Design and Optimization of Multi-clocked Embedded Systems using Formal Technique

Yu Jiang[2],Zonghui Li[4],Hehua Zhang[1],Yangdong Deng[4],Xiaoyu Song[3],Ming Gu[1],Jiaguang Sun[1]
School of Software, Tsinghua University,TNLIST, KLISS, Beijing, China[1]
Department of Computer Science and Technology, Tsinghua University, TNLIST, KLISS, Beijing, China[2]
Department. ECE, Portland State University, Oregon, USA.[3]
Institute of Microelectronics, Tsinghua University, Beijing, China.[4]

ABSTRACT

Today's system-on-chip and distributed systems are commonly equipped with multiple clocks. The key challenge in designing such systems is that heterogenous control-oriented and data-oriented behaviors within one clock domain, and asynchronous communications between two clock domains have to be captured and evaluated in a single framework. In this paper, we propose to use timed automata and synchronous dataflow to capture the dynamic behaviors of multi-clock embedded systems. A timed automata and synchronous dataflow based modeling and analyzing framework is constructed to evaluate and optimize the performance of multi-clock embedded systems. Data-oriented behaviors are captured by synchronous dataflow, while synchronous control-oriented behaviors are captured by timed automata, and inter clock-domain asynchronous communication can be modeled in an interface timed automaton or a synchronous data flow module with the CSP mechanism. The behaviors of synchronous dataflow are interpreted by some equivalent timed automata to maintain the semantic consistency of the mixed model. Then, various functional properties can be simulated and verified within the framework. We apply this framework in the design process of a sub-system that is used in real world subway communication control system.

Categories and Subject Descriptors

B.3.3 [**Performance Analysis and Design Aids**]: Formal models, Simulation

General Terms

Formal models, Simulation

Keywords

multi-clock, embedded system, data-oriented behavior, control-oriented behavior, timed automata, synchronous dataflow

1. INTRODUCTION

Embedded systems are being widely used in all kinds of applications, and are traditionally designed and optimized using synchronous languages with single clock. Such an assumption of global synchronization greatly helps reduce the complexity of the design. The class of synchronous languages contains mainly Esterel [5], Lustre [7], Signal [11] and Statecharts [8]. Those languages and the corresponding tools are good for compact single-clocked hardware and software design, including modeling, simulation, verification, and synthesis. The Esterel and Statecharts are suitable for specifying control-oriented systems. The Lustre and Signal are good for specifying data-dominated systems. The control-oriented systems control large amounts of decision logic that has to quickly produce outputs in response to input events, while in the data-dominated systems, intensive computations have to be performed on samples that usually arrive in regular intervals. Very often, an embedded system contains both data-oriented and control-oriented parts. For example, the cell-phone must contain the control-oriented network communication protocols running on the processor and the data-dominated algorithms for dealing with the voice signal. Furthermore, today's embedded systems are increasingly adopting multi clock solution due to the low-power requirement and the pervasive usage of IPs from different vendors. Hence, there has been a recent surge in demand for methods to design multi-clocked embedded systems.

In this paper, we present a timed automata [1] and synchronous dataflow [2, 12] based design framework to address the above-mentioned problem in modeling and validating the heterogenous behaviors of multi-clocked embedded systems. In our framework, a system is modeled as a combination network of timed automata and synchronous dataflow, which is a collection of local synchronous domains and asynchronous communications. The main novelties of our framework are three fold: (1) First, with the help of different clock remapping based guards on each transition of automata, each local system component can be modeled as a purely synchronous node analogous to a synchronous reactive program in a language like Esterel. (2) Second, with the help of shared variables, special synchronize input/output actions and parallel composition operator, the inter clock-domain asynchronous communication through hand-shake protocol can be modeled in an interface timed automaton without clock or a synchronous dataflow module. (3) Third, the control-oriented parts can be modeled by timed automata, the data-oriented parts can be modeled by synchronous data

ESEC/FSE'13, August 18–26, 2013, Saint Petersburg, Russia
Copyright 2013 ACM 978-1-4503-2237-9/13/08...$15.00
http://dx.doi.org/10.1145/2491411.2494575

flow, and the behaviors of each synchronous dataflow module are represented as equivalent timed automata to maintain the semantic consistency of the mixed model. We apply our framework to the design of a real world subway system and find a safety-critical bug caused by out of sync in the original design that cannot be detected by traditional techniques.

2. RELATED WORK

A large body of work has been dedicated to modeling and validation of multi-clocked systems. In the literature, the formal language based approach (e.g., CRP [3], MC-Esterel [4], SHIM [6]) is appealing because it provides a unified basis for formal analysis to achieve expected correctness. The CRP language combines the synchronous reactive model of Esterel [10] with the asynchronous coupling of CSP [9] to offer a mathematically elegant framework. Locally synchronous Esterel modules communicate through rendezvous channels. The problem is that it is hard to support data-driven operations and rendezvous protocol through asynchronous coordinators. Its variants such as CRSM, ECRSM [13] have similar properties and limitations. The MC-Esterel language was specifically developed for the design of multi-clocked digital systems. The designer is responsible for creating communication mechanisms among different clock domains. Every Esterel module needs an explicit clock and a designer has to construct low-level synchronizers to guarantee the synchronization among these modules. While MC-Esterel provides a powerful mechanism for modeling asynchronous and multi-rate systems, the main problem is that a designer has to work at a relatively low abstraction level and the productivity is limited. In addition, its support for data-driven operations is quite limited due to its reliance on Esterel.

3. PROPOSED MODELING AND VALIDATION FRAMEWORK

To model multi-clock embedded systems with both data-oriented behaviors and control oriented behaviors, a set of timed automata and synchronous dataflow modules are composed into a network over a set of clocks and actions with parallel composition operators. This kind of mixed data-oriented, control-oriented, and multi-clock domains compositions make the proposed model more close to the real implementation. In the design process, we can validate the design, compare different design models, avoid some wrong solutions that may lead to rework, and choose the best solution. In the implementation process, we can abstract the model from the implemented system, and apply formal verification techniques to validate whether the system meets the requirements or not. The overall framework is depicted in Figure 1.

It is clear in the above framework that all the control-oriented parts are modeled as timed automata, which are connected in parallel. Each automaton is equipped with an extra synchronous clock to control the local behaviors. When a time based guard becomes true, the automaton will make a transition. Although the increasing speed of all clocks are synchronous, we provide mechanisms for different time based guards to support multiple clock domains. The asynchronous communication between two timed automata is realized by rendezvous CSP style. All the data-oriented parts are modeled as synchronous dataflow. The actors represent the data related computations, the firing rules specify the number of tokens that the actor consumes and produces, and the schedule defines the firing sequences of actors in a single iteration. Similar to the parallel connection of timed automata, we need to define the parallel composition operator to connect a timed automaton with a synchronous dataflow module. The synchronous dataflow module has to communicate with the timed automata to accomplish two functions: (1) The first is reading tokens from the connected automata through the input channel to prepare for the firing of actors. (2) The second is sending tokens produced by the previous firing of actors to the connected automata through the output channel. The local synchronize modeling and communication between automata and synchronous dataflow are described below.

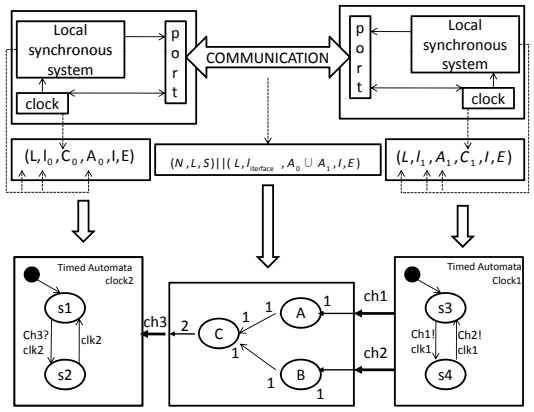

Figure 1: Modeling framework for multi-clock embedded systems with heterogenous behaviors.

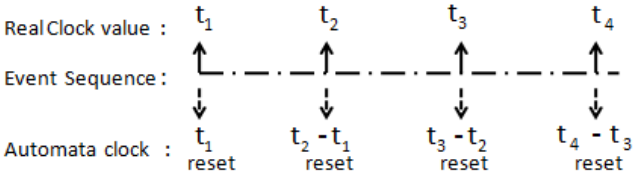

Figure 2: The mapping mechanism from the real clock value to the action of local clock in the timed automata to ensure synchronous reaction behavior.

Local synchronization modeling: Each component of local synchronization system has two kinds of reactions: inner reactions within a component and communication reactions cross two components. Both of them are controlled by its own clock. If the component is modeled as a timed automaton, the inner reaction can be described as regular transition edge with update action (A, E_{inner}). The communication reaction can be described as a transition edge with special synchronize action (A, E_{commu}) on channel n. The input action $n?$ represents receiving an event from the channel n sending by the other component, while the output action $n!$ represents sending an event on the channel n. The control clock of the real system component is modeled by a clock variable (C). Then, we add an extra guard and an extra update action on each transition edge $(A, E_{inner} \bigcup E_{commu}, G)$. The guards are defined on the period and interval of the real

control clock. The update action is to reset the clock variable. The mapping mechanism from the real control clock value to the behavior in the local timed automata clock is described in Figure 2. Because a transition represented by an edge can be triggered when the clock values satisfy this guard, the mechanism that each reaction will be triggered at the upper edge of the clock are captured in our timed automata. If the component is modeled as a synchronous dataflow, the inner reaction can be described as regular firings of the actors with the consumed and produced tokens (N, L_{inner}). The communication reaction can be described as regular link (L_{commu}) with special synchronization actions on channel n. Because the model is executed in a periodic fashion, we can add a period for each iteration similar to the run-time implementation of Ptolemy, regardless of each firing inside an iteration. This period will be interpreted by the clock variable of the equivalent timed automata in the semantic definition part.

Asynchronous communication modeling: The hand-shake asynchronous communication between two local system components M_1 and M_2 can be abstracted as an interface timed automaton or a synchronous dataflow module M_3. The two system components will transfer control message and data through bus to each other. If M_3 is a timed automaton, we can use some empty state transitions in M_3 to model the stochastic factors such as packet loss. When M_1 wants to send some data to M_2, it will send the data onto the bus and produce the special synchronization signal send_D!. The interface automata will be synchronized by receiving the signal send_D?, and will produce the special signal rec_D! with a transmission delay. Then, M_2 can be synchronized by receiving the signal rec_D?. If the data packet is lost, the M_3 will switch back to the initial state without sending the signal rec_D!. The control message transfer is similar to the data message. With the interface automata M_3 consisting of transitions that do not depend on clock 1 and clock 2, the asynchronous communication mechanism is captured directly. If M_3 is a synchronous dataflow module, it will also read data from M_1 through the read action send_D?, and send the produced token during the firing of actors through the write action rec_D!. If $M_1(M_3)$ attempts to write when $M_3(M_2)$ is not ready, $M_1(M_3)$ will be blocked. If $M_3(M_2)$ attempts to read when $M_1(M_3)$ is not ready, $M_3(M_2)$ will be blocked. They can communicate with each other when both sides are in the corresponding states. If not, the communication will be blocked.

Semantic of the proposed model: In the original timed automata, all parallel connected automata are combined together to get a single flat finite state machine, whose behavior is equivalent to the original modules. In order to integrate synchronous dataflow, we translate the synchronous dataflow into an equivalent timed automata. The translation procedure is modular. Each atomic element is represented by a timed automaton with channels to read tokens and output tokens. Additionally, a timed automaton may contain internal variables and functions, depending on the computation carried by the corresponding actor. The three timed automata presented in Figure 3 show three basic elements in the synchronous dataflow. The first is for the link between two actors and the link at the margin communicating with another module. It will read the tokens from the producer and pass it to the consumer without any computation. The second is for the actors which consume one token (x) and produce a token (y), with the function $(func())$ to finish the computation of the actor. The third is for the actor that consume two tokens (x, y) from two channels (c_1, c_2) and produce a single token (z). We use two local variables to denote the status of c_1 and c_2. When both of them have tokens, the transition will take, and the output tokens will be produced after the computation function. Other atomic elements connecting with arbitrarily multiple channels can be translated in the similar way of the third example. After all elements being translated, those timed automata will be connected in parallel. A synchronous dataflow module will be translated into a network of timed automata, whose behavior is the same as the original module.

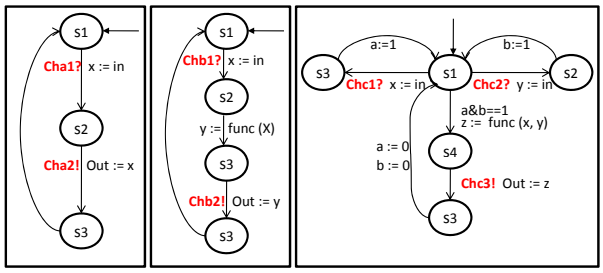

Figure 3: The translation from the synchronous dataflow to the timed automata.

4. EVALUATION EXAMPLE

We conduct some experiments on a train communication control system that is used in real world subway systems to describe how our framework enables the design of multi-clock embedded systems. The train communication control system is a safety-critical embedded system and is implemented as a network card as shown in the left side of the Figure 5. There are two processors on the card: the FPGA processor (big white cycle on the figure), and the ARM processor (small white cycle on the figure). The two components are controlled by their own clock, and communicate with each other through data bus in asynchronous manner. Some main time-critical functions are implemented by VHDL module running on the FPGA processor, and the and other functions are implemented by C code running on the ARM processor. When we embed a card into a vehicle, we need to initialize the card with the device configuration information. The ARM component will pass those data to the FPGA component. The data transfer is controlled by the GPIO of the ARM component. It will send a pulse at the frequency of 10MHZ, and then put 16 bit data on the bus. When the FPGA component samples the pulse from ARM, it needs 58 pulses to finish all the data transfer.

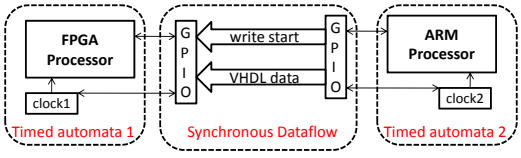

Figure 4: An abstraction model of the function that initialize the card with the device configure information. ARM will pass those data to the FPGA.

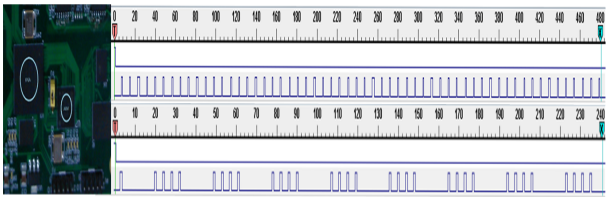

Figure 5: The left side of the figure is our implemented experiment platform of the control system. The right side are the results run-time comparison for different design, 58, 33 pulses sampled for 24 MHZ, 12 MHZ, respectively.

Because the clock frequency of GPIO on ARM is 10MHZ, the clock frequency of FPGA should be higher than 10MHZ to sample all pulses sent by GPIO on ARM. According to the static analysis, the clock frequency of FPGA is set as 12MHZ to meet the power constraint and hardware constraint in the original design. The prototype implemented by SystemC is simulated correctly. But the corresponding implemented VHDL code on FPGA based on this clock frequency fails to finish the initialization process at times. This is a very interesting phenomenon that motivates our work.

We build a model of this function which is presented in Figure 4 based on our framework. We build the mixed timed automata and synchronous dataflow based on the behavior model of the two components, the asynchronous communication of GPIO, and the two clock frequencies of the processors. Then, we performed formal verification on the constructed module, and find that the property that every pulse should be sampled correctly is violated. We use the random simulation mechanism provided in UPPAAL [1], and find a counter-example after 200 initialization times. When the clock of FPGA starts later than the ARM component for about half of the period, it will loss some pulses. We simulate the original code on the implemented platform again and set the running environment according to the counter-example, the waveform get by chipscope demonstrates that the FPGA component only obtain 33 pulses because of the out of sync. This bug is not easy to find because the occurrence rate is low and it is hard to trigger. We need to reduce the corresponding clock period of the guard of the timed automata. The property is satisfied when the frequency of FPGA is set as 24MHZ. The waveform get by chipscope for the new design is also demonstrated on the right side of the Figure 5. We can find the number of pulses is right. With the help of formal modeling, we make a contribution to avoid potential error of the subway system.

5. CONCLUSION

In this paper, we present the timed automata and synchronous dataflow based design framework for modeling and validating the dynamic behaviors of multi-clock embedded systems. Each local component of the system is modeled as a timed automata with a local synchronous control clock or a synchronous dataflow. The hand-shake asynchronous communication between two local components is realized by shared variables. Input and output actions are modeled as an interface automata or synchronous dataflow. After that, we present a mechanism to integrate the semantics of synchronous dataflow into the semantics of timed automata. Then, all the timed automata can be executed concurrently, and various properties can be simulated and verified with UPPAAL. Initial experiments results applied to a real system design encourage us. We are carrying out more experiments to check the scalability of our framework and implementing a tool to abstract timed automata network from cooperated VHDL modules and C functions.

6. ACKNOWLEDGEMENT

This research is sponsored in part by NSFC Program (No. 61202010, 91218302), National Key Technologies R&D Program (No.SQ2012BAJY4052) and 973 Program (No.2010CB 328003) of China.

7. REFERENCES

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[2] P. Amagbégnon, L. Besnard, and P. Le Guernic. Implementation of the data-flow synchronous language signal. In *Proceeding of the ACM SIGPLAN PLDI*, volume 30, pages 163–173. ACM, 1995.

[3] G. Berry, S. Ramesh, R. Shyamasundar, et al. Communicating reactive processes. In *Proceedings of the ACM SIGPLAN-SIGACT symposium on POPL*, volume 20, pages 85–98. ACM, 1993.

[4] G. Berry and E. Sentovich. Multiclock esterel. *Proceedings of the Correct Hardware Design and Verification Methods*, pages 110–125, 2001.

[5] F. Boussinot and R. De Simone. The esterel language. *Proceedings of the IEEE*, 79(9):1293–1304, 1991.

[6] S. Edwards and O. Tardieu. Shim: A deterministic model for heterogeneous embedded systems. *IEEE Transactions on Very Large Scale Integration Systems*, 14(8):854–867, 2006.

[7] N. Halbwachs, F. Lagnier, and C. Ratel. Programming and verifying real-time systems by means of the synchronous data-flow language lustre. *IEEE Transactions on Software Engineering*, 18(9):785–793, 1992.

[8] D. Harel. Statecharts: A visual formalism for complex systems. *IEEE Transactions on Software Engineering*, 8(3):231–274, 1987.

[9] C. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[10] L. Ju, B. Huynh, S. Chakraborty, and A. Roychoudhury. Context-sensitive timing analysis of esterel programs. In *Proceeding of the 46th ACM/IEEE Design Automation Conference, 2009.*, pages 870–873, 2009.

[11] E. A. Lee and D. G. Messerschmitt. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Transactions on Computers*, 100(1):24–35, 1987.

[12] E. A. Lee and D. G. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

[13] S. Ramesh, S. Sonalkar, V. Dąrsilva, N. Chandra R, and B. Vijayalakshmi. A toolset for modelling and verification of gals systems. In *Proceedings of the International Conference on Computer Aided Verification*, pages 385–387. Springer, 2004.