HDL Modeling for Analysis and Optimization of Asynchronous Controllers

Jung-Lin Yang, Erik Brunvand, and Sung-Min Lin

Institute of Electronic Engineering
Southern Taiwan University of Technology, Tainan County 71005, Taiwan (R.O.C.)
School of Computing
University of Utah, Salt Lake City, UT 84112, USA
Graduate School of Information, Production, and Systems
Waseda University, Fukuoka 808-0135 Japan
jyang@mail.stut.edu.tw, elb@cs.utah.edu, and cedric@ruri.waseda.jp

ABSTRACT

We propose a simulation-based technique for analysis and optimization of extended burst-mode (XBM) asynchronous controllers. In asynchronous controllers of this sort, timing information on control signals is significant both for performance enhancement and timing validation. Timing information, specifically information about relative signal arrival times, helps us improve the controller's response time and to detect delay faults within controllers in the early synthesis stage. If the timing information of the controller's environment is also known, we can use this information to identify fundamental mode violations. Our approach uses stochastic simulation of HDL programs derived from the original XBM specifications to gather information about signal timing and long-term transition probabilities.

1. INTRODUCTION

The most common technique for optimizing the low level details of asynchronous controllers using BM/XBM design it to compute the so-called long-term transition-probabilities of the control signals [1], [2]. These probabilities are used to optimize the circuits to take advantage, for example, of late- or early-arriving signals to the controller. These methods are very useful, but if they are used exclusively they may ignore other factors. For example, a higher transition-probability burst may not always be the one that contributes most to the controller delay unless the burst is clearly identified as the one that will result in the greatest performance enhancement.

Although the transition-probability provides information on which transition is more critical than the others, it only provides the frequency of the transitions and allows designers to compare them without other valuable information. In the situation where long-term transition-probabilities are exclusively used in the analysis, the optimization searching process can easily get stuck at a local minimum, and the searching process may never have a chance to obtain alternative solutions.

In addition, when the original specification of the controller is transcribed into the coded VHDL models using our techniques, we are able to use the HDL simulator to validate and determine the implementation over all expected BM/XBM transition behaviors by running stochastic simulations. One noteworthy feature of our proposed HDL modeling technique is through simulation we are able to detect and correct fundamental mode violations caused by rapid feedback state variables.

2. BM/XBM TRANSITION PATTERNS

Our simulation-based approach does involve the possibility of producing non-optimal circuits, however, which is very different from the analytical timed circuit approach [3]. The proposed method produced a nearly optimal implementation when the provided timing information is close enough to the actual behavior of the asynchronous environment. Unpredictable factors such as unexpected interconnection delay, large process variations, or simply wildly incorrect timing information are seen, the mapped circuits may still operate correctly, but may require significant timing safety margins.

BM/XBM is a fundamental mode sequential state machine model introduced for specifying asynchronous controllers. By definition, the changing pattern of the edge signals in a burst (a specified set of signal changes that cause a state change) can be arbitrary [6]. Thus, a BM/XBM specification provides no further information about the triggering behavior inside a transitional burst. The changing pattern of the transitional burst includes the signal arrival sequence and its correlative arrival delays. Knowing the exact changing pattern helps synthesize a better circuit by adjusting gate topologies to account for

0-7803-9266-3/05/$20.00

different signal arrival times. Conflicting constraints may make an optimal circuit impossible, but understanding the most likely changing pattern still helps synthesize a better circuit, especially the reactive nature of these asynchronous controllers.

The primary objective of our HDL modeling technique is to understand the relative time of signal changes in a burst. The more detail about the target environment that is given; the more accurate a statistical pattern result is obtainable by the simulation. Our technique uses two methods for providing environmental information. One is to provide signal changing upper/lower delay bounds for all input/output signals to the subject controller, (technique also used by ATACS [4]). The second method is to model the entire target environment in another HDL program using known datapath response delays.

3. HDL MODELING

The basic techniques for developing controllers using XBM techniques are described in [6]. Sample circuits demonstrate various ways that these controllers operate. Branching in a controller, for example, can be accomplished by using signal levels, or by disjointed signal transitions. Our tool is capable of transcribing all these transition types and turning them into corresponding HDL models that implement the given BM/XBM specifications.

Our tool generates two versions of VHDL models for the given BM/XBM specification. One represents the burst-mode controller itself, and the other is the mirrored controller. The mirrored controller is a generic (most general) environment that interacts with the target controller when a more specific model environment is not given. The generic model implements response behavior with timing constraints when the delay bounds are specified. Once the VHDL models are generated, VHDL simulator that is compatible to the 93' VHDL standard is attainable for gathering the timing information of each transition [8].

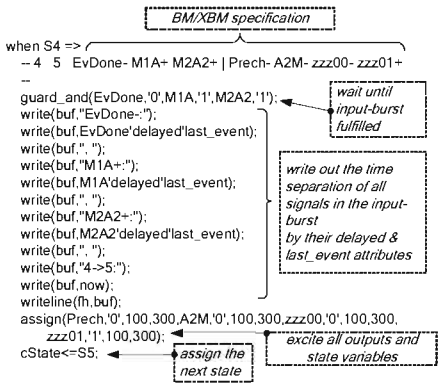


Figure 1. Translated Simple BM/XBM Transition

3.1 Case 1: Simple BM/XBM Transition

In the following sections, the translation procedure is described. The translated VHDL models use a channel package [4] that models guarded event signals and variable delays to guard the state transition behavior and assigned outputs. When a transition condition is fulfilled, we use the delayed and last-event attributes of the excited signals to extract the relative time separation of the transition that is fired [8].

Figure 1 demonstrates a simple BM/XBM transition and its corresponding VHDL code. The VHDL models can simulate with or without a user-specified environment. When the environment is not given, the generic mirrored environment is used for simulation. During the simulation, whenever a state is initiated, the write statement records the related timing information to a log file. Then, we can import the simulation data in a database for off-line analysis.

3.2. Branching by Level Signals

Branching based on level signals is more complicated. The translated VHDL code segment in Figure 2 shows how to turn multiple outgoing BM/XBM transitions into the corresponding VHDL code. There are three different types of signals in the input-burst: the level signal *C*, the edge signal *EvDone*, and the directed-don't-care signals *M1A* and *M2A2*. If we interpret these two transitions in English, which would be "when the *EvDone*↑ arrives, go to state 6 if *C* is '0'; otherwise, go to state 9". Of course, the outputs and state variables must be assigned to their new values accordingly.

Notice that the M1A and M2A2 signals did not appear in the translated code. This is because the firing transition leading to states 6 and state 9 has nothing to do with both of the signals. The XBM specification says that they can be changed at most once, but they are not required to change in these transitions [7]. Therefore, directed-don't-care signals are simply ignored until they are forced to change.

```
when S5 =>
  -- 5  6  [C-] EvDone+ M1A* M2A2* |
  --       Prech+ seldx+ selym2+
  -- 5  9  [C+] EvDone+ M1A* M2A2* |
  --       Prech+ selym2+ zzz00+ zzz01-
  --
  guard(EvDone,'1');
  if C='0' then
    ... write out timing information
    assign(Prech,'1',100,300,
         seldx,'1',100,300,
         selym2,'1',100,300);
    cState<=S6;
  elsif C='1' then
    ... write out timing information
    assign(Prech,'1',100,300,
         selym2,'1',100,300,
         zzz00,'1',100,300,
         zzz01,'0',100,300);
    cState<=S9;
  end if;
```

Figure 2. Translated BM/XBM Branching by Level Signals

3.3. Branching by Disjointed Edge Signals

Figure 3 shows another branch controlled by a disjoint set of edge signals and illustrates how a more complicated BM/XBM specification can be translated into the corresponding VHDL code.

```
when S5 =>
  -- 5  6  cmackn- memhit+ |
  --       we0n+ we1n+ we2n+ we3n+ wetagn+
  -- 5  9  cmackn- memmiss+ |
  --       we0n+ we1n+ we2n+ we3n+ wetagn+ tagvalid+
  --
guard(cmackn,'0');
guard_or(memmiss,'1',memhit,'1');
if memmiss='1' then
   ... write timing information
   assign(we0n,'1',100,300,
        we1n,'1',100,300, we2n,'1',100,300,
        we3n,'1',100,300, wetagn,'1',100,300,
        tagvalid,'1',100,300);
   cState<=S9;
elsif memhit='1' then
   ... write timing information
   assign(we0n,'1',100,300,
        we1n,'1',100,300, we2n,'1',100,300,
        we3n,'1',100,300,  wetagn,'1',100,300);
   cState<=S6;
end if;
```

guard common signal(s) : common edge signals of the all outgoing transitions

guard disjoined signals: any of signal fired indicates a smaller set of outgoing transitions are selected

Figure 3. Translated BM/XBM Branching by Edge Signals

The assign statements are always the easiest part of the translation because they directly point to the next-state result. The first guard statement is used to filter out edge signals that are common to all outgoing transitions. The common signals can be removed from all transitions after this guard statement. There must, however, be at least one edge signal left in every transition or the transition will violate the BM/XBM "distinguishability" constraint [5].

We should have a smaller set of outgoing transitions after each *guard_and* and *guard_or* statement. If the initial transitions are as simple as shown in Figure 1, the whole outgoing transition set can be translated using the procedure described. When a more complicated case is encountered, the translation procedure is applied recursively to the transitions until all given subset transitions can be interpreted as a single-level *if-then-else* statement.

```
def xbm2vhdl_pt(trans,L=0,var=None):
        # trans: outgoing transitions
        # L    : recursion level
        #
        # pt   : global variable for storing parsed information
        global pt
        stm=[L]
        stm.append(trans)
        # having exact 1 outgoing transition
        if len(trans)==1:
                stm.append(trans[0])
                pt.append(stm)
                return
        common = trans.intersect_edges()
        disjoin = trans.disjoin_edges()
        stm.append(common)
        stm.append(disjoin)
        pt.append(stm)
        # grouping transitions based on the next edge signal fired
        gps={}
        for t in trans:
                map(t.remove,common)
                for s in disjoin:
                        if t.count(s)>0:
                                gps[s].append(t)
    # differentiate transitions by next edge signal fired
        for ts in gps.items():
                xbm2vhdl_pt(ts[1],L+1,ts[0])
```

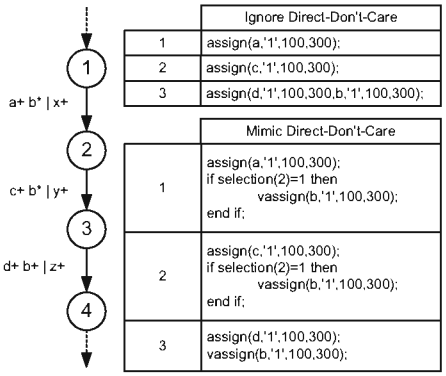
Figure 4. XBM Transition Parsing Algorithm

3.4. Translation Procedure

Our XBM to VHDL translation procedure involves two steps: parsing the XBM specification, and generating the corresponding VHDL code based on the parsed information. Figure 4 shows a simplified version of the parsing procedure. This algorithm differentiates given transitions recursively until the targeting transitions can be resolved. Then, the second procedure uses the parsed information to generate VHDL codes accordingly. The mapping between the parsed information and the VHDL is straightforward and not shown here.

3.5. Dummy Testing Environment

Generating the generic testing environment (the mirrored BM/XBM controller) is simpler than the controller translation procedure. The multiple-outgoing transition case is handled in a two-step procedure. First, a random number generator *selection(...)* and *if-elsif* statements are used to decide which transition is going to be fired. Then, the rule for the single outgoing transition is used to translate each transition into the body of the *if-elsif* statement. If the branching probability is not given, every transition gets an equal chance to be fired. Otherwise, the given branch distribution is normalized.



	Ignore Direct-Don't-Care
1	assign(a,'1',100,300);
2	assign(c,'1',100,300);
3	assign(d,'1',100,300,b,'1',100,300);

	Mimic Direct-Don't-Care

1	assign(a,'1',100,300);
	if selection(2)=1 then
	vassign(b,'1',100,300);
	end if;

2	assign(c,'1',100,300);
	if selection(2)=1 then
	vassign(b,'1',100,300);
	end if;

3	assign(d,'1',100,300); vassign(b,'1',100,300);

Figure 5. Handling Directed-Don't-Care

3.6. Directed-Don't-Care Signals

Directed-don't-care signals can be reduced to terminating signals as we did in translating BM/XBM controllers. Or, we can use selection(...) and vassign(...) functions to mimic the actual directed-don't-care behavior [5]. The second choice requires more attention when generating assign statements for the edge signals in an input burst. If the edge signal is directly followed by its directed-don't-care transition, the edge signal must be assigned by the vassign(...) function. Otherwise, the simulator might be stalled by a vacuous assignment when the signal is already assigned in the earlier transition [4]. In Figure 5, we show how to translate directed-don't-care signals using both methods.

4. RESULT AND ANALYSIS

Our HDL models output timing information to a log whenever a transition burst takes place. The recorded timing information can be compiled to show the relative arrival time of all signals within a burst and then export it to a database (see Figure 6). The simulation result contains accurate long-term transition-probability based on the targeted testing environment when the sampling time is sufficiently long. The translated controller can also be simulated with its generic environment when the target environment is not present or is difficult to model. In such cases, the simulated result returns the long-term transition-probability based on the initial branch probability.

Figure 6 shows partial VHDL simulation data stored in the database for the ALU2 XBM specification [6]. The resultant data is read with "time" in columns, transition state cstate, nstate, EvDone, and signals M1A, M2A2 in rows. The first column "time" is the absolute simulation time of each recorded transition pattern. If a signal value is "0," the signal is the latest arriving signal for the current transition. If a signal value is "-1," the signal does not change in the transition; it is then called the context signal in the transition. Also, if a signal value is greater than 0, the value represents the relative separation time between that occurrence of the signal and the latest arriving signal.

We can construct some formulas to estimate the possible performance gain with the extracted relative arrival timing information. We define the weight of every transition based on the product of its long-term transition-probability and the statistical-relative arrival timing information. Also, we define the "maximal performance enhancement" as the transition weights instead of relying exclusively on the long-term transition-probability. We could apply the extracted relative arrival timing information as parameters to generate the performance gain functions for various logic families.

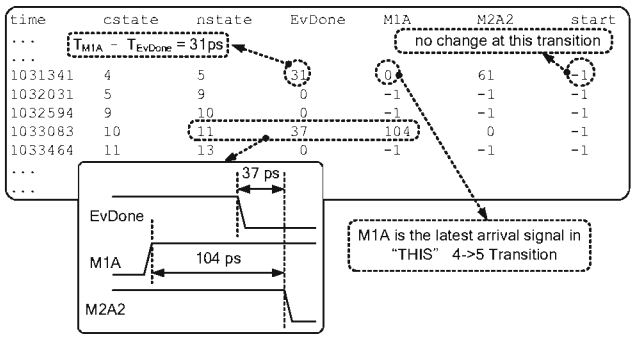


Figure 6. HDL Simulation Result – ALU2

Context signals must be stabilized before any edge signal in the current transition is excited. Thus, context signals always arrive earlier than the edge terminating signals in the current transition. However, the priority in signal transitions only allows separating the context variables from the triggering edge signals. The related weight and significance of each context variable is still uncertain. A simple algorithm was developed to use the back-annotated BM/XBM specification to evaluate which state has every context variable set and attaches the relative worst-case timestamp to it.

In summary of the analysis above, we have obtained long-term transition-probability, relative arrival times of triggering signals, and the timestamp of all context signals. Long-term transition-probability reveals the rate of every transition visited [1]. Relative arrival time gives the parameter for evaluating the average gain that a transition is most likely to provide. A context signal's timestamp tells how far in advance the signal is set prior to the current transition. The combination of these three factors with pre-calibrated performance gain look-up-table or analytical function serves different needs for optimization.

5. CONCLUSION

The proposed translation technique for converting BM/XBM to VHDL models and deriving timing information through simulation is useful in analysis and optimization of the controller. The translated HDL model generates significant timing information and is allows for transition pattern analysis and representation of an ideal behavior of the specified XBM controller. We have systematically elaborated the translation of the BM/XBM to VHDL and automated it as a tool for technology mapping.

6. REFERENCES

[1] P. A. Beerel, K. Y. Yun, and W. C. Chou, "Optimizing average-case delay in technology mapping of burst-mode circuits," in *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 1996.

[2] K. W. James and K. Y. Yun, "Average-case optimized transistor-level technology mapping of extended burst-mode circuits," in *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998.

[3] C. J. Myers and T. H.-Y Meng, "Synthesis of timed asynchronous circuits," in *IEEE Transactions on VLSI Systems*, 1(2), June 1993, pp. 106-119.

[4] C. J. Myers, *Asynchronous Circuit Design*, NJ: John Wiley and Sons, July 2001.

[5] K. Y. Yun, D. L. Dill, and S. M. Nowick, "Synthesis of 3D asynchronous state machines," in *Proc. International Conf. Computer Design (ICCD), IEEE Computer Society Press*, October 1992, pp. 346-350.

[6] K. Y. Yun, "*Synthesis of asynchronous controllers for heterogeneous systems*," Ph.D. dissertation, Stanford University, CA, August 1994.

[7] K. Y. Yun and D. L. Dill, "Automatic synthesis of extended burst-mode circuits: part I (specification and hazard-free implementation)," in *IEEE Transactions on Computer-Aided Design*, 18(2), February 1999, pp. 101-117.

[8] Institute of Electrical and Electronics Engineers, "*IEEE Standard VHDL Language Reference Manual*, Std 1076-1993," 345 East 47th Street, NY, 1993.