, San Diego
La Jolla, CA 92093

Abstract

We introduce a novel behavioral test synthesis methodology that attempts to increase module reachability, driven by powerful global design path analysis. Based on the notion of transparency channels, test justification and propagation bottlenecks are revealed for each module in the design. Subsequently, the proposed behavioral test synthesis scheme eliminates, during scheduling, allocation and binding, as many reachability bottlenecks, as possible. Furthermore, it identifies the control states and provides the templates required for translating each module's test into global design test. We demonstrate our scheme on a representative example, unveiling the potential of path analysis based techniques to accurately identify and eliminate module reachability bottlenecks, thus guiding behavioral test synthesis.

1. Introduction

High-level synthesis [3] has become an inseparable part of modern circuit design, providing an automated mechanism of exploring hardware implementation alternatives while optimizing design attributes. Local neighborhood design optimizations, typically performed by high-level synthesis tools, have proven efficient for improving area, power and performance requirements. Nevertheless, future high-level synthesis success may rely on global design examination in order to efficiently address increasing performance requirements and additional attributes such as testability. Considering design reachability paths during high-level synthesis not only provides an edge for testability related optimizations but also enhances the scope of critical timing path and power dissipation improvements. Towards this direction, design path analysis information constitutes a precious resource for innovative high-level synthesis solutions.

Behavioral test synthesis [2,7] incorporates testability considerations in the high-level synthesis process. Its success has been questionable, however, mainly due to a discrepancy in the abstraction level of test paths and testability optimizations. Test has been traditionally based on structural accessibility paths, while behavioral test synthesis is applied on a high-level design model. The lack of a sufficient bridging mechanism between the two levels has limited the efficacy of previous approaches. Recently, the tremendous complexity of modern circuits and new approaches such as core-based designs have effected a paradigm shift in the test area. Viable test solutions are based on hierarchical schemes, wherein high-level module reachability path knowledge is required. Therefore, behavioral test synthesis for improved module reachability constitutes an innovative solution, akin to future test trends. In the proposed approach, a powerful channel-based path analysis methodology described in section 2, facilitates an efficient behavioral test synthesis scheme. An orthogonal approach for BIST through behavioral synthesis is presented in [5], while an early modular scheme can be found in [6].

2. Scheme overview

The proposed behavioral test synthesis scheme depicted in figure (1), targets circuits that are tested in a modular fashion. Local test is generated for each module and subsequently translated and applied from the design

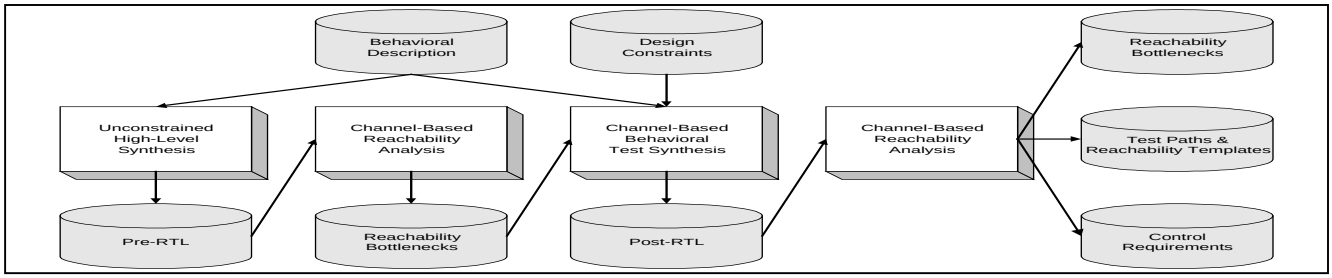


Figure (1): Proposed methodology scheme

* This work is supported in part through a research grant from Intel Corporation under contract CSE 0129-58678A.

boundary through test paths, thus eliminating complete circuit ATPG. However, due to lack of global design knowledge during local test generation, test translation relies on sufficient module access. The objective of the proposed method is to supply, by construction, adequate module access paths, thereby improving reachability. Such path information may further be used to generate the templates for translating local vectors into global test and the control states required for instantiating the test paths.

Starting with a behavioral design description, we perform an initial high-level synthesis step that transforms a scheduled data-flow graph (DFG) into an RTL representation, denoted as *Pre-RTL*. We analyze the RTL design and identify the reachability bottlenecks based on the notion of module *transparency channels*, as explained in section 3. These bottlenecks are considered in conjunction with design constraints during the scheduling, allocation and binding phases of the behavioral test synthesis step, described in section 4. This step attempts to eliminate as many reachability bottlenecks as possible from the resulting RTL representation *Post-RTL*, by increasing module test path accessibility. A final analysis phase is subsequently applied in order to identify any remaining bottlenecks and to supply the test paths, the reachability templates and the necessary control signals, as described in sections 5 and 6, respectively.

3. Channel-based reachability analysis

Test paths and bottlenecks are identified through an RTL test justification and propagation path analysis methodology. The complexity associated with examining exhaustively the functional space of a design during reachability analysis would doom any such approach. Instead, the proposed analysis scheme utilizes only test translation related behavior, defined in terms of transparency channels.

3.1. Methodology description

Transparency channels are *bijection functions* between input and output *signal entities* of a module. Transparency channels capture the ability of a module to become transparent during test justification and propagation and may be defined across clock cycles. Channels are activated based on the compliance of one or more *conditions* defined on signal entities and combined through operators. Controllability and observability of primary inputs and outputs is captured through the *well* and *drain* notion, respectively. Wells and drains comprise a *potential* that defines the types of signals that can be generated or evaluated. The reachability potential of a test path is the composite function of the transparency channels on it.

Transparency channels are combined through a design traversal algorithm that examines the reachability of each module for test justification and propagation. The analysis algorithm is capable of addressing in a uniform manner, combinational and sequential, data and control path logic. Furthermore, the algorithm handles efficiently feedback loops, reconvergent paths and variable bitwidth signals. A minimal set of bottlenecks is obtained through further analysis and combination of the bottlenecks identified for each module. A detailed description of the channel-based test reachability analysis scheme can be found in [4].

3.2. Example circuit

In order to demonstrate the proposed methodology, we will extensively use as an example circuit a complex pipelined multiplier accumulator (MAC), originally described in [1]. The MAC operates on two sequences of complex numbers $\{x_i\}$ and $\{y_i\}$, multiplying corresponding elements of the sequences and accumulating the sum of the products. Each complex number is represented in Cartesian form consisting of a real and an imaginary part. The MAC calculates the result by forming the complex product of successive pairs of complex numbers and accumulating them in a register in a pipelined fashion.

The MAC DFG is shown in figure (2), scheduled for 3 pairs. In figure (3) we depict the corresponding synthesized *Pre-RTL* block diagram and an example of how the analysis scheme identifies reachability bottlenecks by examining the potential at each module boundary. Table (1) summarizes the RTL circuit, pointing out the reachability bottlenecks that the behavioral test synthesis scheme will attempt to eliminate.

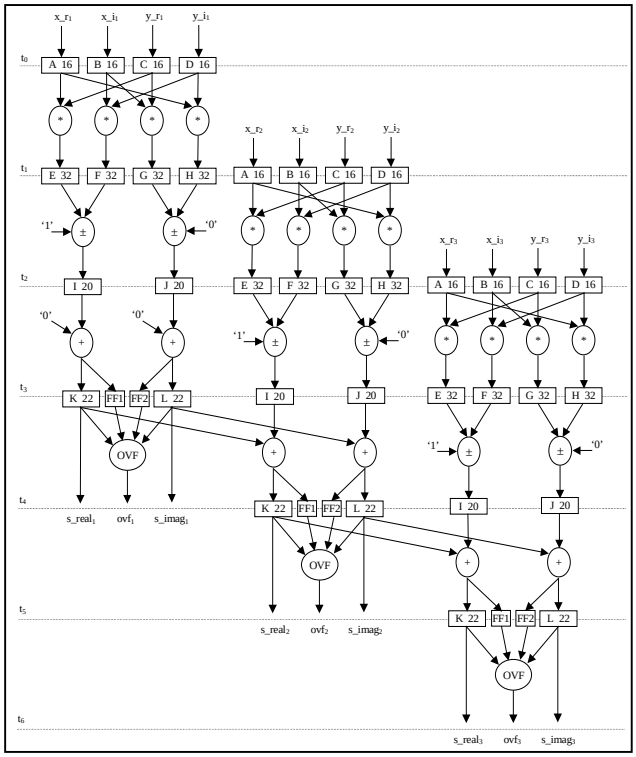
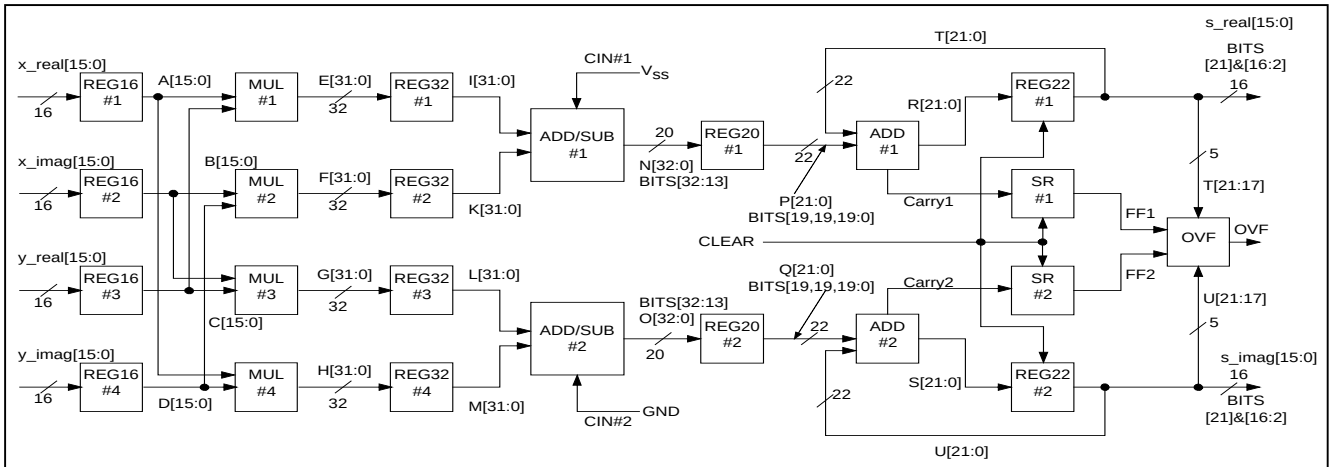


Figure (2): Scheduled DFG for MAC (3 pairs)



REACHABILITY OBJECTIVE EXAMINATION EXAMPLE: REG22#1

Objective: JUSTIFY (R[k], CLEAR), k=[0…21] at [t_0, t_0+1] ⇒ *THROUGH PRIMARY INPUT WELL CAPABILITIES AND ADD#1 CHANNEL (P[k] at [t] MAPS TO R[k] at [t] IF T[21:0]=constant)* ⇒ JUSTIFY P[k] at [t_0, t_0+1] **(justification of condition verifies its compliance through *clear='1' at[t_0-1] over feedback loop)*** ⇒ *THROUGH REG20#1 CHANNEL (N[j] at [t] MAPS TO P[t] at [t+1], j=[32…13])* ⇒ JUSTIFY N[j] at [t_0-1,t_0] ⇒ *THROUGH ADD/SUB#1 CHANNEL (I[j-1] at [t] MAPS TO N[j] at [t] IF SAME (I[j-1], K[j-1]) at [t])* ⇒ JUSTIFY SAME (I[j-1], K[j-1]) at [t_0-1, t_0] ⇒ *THROUGH REG32#1,2 CHANNELS (E[j-1] at [t] MAPS TO I[j-1] at [t+1], F[j-1] at [t] MAPS TO K[j-1] at [t+1])* ⇒ JUSTIFY SAME (E[j-1], F[j-1]) at [t_0-2, t_0-1] ⇒ *THROUGH MUL#1,2 CHANNELS (A[m] at [t] MAPS TO at E[n] at [t], m=[15…0], n=[31, 28…0], B[m] at [t] MAPS TO F[n] at [t])* ⇒ JUSTIFY SAME (A[m], B[m]) at [t_0-2, t_0-1] **(Controllability bottlenecks E[30:29], F[30:29])** ⇒ *THROUGH REG16#1,2 CHANNELS (x_real[m] at [t] MAPS TO A[m] at [t+1], x_imag[m] at [t] MAPS TO B[m] at [t+1])* ⇒ JUSTIFY SAME (x_real[m], x_imag[m]) at [t_0-3, t_0-2] ⇒ *THROUGH PRIMARY INPUT WELL CAPABILITIES* ⇒ **Satisfied**

Figure (3): Unconstrained synthesized RTL diagram of MAC – Reachability analysis example

H/W Resources	*4 16-bit Registers, 4 32-bit Registers, 2 20-bit Registers, 2 22-bit Registers, 2 SR Flip-Flops, 4 32-bit Multipliers, 2 32-bit Add/Subs, 2 22-bit Adders, 1 Overflow Unit*
Latency	*4 clock cycles*
Throughput	*1 sum per clock cycle*
Reachability Bottlenecks	***Controllability:*** *Full Potential on E[30:27] F[30:27] G[30:27] H[30:27] P[21:20] Q[21:20] E[11:0] F[11:0] G[11:0] H[11:0] at [t , t+1] Constant '0' on CIN#1 at [t] Constant '1' on CIN#2 at [t]* ***Observability:*** *Full Potential on N[12:0] O[12:0] T[1:0] U[1:0] at [t, t+1]*

Table (1): Synthesized MAC summary

4. Channel-based behavioral test synthesis

The powerful path information, provided by the channel-based reachability analysis methodology, may guide high-level synthesis optimization decisions for path-related design attributes, such as testability. Within the proposed scheme, testability is evaluated in terms of module reachability according to the test justification and propagation bottlenecks revealed by the channel-based reachability analysis on the *Pre-RTL*. Resolution of these bottlenecks pinpoints the reachability path improvements that behavioral test synthesis needs to accomplish on the design, in order to increase testability.

Channel-based, behavioral test synthesis attempts to eliminate the reported bottlenecks during allocation, scheduling and binding, while observing the design constraints. During allocation, we attempt to maximize the number of module reachability paths, by providing functional units with the appropriate transparency channels. During scheduling, this maximization is performed across clock cycles. The binding phase further increases reachability through test path combination, while assigning variables. A key point of the behavioral test synthesis transformations is that they can be applied on sub-word, variable bit signal entities, as demonstrated in sections 4.1 and 4.2.

4.1. Reachability through allocation & binding

In this case, the behavioral test synthesis algorithm considers the scheduling specifications of the design, along with the reachability bottlenecks identified through the channel-based path analysis. Scheduling specifications define the minimum number of functional units required for satisfying the design constraints. Subsequently, binding attempts to maximize reachability of each module through variable assignment to registers. Registers are combined or split based on primary input/output proximity and existing paths of transparency channels, targeting the bottlenecks reported by the analysis. In case variable binding cannot resolve the bottlenecks of a module, additional functional units are allocated and the process is repeated until no more bottlenecks are resolved.

In the pipelined MAC, with scheduling constraints of a latency of 4 clock cycles and a throughput of 1 sum per 2 clock cycles, the algorithm for module reachability optimization through allocation and binding, results in the DFG depicted in figure (4). The corresponding synthesized RTL is shown in figure (5) and the circuit is summarized in table (2). As an example, to ensure maximum reachability of the *ADD/SUB#1* unit inputs, 4 16-bit registers are used. Through the transparency channels of the 2 allocated multipliers, the 16 LSB are bound to registers *E* and *F*, while the non-controllable 16 MSB are stored in the input registers *A* and *B*. Thus, the complete *ADD/SUB#1* input space is controllable. Under the given scheduling specifications, our behavioral test synthesis scheme eliminates 96% of the controllability bottlenecks in the original DFG but is unable to eliminate any observability bottlenecks.

H/W Resources	*8 16-bit Registers, 2 22-bit Registers, 2 SR Flip-Flops, 6 16-bit Multiplexers, 2 6-bit Multiplexers, 4 32-bit Multipliers, 2 32-bit Add/Subs, 2 22-bit Adders, 1 Overflow Unit*
Latency	*4 clock cycles*
Throughput	*1 sum per 2 clock cycles*
Reachability Bottlenecks	***Controllability:*** *Constant '0' on CIN#1 at [t] Constant '1' on CIN#2 at [t]* ***Observability:*** *Full Potential on N[12:0] O[12:0] T[1:0] U[1:0] at [t, t+1]*

Table (2): Allocated & bound MAC summary

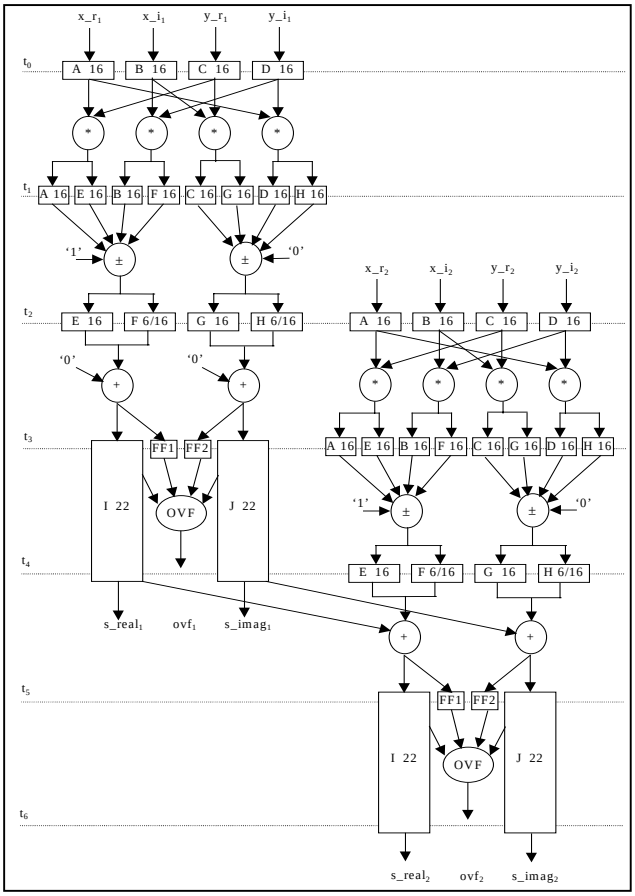


Figure (4): DFG for allocated & bound MAC

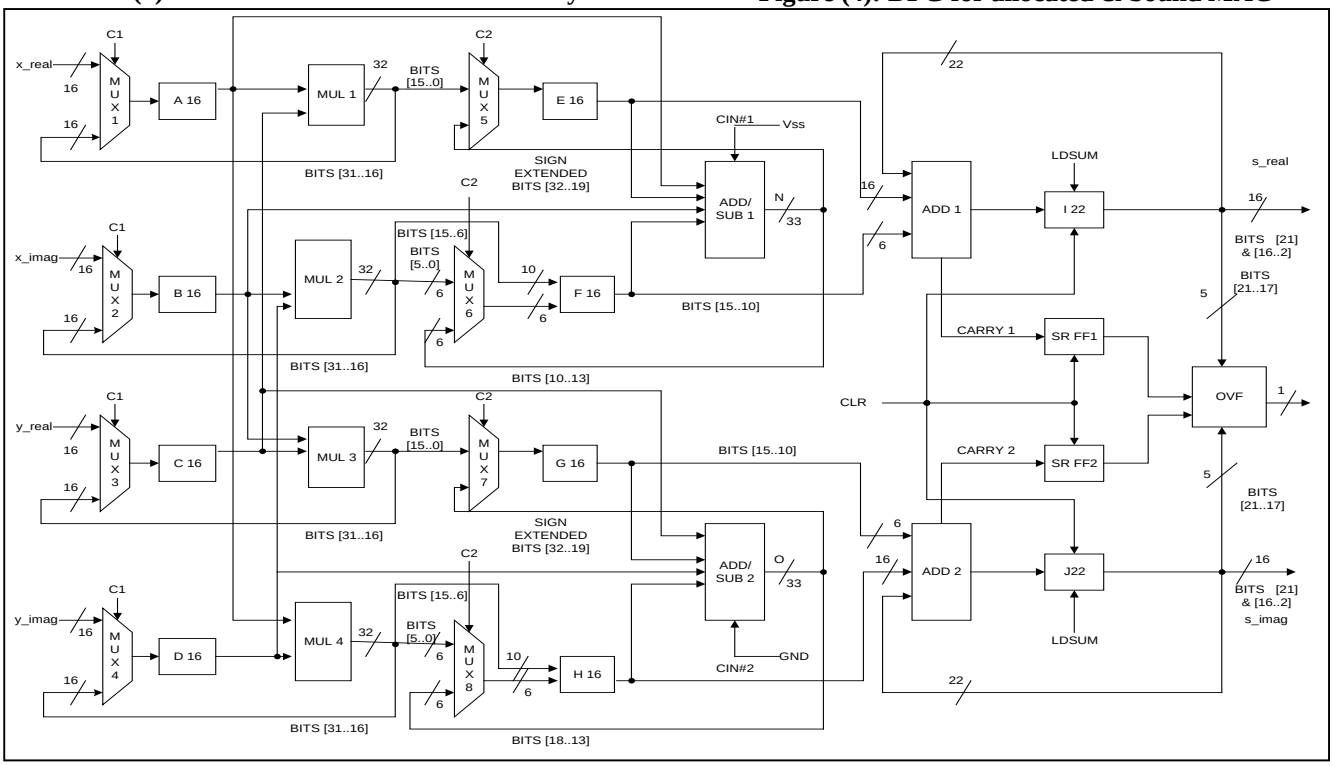


Figure (5): Synthesized RTL block diagram of MAC – Optimizing reachability through allocation & binding

4.2. Reachability through scheduling & binding

In this case, with given allocation constraints, the reported bottlenecks are resolved through scheduling and binding, maximizing module reachability. Scheduling assigns in time the operations to the given functional units, minimizing the latency and throughput of the circuit. Subsequently, binding attempts to maximize reachability of each module through variable bitwidth register assignment. If reachability bottlenecks still exist, rescheduling with increased latency and throughput is attempted until no more bottlenecks can be resolved.

Given allocation constraints that allow two multipliers and two *ADD/SUB* units for the pipelined MAC, the behavioral test synthesis algorithm results in the DFG and the RTL diagram depicted in figure (6) and figure (7). Table (3) summarizes the circuit, revealing the success in bottleneck elimination. All the controllability bottlenecks and 85% of the observability bottlenecks are eliminated. Four new controllability bottlenecks are introduced and reported through the final reachability analysis phase.

H/W Resources	*9 16-bit Registers, 2 22-bit Registers, 2 SR Flip-Flops, 8 16-bit Multiplexers, 4 23-bit Multiplexers, 2 6-bit Multiplexers, 2 32-bit Add/Subs, 1 Overflow Unit*
Latency	*5 clock cycles*
Throughput	*1 sum per 3 clock cycles*
Reachability Bottlenecks	***Controllability:*** *Constant '1' on 1-bit input of Multiplexers MUX12, MUX13, MUX14, MUX15 at [t]* ***Observability:*** *Full Potential on T[1:0] U[1:0] at [t, t+1]*

Table (3): Scheduled & Bound MAC Summary

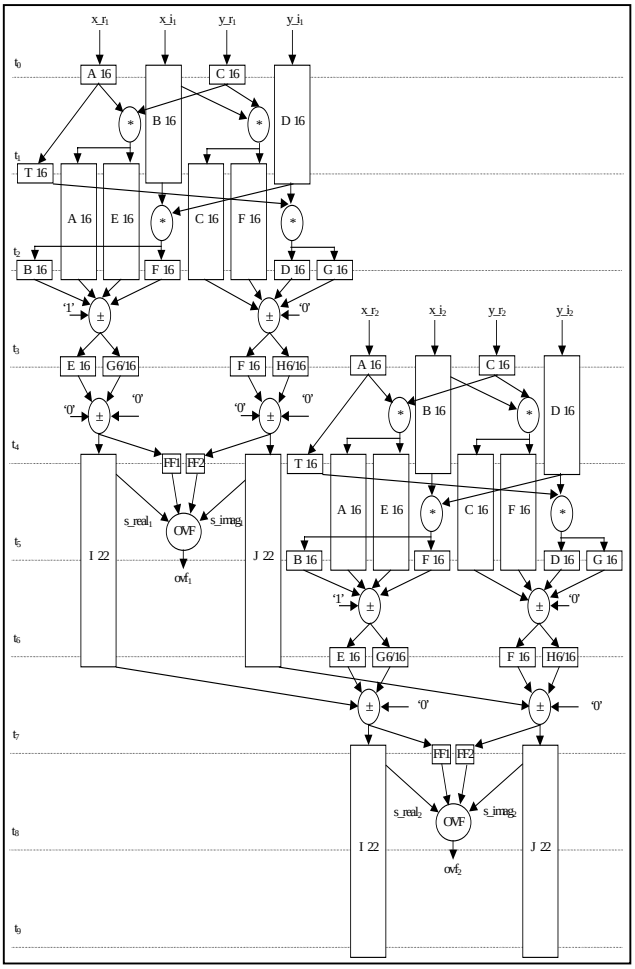


Figure (6): DFG for scheduled & bound MAC

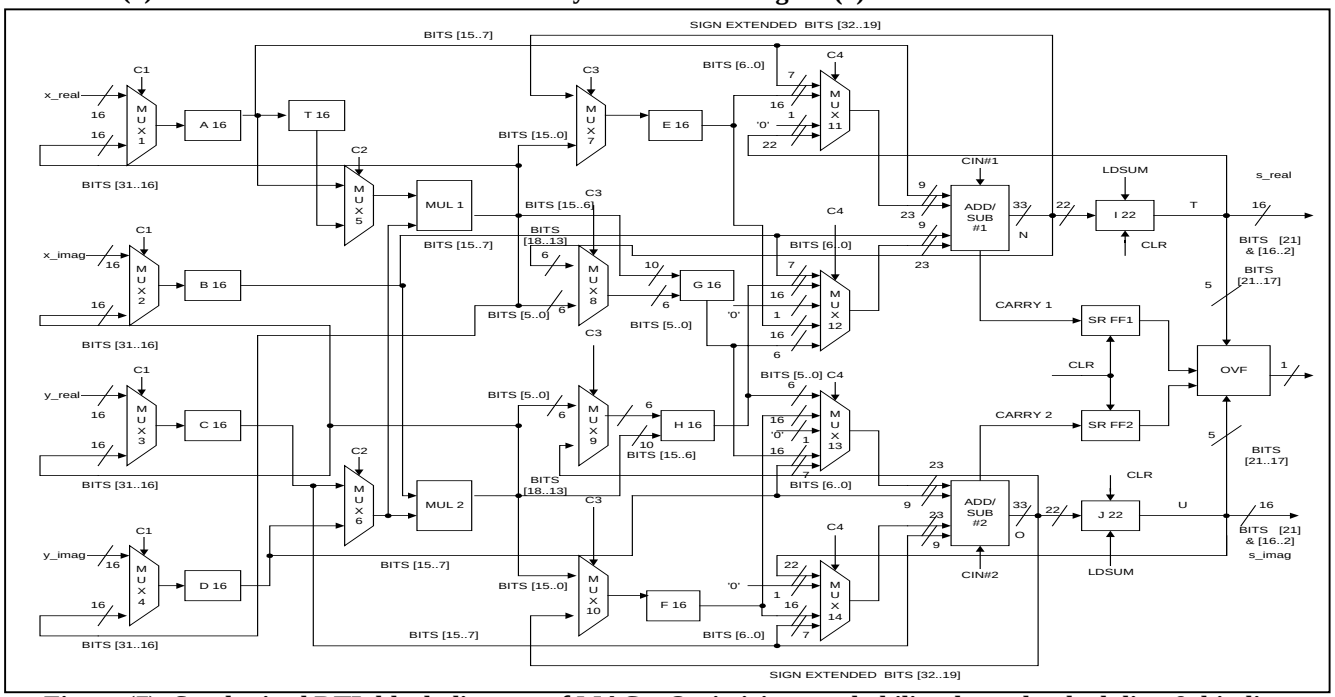


Figure (7): Synthesized RTL block diagram of MAC – Optimizing reachability through scheduling & binding

5. Test paths & reachability templates

In this section we demonstrate how the reachability analysis scheme of section 3 can be applied on the *Post-RTL*, in order to assist the local to global test translation process. While the previously outlined reachability analysis scheme identifies the remaining bottlenecks and the test justification and propagation paths in the design, instantiation of each such test path further requires a number of conditions to be fulfilled. Based on the test paths, the conditions and the transparency channels, reachability templates are composed. These templates are used to automate module test translation and eliminate the need for global circuit ATPG. Thus test can be locally generated at the boundary of each module and subsequently translated through the templates into test meaningful at the global design boundary. In figure (8), we illustrate a test justification template for the *ADD/SUB#1* of the MAC of figure (6), where *f* and *g* are the composite channel functions on each path.

	Test Path	**Conditions**
t_0	x_real=x_r_1 x_imag=x_i_1	C_1=0, y_real='0…01', y_imag='0…01'
t_1	x_real=x_r_2 x_imag=x_i_2	C_1=0, C_2=0, y_real='0…01', y_imag='0…01'
t_2	ADD/SUB#1 INPUT1 = f(x_r_2, x_r_1) ADD/SUB#1 INPUT2 = g(x_i_2, x_i_1)	

Figure (8): Example template for ADD/SUB#1

6. Control logic implications

During channel-based reachability analysis, behavioral test synthesis and test template identification, we have assumed complete controllability of the control signals. In reality, these signals are driven by a control logic FSM, wherein states and transitions are encoded and minimized. Our channel-based scheme exploits, for test purposes, module reachability paths that are not part of design functionality (false paths). The instantiation of these paths relies upon the availability of the appropriate control signals. It is essential, therefore, that the additional control logic requirements be defined and considered during the control logic implementation.

Within our scheme, the additional control logic requirements can be easily identified, by examining the test paths and templates. The conditions for instantiating transparency channel paths provide information for the required control logic behavior. The states and transitions necessary for false test path instantiation are captured and subsequently utilized during control logic implementation. In figure (9), we demonstrate the functional control requirements for the scheduled and bound MAC of figure (6) and the additional control for testing the modules.

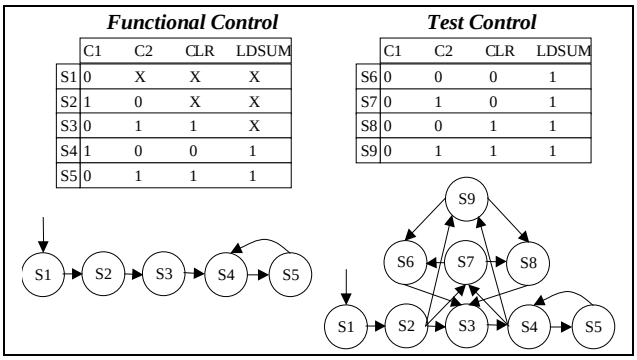


Figure (9): Control logic for scheduled & bound MAC

7. Conclusion

We introduce in this paper a novel methodology for improving module reachability for test purposes, through behavioral test synthesis. The proposed scheme commences with a reachability analysis phase that identifies hierarchical design testability bottlenecks, based on the notion of transparency channels. These bottlenecks, endangering the translation of local module vectors into global design test, are subsequently minimized through the scheduling, allocation and binding phases of the proposed behavioral test synthesis approach. A final analysis phase provides the local to global test translation templates, eliminating the need for global circuit ATPG. Appropriate control states and transitions for module test reachability are also obtained, in order to guide efficient control logic FSM implementation. The effectiveness of the scheme is experimentally validated, reflecting the efficacy of behavioral test synthesis approaches based on path analysis and encouraging further research in the area.

References

[1] P. Ashenden, *The Designer's Guide to VHDL*, Morgan-Kaufmann Publishers Inc., 1996.

[2] L. Avra, E. J. McCluskey, "High-Level Synthesis of Testable Designs: An Overview of University Systems", *Digest of Papers, Test Synthesis Seminar, ITC*, pp. 1-8, 1994.

[3] D. Gajski, N. Dutt, A. Wu, S. Lin, *High-Level Synthesis: Introduction to Chip and System Design*, Kluwer Academic Publishers, 1992.

[4] Y. Makris, A. Orailoğlu, "RTL Test Justification and Propagation Analysis for Modular Designs", *Journal of Electronic Testing: Theory & Applications*, vol. 13, no. 2, pp. 105-120, 1998.

[5] A. Orailoğlu, I. G. Harris, "Microarchitectural Synthesis for Rapid BIST Testing", *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 16, no. 6, pp. 573-586, 1997.

[6] P. Vishakantaiah, T. Thomas, J. A. Abraham, M. S. Abadir, "AMBIANT: Automatic Generation of Behavioral Modifications for Testability", *ICCD*, pp. 63-66, 1993.

[7] K. Wagner, S. Dey, "High-Level Synthesis for Testability: A Survey and Perspective", *33rd DAC*, pp. 131-136, 1996.