Symbolic Verification of Timed Asynchronous Hardware Protocols

Krishnaji Desai and Kenneth S. Stevens
Electrical and Computer Engineering
University of Utah, USA
Email: krishnaji.desai@utah.edu, kstevens@ece.utah.edu

John O'Leary
Intel Corporation
Hillsboro, Oregon, USA
Email: john.w.oleary@intel.com

***Abstract*—Correct interaction of asynchronous protocols requires verification. Timed asynchronous protocols add another layer of complexity to the verification challenge. A methodology and automated tool flow have been developed for verifying systems of timed asynchronous circuits through compositional model checking of formal models with symbolic methods. The approach uses relative timing constraints to model timing in asynchronous hardware protocols – a novel mapping of timing into the verification flow. Relative timing constraints are enforced at the interface external to the protocol component. SAT based and BDD based methods are explored employing both interleaving and simultaneous compositions. We present our representation of relative timing constraints, its mapping to a formal model, and results obtained using NuSMV on several moderate sized asynchronous protocol examples. The results show that the capability of previous methods is enhanced to enable the hierarchical verification of substantially larger timed systems.**

I. Introduction

Asynchronous hardware protocols are hard to implement and hard to get right. Formal verification is required to ensure that individual protocols implemented with logic gates are realized correctly. This is the domain of conformance checking, in which an implementation of a leaf-level hardware component is checked for adherence to a specification [1]. In this scenario both a global specification and an exact model of the system's environment are usually unavailable, but model checking can still be employed to verify deadlock freedom, liveness, safety and other necessary correctness properties. This paper addresses the composition problem: we present an approach for automatic generation of asynchronous system models and their properties, with a push button CAD solution to verify their correctness.

Verification complexity grows significantly when timing must be considered since both logic and timing interact to effect design correctness. Formal analysis of asynchronous protocols usually employs explicit state enumeration and is especially vulnerable to state explosion. One focus of this paper is to reduce the run time of system level verification by applying symbolic model checking techniques to help mitigate the state explosion problem [2].

The performance and power of asynchronous hardware circuits and protocols can be vastly improved with judicious application of timing constraints. The second primary focus of this work is to implement in commercial-grade model checking engines a *relative* timing model. With *Relative timing* (RT), delays in a system are represented by their net effect on system behavior: the sequencing or ordering of signals in related race paths [3]. We start with a set of relative timing constraints that have been automatically generated for each timed handshake protocol module used in a system. The RT constraints are generated by an explicit state verification engine, ARTIST (Automatic Relative Timing Identifier based on Signal Traces) [4], that is based on the bisimulation formalism. This work integrates these timing constraints into verification models to prove timed behavioral correctness of systems employing timed protocol elements.

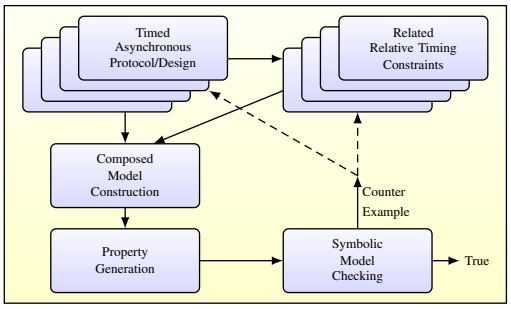


Fig. 1. Illustrating the Model Checking Flow.

The basic CAD flow reported here is illustrated in Fig. 1. A set of timed handshake protocol elements are designed and their associated timing constraints are generated and formally proven correct and complete. These are composed into a system that is to be verified by this work. A formal timed model is constructed from the specifications and related RT constraints. This model is checked for correctness with the industry-strength symbolic engine NuSMV [5]. A counter example may indicate a bug in the protocol or the need to tighten the RT constraints to ensure correct operation.

II. Background

Asynchronous systems do not have the global synchronization that a clock signal provides in synchronous systems, and therefore must rely on handshake protocols for correct sequencing and behavior. Timing constraints must be enforced in timed protocols to avoid any input changes in unaccepting states to a protocol. These timing constraints ensure compositional correctness of the composed modules. Fig. 2 shows

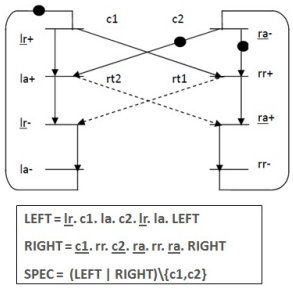


Fig. 2. Petri net and CCS specification of timed protocol LC.

the petri net and Calculus of Communicating Systems (CCS) specification for a linear pipeline controller (LC) example [6], [7]. This specifies a timed handshake protocol with `lr` (left request) and `ra` (right acknowledge) as inputs and `rr` (right request) and `la` (left acknowledge) as the outputs. LEFT and RIGHT processes in the CCS specification are barrier synchronized with causal arcs `c1` and `c2`. Dotted arcs `rt1` and `rt2` are the relative timing arcs in the petri net that explicitly enforce timing by constraining the input arrival times.

Relative timing (RT) is a method that explicitly represents the signal-ordering effects timing provides to a system. Timing constraints are represented by the equation **pod** $\mapsto$ **poc**$_0$ $\prec$ **poc**$_1$, where poc$_0$ must strictly occur before poc$_1$. Both events have a common timing reference represented by the point-of-divergence signal `pod`. When the constraints are enforced, the state space is pruned due to subgraph elimination.

These results are general and apply to any system that can be represented as a set of communicating timed asynchronous protocols.

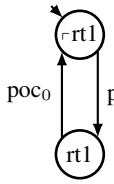
III. Relative Timing Modeling Techniques

NuSMV is used as the modeling language. It provides various constructs to model the protocols. Assign statements are used for initializing and updating variables using the `init` and `next` operations. The `union` keyword is used for introducing non-determinacy. This work builds specifications using both simultaneous and interleaving models. Speed-independent asynchronous systems are modeled using unbounded gate delays and no wire delays, whereas delay-insensitive systems use an unbounded delay model for both gates and wires.

A. Modeling Relative timing

Relative timing constraints enforce specific signal sequencing to occur. These constraints interact with the system through RT variables as shown in Fig. 3. Each RT constraint has an associated RT variable. Each variable is set by the `pod` signal and reset by the poc$_0$ signal from the RT equation pod $\mapsto$ poc$_0$ $\prec$ poc$_1$. The signal poc$_1$ is constrained to fire only when the RT variable is not set. RT variables are introduced at the level of hierarchy in a design where the `poc` and `pod` signals reside, and are implemented as independent processes that do not modify the logical behavior of a protocol.

For example, the arc `rt1` in Fig. 2 is a timing constraint represented by lr+ $\mapsto$ rr+ $\prec$ lr-. The variable `rt1` will



rtc = rt1: pod $\mapsto$ poc$_0$ $\prec$ poc$_1$

poc$_1$ can fire only when the relative timing variable rt1 is reset – i.e., after poc$_0$ fires.

Fig. 3. Modeling RT Constraint.

be asserted upon `lr+` and reset with `rr+`. Signal `lr-` is constrained to fire only when variable `rt1` is not asserted, which guarantees signal ordering between `rr+` and `lr-`.

B. Interleaving modeling technique

Interleaving semantics are a "natural" model for asynchronous designs. Interleaving models allow a single process at a time to undergo a state transition. This is modeled with the transition relation *R* in Eqn. 1. The disjunctive firing of components models delay-insensitive asynchronous protocols.

$$R = \vee_i R_i \text{ where } R_i = (v_i' \Leftrightarrow f_i) \wedge (\wedge_{j \neq i}(v_j' \Leftrightarrow v_j)) \quad (1)$$

In every transition R, the new modified value v_i' is evaluated with function f_i for one of the component state variables and the rest of the variables remain unchanged.

The first step to automate the model generation is to convert the formal protocol specifications (Fig. 2) into a minimized state graph. Minimized specifications are preprocessed with a state graph utility developed for adding rise/fall transitions based on the initial signal conditions. The sequential behavior of the state graph, output signals, and RT variables are then modeled with init() and next() operators in ASSIGN statements. The NuSMV code for LC `left` and `right` processes of Fig. 2 are shown. In general, observation of signal changes only occurs during the execution of the process. Hence, when modeling in NuSMV, the signal events corresponding to `pod` and `poc` that set and reset RT variables cannot be probed at the compositional level of hierarchy. In this code, the RT variable `rt1` is set by `left` upon transition from state `s0` to `s1` on `lr+` signal, and is reset by `right` upon transition from state

```
MODULE left(lr,c1,c2,rt1,rt2)
VAR
state: {s0,s1,s2,s3};
la:boolean;--output
ASSIGN
init(state) := s0;
next(state) := case
  (state = s0) & (lr=1): s1;
  ...
  1: state;
esac;

ASSIGN
init(la) := 0;
next(la) := case
  state=s1&(next(state)=s2):1;
  ...
--rt1 set
ASSIGN
init(rt1) := 0;
next(rt1) := case
(state=s0)&(next(state)=s1): 1;
1                    : rt1;
esac;
--rt2 reset similarly
--c1, c2 are set,reset
esac;
FAIRNESS running
```

```
MODULE right(ra,c1,c2,rt1,rt2)
VAR
state: {s4,s5,s6,s7};
rr:boolean;--output
ASSIGN
init(state) := s5;
next(state) := case
   (state = s5) & (c1=1) : s6;
   ...
esac;

--rt1 reset
ASSIGN
next(rt1) := case
  (state=s5)&(next(state)=s6): 0;
  1 : rt1;
esac;
--rt2 set similarly
--Also rr is set/reset,
--c1, c2 are reset,set

MODULE lc(li,ri)
VAR
c1:boolean; c2:boolean;
rt1:boolean; rt2:boolean;
lproc:process left(li,c1,c2,rt1,rt2);
rproc:process right(ri,c1,c2,rt1,rt2);
FAIRNESS running
```

s5 to s6 on rr+ signal. The processes left and right both participate in the shared constraints c1, c2, rt1 and rt2 so these signals are exposed at the interfaces of left and right and hidden within the composed module lc.

The signal ordering imposed by RT constraints is modeled without changing the internal behavior of a protocol. This is achieved by modifying the behavior of an input to a protocol from its environment. The model for two LCs composed with two RT constraints modeled at the interface is shown and pictured in Fig. 4. Constraint rt1 from Fig. 2 is lr+ $\mapsto$ rr+ $\prec$ lr-. This ensures that lc1.rproc.rr (rr+) occurs before lc1.lproc.lr (lr−) by constraining lc0.rproc.rr as c_lr to lc1.lr. The constraint rt2 ensures that lc0.lproc.la occurs before lc0.rproc.ra by constraining lc1.lproc.la as c_ra to lc0.ra of module lc0 by probing rt2. As can be seen, the mapping of RT variables and constraining on to design instances can be relatively complicated. This CAD tool generates the mapping automatically.

```
MODULE main                    ASSIGN
VAR                            init(c_lr) := 0;
li:boolean;ri:boolean;         next(c_lr) := case
c_lr:boolean;c_ra:boolean;      (lc0.rproc.rr=1) : 1;
....                            (lc0.rproc.rr=0)
lc0:process lc(li,c_ra);          & (lc1.rt1=0) : 0;
lc1:process lc(c_lr,ri);       1                : c_lr;
                               esac;
                               --Similarly ASSIGN for c_ra
```

C. Simultaneous modeling technique

In simultaneous models, all processes undergo state transitions in lock-step. Each process, however, makes an independent choice to either transition to a new state or "stutter" in its current state. Modeling of the simultaneous model in NuSMV requires the following changes:

i. The union keyword is used to model the arbitrary behavior of next state transitions [2].
ii. Module instances are not composed using the process keyword.

Simultaneous semantics model speed-independent delays by making input changes and their associated state transitions atomic, while modeling arbitrary delays for the outputs. Thus all processes observe and react to inputs simultaneously giving wires zero delay. Eqn. 2 formally describes the output relation O, and Eqn. 3 the input and state change relationship with v_i' being evaluated with function f_i.

$$O = \wedge_{1\leq i\leq n} O_i \text{ where } O_i = (v_i' \Leftrightarrow f_i) \vee (v_i' \Leftrightarrow v_i) \quad (2)$$

$$I = \wedge_{1\leq i\leq n} I_i \text{ where } I_i = (v_i' \Leftrightarrow f_i) \quad (3)$$

The NuSMV model for simultaneous switching is derived similarly to the interleaving model with the above changes listed. Code snippet is not shown for brevity.

IV. Property Verification

A. Deadlock Freedom

A composition of asynchronous protocols must be free from deadlocks. This is verified by checking the reachability of the initial protocol states using possible paths with CTL properties [8]. The properties checked in the LC example are the following:

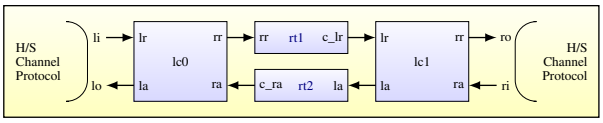


Fig. 4. Composition of Two Linear Controllers with RT Constraints.

```
SPEC AG EF lc1.rproc.state = s5;
SPEC AG EF lc1.lproc.state = s0;
```

B. RT Signal Ordering Properties

The correctness of a timed protocol is stated with PSL properties [9]. In the LC example, the following PSL property ensures that the imposed ordering of RT constraint lr+ $\mapsto$ rr+ $\prec$ lr- always holds in the protocol composition in Fig. 4. The sequence of c_lr within the { } braces (c_lr+) sequentially implies that rr+ should occur before c_lr−.

```
PSLSPEC always ({c_lr=0;c_lr=1}
((lc1.rproc.rr=0 & next(lc1.rproc.rr=1))before
(c_lr=1 & next(c_lr=0))));
```

C. Failure Reachability

We model timed protocols with semi-modular processes [10]. Upon receiving an input, a semi-modular process must emit an output before another input change occurs on the same input. Failures occur when an invalid timed state f (or bottom) can be reached in any system composed of timed protocols due to illegal inputs. This is illustrated for an inverter:

```
INV01 =  (in+).INVa1
INVa1 =  (in-).f     +  (out-).INVa0
INVa0 =  (in-).INV00
INV00 =  (in+).f     +  (out+).INV01
```

Failure reachability can be specified separately for each protocol. In our implementation a global boolean flag variable is maintained that is initially false and will be set true if any of the protocols reach a failure state. Global failure reachability can be verified with the single CTL property !EF failure=1.

V. CAD and Tool Flow

This CAD flow is implemented with a tool named TNSMVART (Translation for NuSMV Automated with RT). It verifies a Verilog design. The control structure of the design is checked for correctness after generating the NuSMV models as described in the previous section. Inputs to the flow include the Verilog, a file with relative timing constraints for each timed asynchronous protocol used in the Verilog design, mapping of the RT constraints onto design instances, and a file with a formal semi-modular representation of each boolean logic gate in the design library. The top level flow is shown in Fig. 5.

The Verilog is converted into the OpenAccess database using the verilog2oa program and OpenSourceLiberty [12], [13]. The RT constraints for each asynchronous protocol used in the design are mapped to all instances in the design as sdc constraints. TNSMVART reads the OpenAccess database

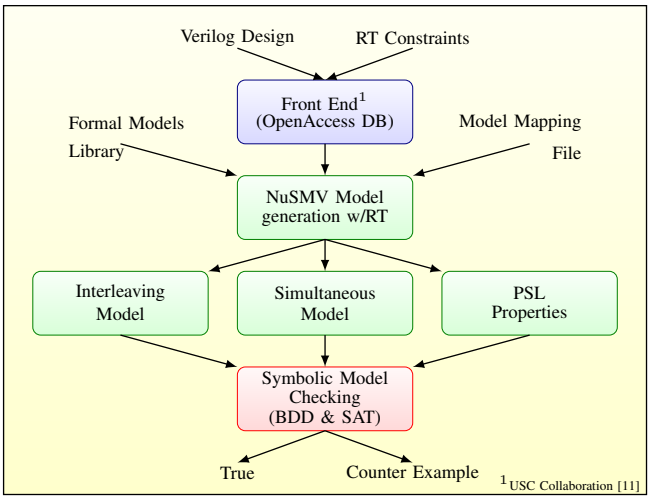


Fig. 5. TNSMVART Tool: Top Level Flow Diagram

containing the design and the RT constraints file with an option to specify transition counts for multicycle RT constraints. The data path is abstracted out of the design. The NuSMV models of the control structure of the design are created as described in Sec. III-B and III-C. The resulting models are mapped using either interleaving or simultaneous semantics. A correct protocol interface is created for the channels that interact with the restricted environment provided to the tool. Verilog library gates and modules are mapped into their equivalent semi-modular state graph representation (Sec. IV-C). Finally, a set of properties are generated to verify the correctness of the model as described in Sec. IV.

VI. Case Studies and Results

This tool flow has been applied to several examples of varying complexity. Some of these are presented in this section. The run times of the results are all reported running on a Dell workstation with Intel® Core^{TM} i7 processor, 2.67GHz and 3GB memory (free memory up to 600MB) machine. The NuSMV engine is executed with reachable state computation enabled with all other default configurations. The TNSMVART tool translates the structural Verilog files into the NuSMV formal modeling language with RT constraint modeling. Executions times for the translation were between 0.1 and 2 seconds for the case studies.

A. C-element

The C-element is a basic asynchronous circuit used as a merging element. A timed gate level design can be built using three two input NAND gates and one three input NAND gate. The behavior of each NAND gate is described as a semi-modular protocol similar to the inverter of Sec. IV-C. The following four relative timing constraints are required for this C-element design to operate without failure.

$$c+ \mapsto ac- \prec a- \qquad c+ \mapsto bc- \prec a-$$
$$c+ \mapsto ac- \prec b- \qquad c+ \mapsto bc- \prec b-$$

Interleaving and simultaneous models are composed that employ the constraints. The C-element model was verified with the NuSMV verification engine tool flow for deadlock, failure unreachability, point-of-divergence event state reachability, and a PSL property verifying the relative ordering property of an RT constraint.

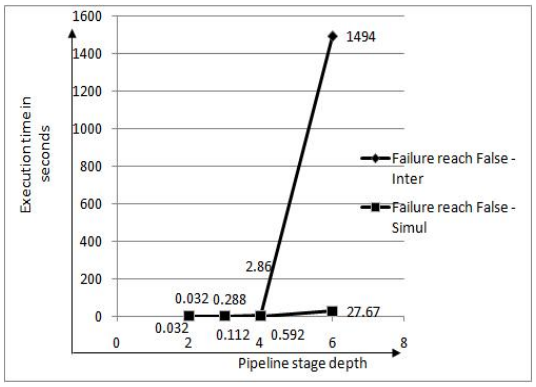


Fig. 6. LC Example: BDD Execution Times for Semimodular Failure reach Property being False of Interleaving and Simultaneous Models

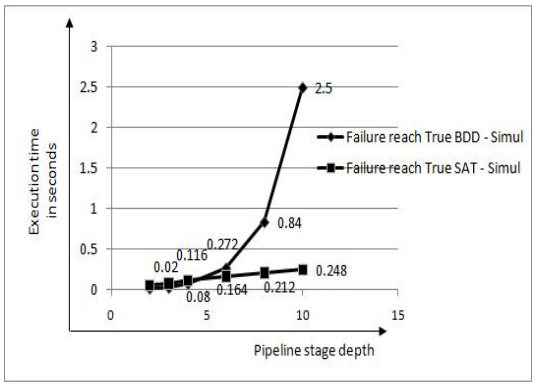


Fig. 7. LC Example: Execution Times for Semimodular Failure reach Property being True in BDD and SAT Methods with Simultaneous Models

B. Linear Controller

Pipelines of various depths were evaluated using the LC protocol specified in Fig. 2. The model for a two-deep pipeline is shown in Fig. 4. For these verification results the LC module employs the protocol specification rather than the implementation. Properties verified are similar to that of the C-element example. This *timed* protocol requires the following two RT constraints to operate correctly:

$$lr+ \mapsto rr+ \prec lr- \qquad ra- \mapsto la+ \prec ra+$$

Performance evaluation of LC pipelines is measured with a different number of pipeline stages, modeling types, properties, and symbolic methods. Fig. 6 shows the execution times in seconds for interleaving and simultaneous models for the failure reach property being false. Fig. 7 illustrates the comparison between BDD and SAT methods for finding a failure in the case of the simultaneous model when all RT constraints are not modeled. Fig. 8 shows that the execution times for failure reachability being false for linear pipelines up to 28 stages deep.

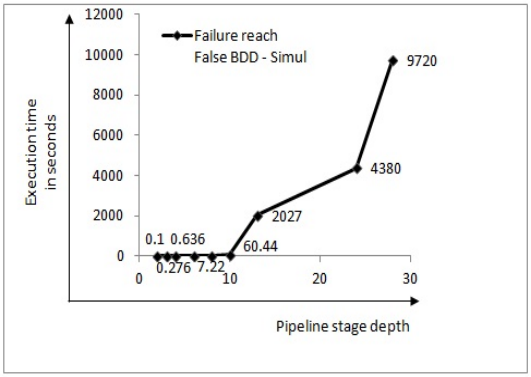


Fig. 8. LC Example: Execution Times for Semimodular Failure reach Property being False in BDD Method with Simultaneous Models

The simultaneous model is more efficient than the interleaving model. This is largely due to the lower state-space complexity and system diameter of speed-independent design compared to delay-insensitive models. Execution times for SAT-based methods are dependent on the bound length of the failures. SAT run times were generally slower when greater bound lengths are necessary to determine failure. For smaller bound lengths, failure was reachable faster compared to the BDD method.

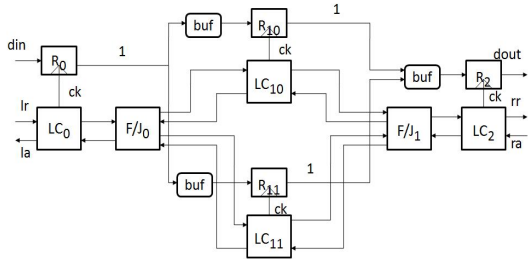


Fig. 9. Arithmetic Pipeline Example $x^2 + 2x$, data path abstracted out

C. Arithmetic Pipeline Example

The case study of Fig. 9 is a combination of the previous examples. Linear controllers are used for handshaking control. C-elements are used to implement broadcast Join/Fork elements. TNSMVART abstracts the data path function to a single latch and data bit, sufficient to represent the timing relationship between the data and control paths.

The tool generates and verifies similar properties for the BDD and SAT methods. The TNSMVART engine created the model in 0.215 seconds and verified failure unreachability in 3.63 seconds with RT constraints imposed in the control path.

D. Global STP (Self Timed Pipeline)

The next example is a challenging timed circuit protocol using signal pulses. The Global STP design is the double frequency arithmetic pipeline implemented in an Intel microprocessor using self-resetting domino gates [14]. The block diagram in Fig. 10 constitutes three global STP stages and two RESET stages along with the Latch Precharge and Inverter blocks for output control. Signal `ck` is the double frequency clock, and `dout` represents valid timing window of the result.

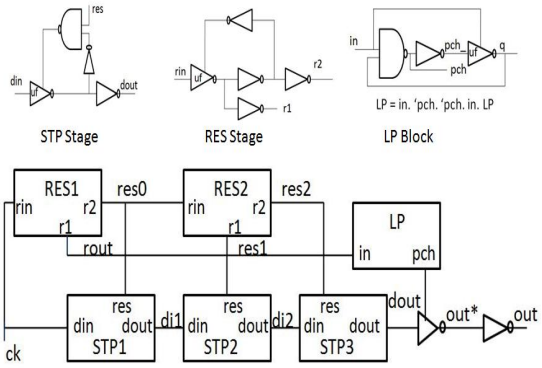


Fig. 10. Global STP: Top Level Block Diagram

The Verilog circuit design of each block is independently modeled and verified against their minimized specifications. The logic of these blocks consists of both domino and static gates. There are about 20 RT constraints for these designs.

The composed model consisting of two RES stages and three STP stages is verified. Unfortunately, the verification engine ran out of memory for the full composition shown in Fig. 10 due to increased modeling complexity with multicycle timing constraints. Resolving this with dynamic reordering and investigating counter examples for understanding the differences between the trace and bisimulation semantics is a topic for further research.

E. Fast Fourier Transform (FFT)

This verification methodology was also applied to verify the control plane of an asynchronous implementation of a high performance 64-point FFT [15]. The same timed protocols used throughout the paper were employed in this hierarchical wavelet design. We are able to verify the correct behavior by employing abstractions.

The verification is performed hierarchically. The control structure of the 4-point FFT consists of a 1:4:4:4:4:1 pipeline with "butterfly" data interconnect using fork/join elements. Failure unreachability is successfully verified in 59 minutes.

The FFT-16 is a 13 deep pipeline containing 8 FFT-4 blocks and 214 total linear controllers. It has an internal parallelism of 4, employing 4-way decimators and expanders. Based on proofs from the abstraction of parallel pipelines [16], the FFT-4 designs are substituted with observationally equivalent five deep linear pipelines. The FFT-16 control structure equivalent design block (1:4:16:1) with 22 LCs, Fork/Joins and relative timing constraints enforced executed for failure unreachability in 86 minutes. An abstracted equivalent 13-deep pipeline verifies for failure unreachability in 35 minutes with relative timing constraints.

The FFT-64 is a 4 way expander, a 4×16 crossbar (consisting of 16 4-way expanders and 4 16-way decimators), a 16-way decimator, and 23 LC blocks in a 5-deep pipeline. A proven abstracted equivalent of the 64-point FFT is the 21 deep linear pipeline. Based on the results in Sec. VI-B, the 24 deep linear pipeline executed for failure unreachability in 73 minutes with relative timing constraints.

TABLE I
ARTIST VS NUSMV - TRUE CASE

Tool	Stages 2	Stages 4	Stages 8	Stages 10
ARTIST	0.008 s	0.036 s	11.105 s	mem
NuSMV (BDD)	0.1 s	0.636 s	10.06 s	60.44 s

The ability to abstract complex parallel and series pipelined protocol structures into equivalent simple linear pipelines, and to prove their interactions at the interfaces with relative timing constraints, is essential to verify large protocol systems.

F. Comparison with ARTIST

One of the primary goals of this work is to extend the verification capability of timed asynchronous protocols to larger systems. Performance of this flow ported to NuSMV fares better than previous methods that support timed protocol verification.

The TNSMVART converted designs are compared to the custom relative timing formal verification tool (ARTIST) using non-symbolic verification that we developed in-house. Results for pipelines up to 10 deep are shown in Tab. I. The TNSMVART translated model of a 24-stage timed linear pipeline was successfully verified in the True case (i.e. failure unreachability) in 73 minutes using the BDD method. A 28 stage pipeline was verified in 162 minutes. Dynamic BDD reordering is used which enhances scalability. ARTIST runs out of memory after the 9th stage.

However, when failures exist in a design ARTIST often excels. For a 40 stage pipeline example, the first failure is found in 0.06 seconds. The TNSMVART translated models are faster without the dynamic reordering option for finding failures. There is no memory out problem until 27 stages both in BDD and SAT methods. The BDD method experienced a memory out problem for the 28^{th} stage. The SAT method was faster than BDD with an increase in number of compositions. This is the case because the bound length for failures was less than 10. With increased bound lengths, the SAT based bounded model checking time to find failures increase substantially. BDD based method performs complete reachability and verifies properties being true. SAT based method finds counter examples for a given bound length quickly.

VII. CONCLUSION

A symbolic model checking CAD tool flow for verifying systems of timed asynchronous protocols with BDD and SAT methods is presented. This is the first engine to support relative timing constraints integrated with general asynchronous sequential protocol verification in a symbolic verification engine enhancing the scalability. For the examples in this paper, TNSMVART run times were between 0.1 and 2 seconds.

Timing failures in a protocol are manifest by the occurrence of unaccepting inputs in a semi-modular representation of a system. A method has been developed to integrate relative timing constraints and their multicycle variants into the protocols in order to prevent timing failures. This method does not modify the behavior of the initial protocols composed in a system. The approach has been applied to both interleaving and simultaneous execution models that represent delay-insensitive and speed-independent families of protocols. In general, the simultaneous models show better performance.

A CAD tool named TNSMVART was implemented to map a Verilog level design into the industry-strength NuSMV verification engine. For the case studies, in the simultaneous model, SAT methods tend to find a counter example in less time than BDD methods.

This work demonstrates improved verification capability over current approaches. It also opens the door to future research as abstraction methods are required to verify large designs implemented with timed protocols. Developing and combining protocol concurrency abstraction methods into TNSMVART will be essential to automatically verify large systems. The tool has been exclusively applied to timed clocked and asynchronous protocols and we anticipate that this work will apply equally well to software and other higher level protocols.

REFERENCES

[1] D. L. Dill, *Trace Theory for Automatic Hierarchical Verification of Speed-Independent Circuits*, ser. ACM Distinguished Dissertations. MIT Press, 1989.

[2] K. L. McMillan, *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[3] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative Timing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 1, no. 11, pp. 129–140, Feb. 2003.

[4] Y. Xu and K. S. Stevens, "Automatic Synthesis of Computation Interference Constraints for Relative Timing Verification," in *26th International Conference on Computer Design*. IEEE, October 2009, pp. 16–22.

[5] R. Cavada, A. Cimatti, C. A. Jochim, G. Keighren, E. Olivetti, M. Pistore, M. Roveri, and A. Tchaltsev, "Nusmv 2.4 user manual," http://nusmv.irst.itc.it.

[6] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, pp. 541–580, Apr. 1989.

[7] R. Milner, *Communication and Concurrency*, ser. Computer Science. Prentice Hall, 1989.

[8] V. Vakilotojar and P. A. Beerel, "RTL verification of timed asynchronous and heterogeneous systems using symbolic model checking," *Integration, the VLSI Journal.*, vol. 24, no. 1, pp. 19–35, December 1997.

[9] Accellera, "PSL Reference Manual," http://www.eda.org/vfv/docs/PSL-v1.1.pdf.

[10] D. E. Muller and W. S. Bartky, "A theory of asynchronous circuits," in *Proceedings of an International Symposium on the Theory of Switching*. Harvard University Press, Apr. 1959, pp. 204–243.

[11] E. Quist, P. Beerel, and K. S. Stevens, "Enhanced SDC support for relative timing designs," in *Digital Automation Conference, User Track Poster*, Jul. 2009.

[12] Si2, "OpenAccess," http://www.si2.org/openeda.si2.org/.

[13] Liberty, "OpenSourceLiberty," http://www.opensourceliberty.org/.

[14] G. Hinton, M. Upton, D. Sager, D. Boggs, D. M. Carmean, P. Roussel, T. I. Chappell, T. D. Fletcher, M. S. Milshtein, M. Sprague, S. Samaan, and R. Murray, "A 0.18 CMOS IA-32 processor with a 4-GHz integer execution unit," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 11, pp. 1617–1627, November 2001.

[15] W. Lee, V. S. Vij, A. R. Thatcher, and K. S. Stevens, "Design of Low Energy, High Performance Synchronous and Asynchronous 64-Point FFT," in *Design, Automation and Test in Europe (DATE)*. IEEE, Mar 2013, pp. 242–247.

[16] G. Birtwistle, "Control states in asynchronous pipelines," in *Asynchronous Interfaces: Tools, Techniques, and Implementations"*, A. Yakovlev and R. Nouta, Eds., July 2000, pp. 45–55.