Design and Optimization of Multiclocked Embedded Systems Using Formal Techniques

Yu Jiang, Hehua Zhang, Zonghui Li, Yangdong Deng, Xiaoyu Song, Ming Gu, and Jiaguang Sun

***Abstract*—Today's system-on-chip and distributed systems are commonly equipped with multiple clocks. The key challenge in designing such systems is that two situations have to be captured and evaluated in a single framework. The first is the heterogeneous control-oriented and data-oriented behaviors within one clock domain, and the second is the asynchronous communications between two clock domains. In this paper, we propose to use timed automata and synchronous dataflow to model the dynamic behaviors of the multiclock train-control system, and a multiprocessor architecture for the implementation from our model to the real system. Data-oriented behaviors are captured by synchronous dataflow, control-oriented behaviors are captured by timed automata, and asynchronous communications of the interclock domain can be modeled as an interface timed automaton or a synchronous dataflow module. 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 that are important to guarantee the correctness of the system can be simulated and verified within the framework. We apply the framework to the design of a control system described in the standard IEC 61 375 and several bugs are detected. The bugs in the standard have been fixed, and the new version has been implemented and used in the real-world subway communication control system.**

***Index Terms*—Control-oriented behavior, data-oriented behavior, multiclock, synchronous dataflow, timed automata, train-control embedded system.**

Manuscript received July 19, 2013; revised November 17, 2013 and January 19, 2014; accepted March 8, 2014. Date of publication April 9, 2014; date of current version January 7, 2015. This work was supported in part by National Natural Science Foundation of China Programs (61202010, 91218302), in part by the National Key Technologies R&D Program (SQ2012BAJY4052) and 973 Program (2010CB328003) of China, and in part by the Tsinghua University Initiative Scientific Research Program (20131089331).

Y. Jiang is with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China; and also with the Tsinghua National Laboratory for Information Science and Technology, Key Laboratory for Information System Security, Ministry of Education, School of Software, Tsinghua University, Beijing 100084, China (e-mail: jiangyu198964@gmail.com).

H. Zhang, Y. Deng, M. Gu, and J. Sun are with the Tsinghua National Laboratory for Information Science and Technology, Key Laboratory for Information System Security, Ministry of Education, School of Software, Tsinghua University, Beijing 100081, China (e-mail: zhanghehua@gmail.com).

Z. Li is with Institute of Microelectronics, Tsinghua University, Beijing 100081, China.

X. Song is with the Department of Electrical and Computer Engineering, Portland State University, Portland, OR 97207-0751 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2014.2316234

I. Introduction

EMBEDDED systems are being widely used in all kinds of applications and are traditionally designed and optimized using a synchronous language with a single clock. Such an assumption of global synchronization greatly helps reduce the complexity of the design. The class of synchronous languages contains mainly Esterel [1], Lustre [2], Signal [3], and Statecharts [4]. Those languages and the corresponding tools such as SCADE [5] are good for compact single-clock 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 output in response to input events, while in 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 contains not only the control-oriented network communication protocols running on the processor but also the data-dominated algorithms for dealing with the voice signal. Furthermore, embedded systems are increasingly adopting multiclock solutions due to the low-power requirement and the pervasive usage of IPs from different vendors. This is particularly true for the train-control system described in the standard international electrotechnical commission (IEC) 61 375. Hence, there has been a recent surge for methods to guarantee the functional and sequential correctness when designing multiclock train-control systems.

In this paper, we present a timed automata [6] and synchronous dataflow [7] based framework to address the problem in modeling and validating the heterogeneous behaviors of the multiclock embedded system. In our framework, the system is modeled as a network of timed automata and synchronous dataflow, which is a collection of local synchronous domains and asynchronous communications. The main novelties of the framework are as follows.

1) With the guard defined on different clock remapping mechanisms of each automata transition, the local system component can be modeled as a purely synchronous node.
2) With the shared variables and special synchronize input/output actions, the asynchronous communication through the handshake protocol of the interclock domain can be modeled in an interface timed automaton without a clock or a synchronous dataflow module.
3) The control-oriented components are modeled by timed automata, the data-oriented components are modeled by

synchronous dataflow, and the behaviors of each synchronous dataflow module are interpreted as equivalent timed automata to maintain the semantic consistency for simulation and verification.

4) A platform for the implementation from our model to the real system is proposed, including two kinds of processing units: the Advanced RISC Machines (ARM) processor for control-oriented parts and field-programmable gate array (FPGA) processor for data-oriented parts. The communications are realized by connections between the pin of FPGA to the general-purpose input/output (GPIO) of ARM. The framework is applied to the design of the train system described in the standard IEC 61 375, and some safety-critical bugs in the standard are detected. The system implemented according to the fixed version is now used in real life.

II. Related Work

Some engineers propose computer-aided simulation methods to validate the embedded system [8]. Simulation methods give accurate results when system failures occur. However, simulations are inefficient when applications are complex and the number of vehicles is large. Another drawback is that simulations are based on simulation patterns. The effectiveness of simulation depends on the number and quality of patterns. Hence, the exhaustiveness cannot be guaranteed. In order to overcome the limitation, formal verification methods for safety-critical systems [9] have been recognized. Formal methods support the automatic verification of the control system to a large extent [10]. Given the formal specification of the system and desired properties, model checkers such as SMV [11] can automatically decide whether the system satisfies those properties or not. Recent work [12] describes a method to verify the safety properties of the embedded system for railway signaling in Korea. However, they cannot deal with the inconsistency caused by multiclock-controlled behaviors.

A large body of work has been dedicated to the modeling and validation of multiclock systems. In the literature, the formal language-based approach (e.g., CRP [13] and MC-Esterel [14]) is appealing because it provides a unified basis for formal analysis to achieve the expected correctness. CRP combines the synchronous reactive model of Esterel [15] with the asynchronous coupling of CSP [16] to offer a mathematically elegant framework. Local synchronous Esterel modules communicate through rendezvous channels. The problem is that it is hard to support the data-driven operations and rendezvous protocol through asynchronous coordinators. Its variants such as CRSM and ECRSM [17] have similar properties and limitations. MC-Esterel was specifically developed for the design of multiclock digital systems. The designer is responsible for creating communication mechanisms among different clock domains. Every Esterel module needs an explicit clock, and the designer has to construct low-level synchronizers to guarantee the synchronization. While MC-Esterel provides a powerful mechanism for modeling asynchronous and multirate systems, the main problem is that the 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. Some translation-based frameworks are also proposed for the analysis of multiclock systems. For example, Ramesh *et al.* [17] take advantage of the asynchronous model checking, using a direct translation of CRP to Promela [19]. Doucet *et al.* [20] use a mixture of synchronous descriptions in Signal and asynchronous descriptions in Promela and provide a translation from Signal modules to Promela processes. Each clock domain is described by a Signal module, and communication between two clock domains is described by the Promela channel. These translation-based frameworks inherit the original limitation and cannot deal with complex data-oriented behaviors neither.

In general, compared to those methods that support multiclock domains, the key differences are as follows.

1) The proposed framework incorporates synchronous dataflow to model the complex data-oriented behavior while the Esterel-based frameworks such as CRP are just suitable for control-oriented behaviors.
2) The proposed framework interprets the mixed data-oriented model and control-oriented model with the labeled transition system to maintain the semantic consistency for both simulation and verification while Statechart-based frameworks such as Ptolemy and Simulink do not support formal verification.

III. Proposed Modeling and Validation Framework

To model the multiclock train-control system 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. The data-oriented, control-oriented, and multiclock domain compositions make the proposed model more close to the real implementation. In the design process, we can validate different design models derived from requirements with simulation and formal verification techniques, avoid potential errors that may lead to rework, and choose the best one. In the implementation process, we can also abstract the model from the implemented system and apply simulation and formal verification techniques to validate whether the system meets the requirements or not. The overall framework is depicted in Fig. 1.

It is clear in the proposed framework that all control-oriented parts are modeled as timed automata, which are connected in parallel. Each automaton is equipped with an extra synchronous clock to control local behaviors. When a clock-based guard becomes true, the automaton will take a transition. Although the increasing speeds of all clocks are synchronous, we provide mechanisms for different clock-based guards to support multiple clock domains. Each state in the timed automaton can be refined. This adds hierarchy and does not change the original semantic of timed automata. The hierarchy model can be translated into a flat timed automaton. The asynchronous communications between two timed automata are realized by rendezvous CSP. All data-oriented parts are modeled as synchronous dataflow. The actors represent the data-related computations, the firing rules specify the number of tokens

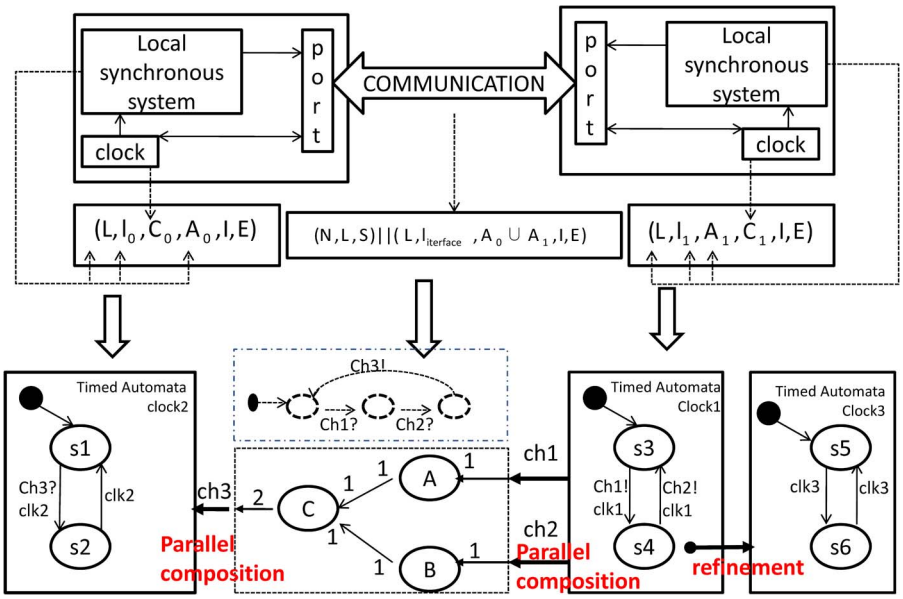


Fig. 1. Modeling framework for multiclock embedded system with heterogeneous behaviors. Each local synchronous component is modeled as a timed automaton with clock remapping and refinement of states. Each data-oriented component is modeled as a synchronous dataflow module. The asynchronous communication is modeled as a synchronous dataflow module or a timed automaton with input/output channels.

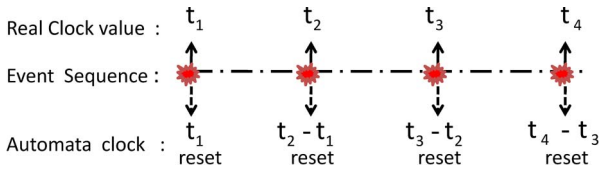


Fig. 2. The real clock value is mapped to the local clock in timed automata to ensure synchronous reaction behaviors. The real clock is redefined as some intervals. Those intervals are defined on the basic clock of timed automata.

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 timed automata to accomplish two functions: The first is reading tokens from the connected automata through the input channels to prepare for the firing of actors, and the second is sending tokens produced by the previous firing of actors to the connected automata through the output channels. The local synchronize modeling and communications between automata and synchronous dataflow are described hereinafter.

Local Synchronization Modeling: Each component of the local synchronization system has two kinds of reactions: the inner reaction within a component and the communication reaction across two components. Both are controlled by its clock. If the component is modeled as a timed automaton, the inner reaction is described as a regular transition edge with update action (A, E_{inner}), and the communication reaction is 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, 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 update action on each transition $(A, E_{inner} \bigcup E_{commu}, G)$. The guard is 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 Fig. 2. Because a transition can be triggered when the clock value satisfies this guard, the mechanism that each reaction will be triggered at the upper edge of the clock is captured. If the component is modeled as a synchronous dataflow, the inner reaction is described as regular firings of the actors with the consumed and produced tokens (N, L_{inner}). The communication reaction is described as a regular link (L_{commu}) with special synchronization actions on channel n. Because the model is executed in a periodic fashion, we add a period for iteration. This period will be interpreted by the clock variable of the equivalent timed automata.

Asynchronous Communication Modeling: The handshake 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 the control message and data through a bus to each other. 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, 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 attempts to write when M_3 is not ready, M_1 will be

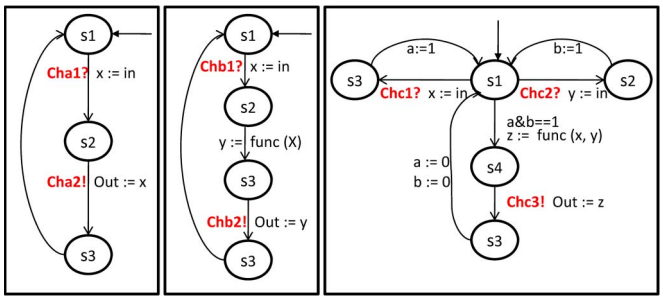


Fig. 3. Translation from synchronous dataflow to timed automata. The first is for the link between two actors and the link communicating with another module. The second is for actors which consume one token. The third is for actors that consume two tokens from two channels.

blocked. If M_3 attempts to read when M_1 is not ready, M_3 will be blocked. They can communicate with each other when both sides are in the corresponding states. If not, the communication is 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 that of the original modules. A set of timed automata $\mathrm{A_i} = \{L_i, l_i^0, A, C, E_i, I_i\}$ are composed into a network over a common set of clocks and actions with a parallel composition operator, and all automata are executed concurrently. Synchronous communication is realized by input and output actions, and asynchronous communication is realized by shared variables. To model the handshake synchronization, the action alphabets are defined on channel n. The input action "$n?$" represents receiving an event from channel n, while the output action "$n!$" stands for sending an event on channel n. The location vector for the automata network is defined as follows: $\bar{l} = (l_1, \cdots l_n)$, and $\bar{l}[l_i'/l_i]$ denotes the vector $\bar{l}$ with l_i being substituted with l_i'. The invariant function is defined on the composed location vectors as $I(\bar{l}) = \wedge_i I_i(l_i)$. Then, the semantic of the composed network is given in terms of the labeled transition system based on a pair $(\bar{l}, u)$, where $\bar{l}$ denotes a vector of current locations of the network and u is, as usual, a clock assignment recording the current values of clocks in the system. The transition rules are defined as follows:

1) $(\bar{l}, u) \rightarrow (\bar{l}, u + d)$ if $\forall d' \in [0, d], (u + d') \in I(\bar{l})$.
2) $(\bar{l}, u) \rightarrow (\bar{l}[l_i'/l_i], u')$ if $(l_i \overrightarrow{g, \tau, r} l_i'), u \in g, u' = [r \mapsto 0]u, u' \in I(\bar{l}[l_i'/l_i])$.
3) $(\bar{l}, u) \rightarrow (\bar{l}[l_i'/l_i, l_j'/l_j], u')$ if there exists $i \neq j$ such that $(l_i \overrightarrow{g, a?, r} l_i'), (l_j \overrightarrow{g, a!, r} l_j'), u \in g_i \wedge g_j, \quad u' = [r_i \cup r_j \mapsto 0]u, u' \in I(\bar{l}[l_i'/l_i], \bar{l}[l_i'/l_i])$.

The first is for the delay transition, which is similar to the case of the single timed automaton where the invariant of a location vector is the conjunction of the location invariants. The second is for the first kind of inner reaction. It defines the local actions where one of the processes makes a move, and the internal actions are denoted by the symbol τ. The third is for the second kind of interreaction. It defines synchronizing actions where two processes synchronize on a channel and take a transition simultaneously. Our model is defined on this semantic. The main challenge is how to demonstrate synchronous dataflow by timed automata.

In order to keep the semantic consistency of the mixed model, we translate the synchronous dataflow into equivalent

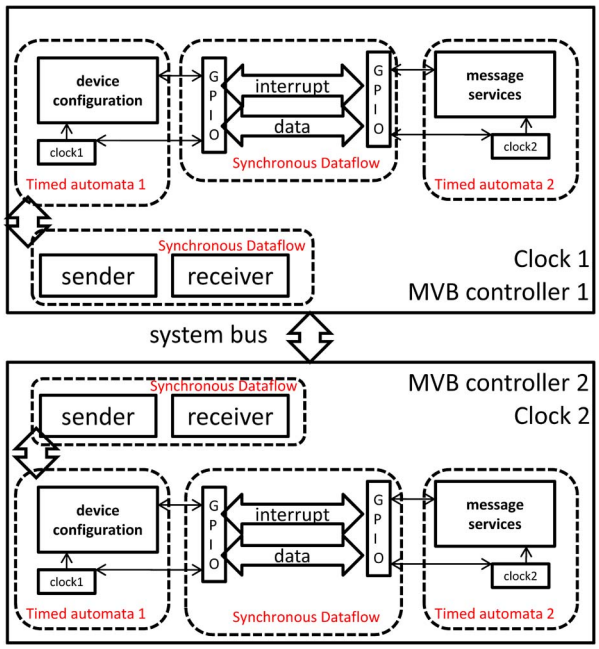


Fig. 4. Abstraction model of the function that initializes the controller with the device configure information, and the message transformation between two controllers with the sender and receiver.

timed automata. The translation procedure is modular. Each atomic element is represented by a timed automaton with channels reading 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 Fig. 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 tokens from the producer and pass them 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 actors that consume two tokens (x, y) from two channels (c_1, c_2) and produce a single token (z). We use two local variables (a, b) to denote the status of c_1 and c_2. When both of them have tokens, the transition will take place, and the output token 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 are 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 that of the original module.

IV. Real System Design

We conduct experiments on the design of the train communication control system described in the standard IEC 61 375 to present how our framework enables the design of the multiclock embedded system, including the system modeling, verification, and implementation.

A. Modeling of the System

The train-control system described in the standard IEC 61 375 is a safety-critical embedded system. The system consists

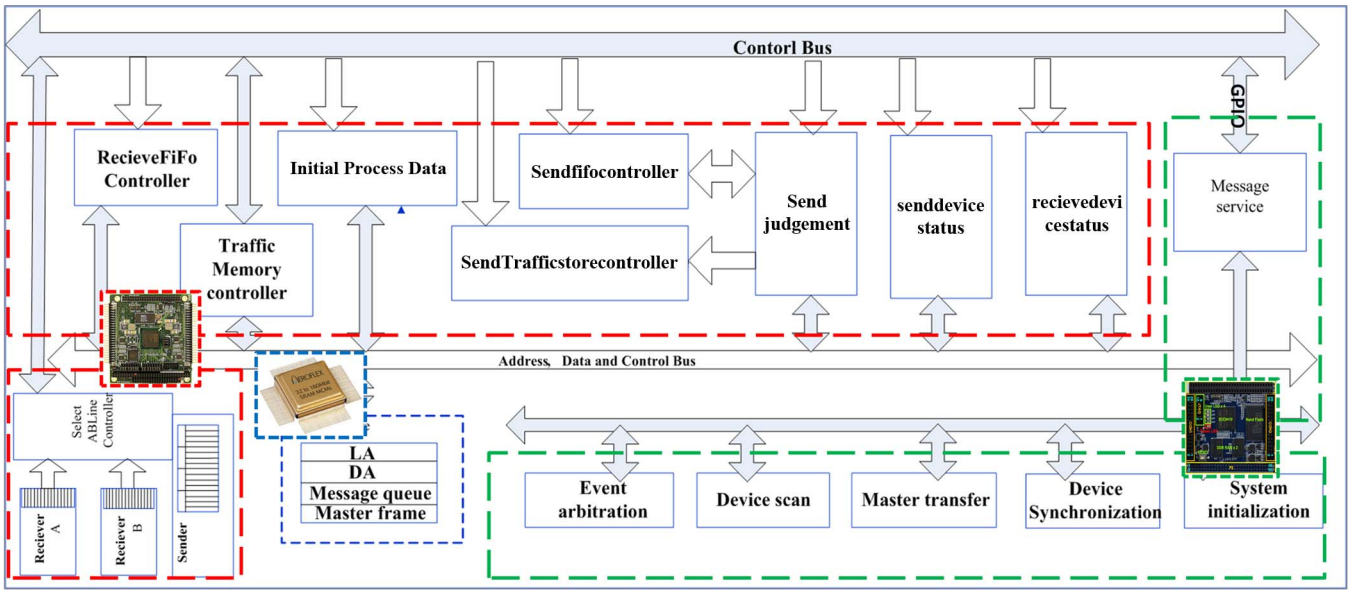


Fig. 5. Eighteen function modules of MVB controller. The modules contained in the red line are data-oriented behaviors modeled as timed automata. The modules contained in the green line are control-oriented behaviors modeled as dataflow which is embedded into ARM processor. The message service module realizes the real-time communication protocol verified earlier.

of two controllers: the multifunction vehicle bus (MVB) controller which interconnects devices within a vehicle and the wire-train-bus controller which interconnects the vehicles of a train. The MVB controller may contain multiple processors, which work under different clock frequencies. For example, data-oriented behaviors such as sampling may be implemented by an FPGA processor, and control-oriented behaviors such as communication rules may be implemented by an ARM processor. The FPGA may send interrupts to ARM through GPIO, and conflicts of multiple interrupts are scheduled by the embedded operating system eCos or embedded Linux [24] running on an ARM processor. This architecture is adopted by the most widely used MVB controller D113 [25] of Duagon company.

Communications between two different MVB controllers will be directed by different clocks. The communications provide two services. The first is the segmentation of the long message to fixed-size packets, and the second is the flow control and error recovery from end to end. The transfer of message shall be divided into three phases: 1) connection establishment; 2) acknowledged data transfer; and 3) disconnection. Before the model abstraction, some facts about the parameters in the standard need to be clarified as follows. The whole process of modeling strictly conforms to the finite-state machine and the transition rules presented in tables 32–35 of the standard 61 375. Following the rule, we construct the model of the communication system between two MVB controllers.

The top level model is presented in Fig. 4. The system bus is modeled as channels embedded into the sender and receiver modules. Each MVB controller is modeled as a node which can be partitioned into 18 modules furthermore as presented in Fig. 5. When embedding a controller into a vehicle, the system needs to initialize the device configuration information. As described in the standard, the data transfer is controlled by the GPIO of the main processor. It is supposed to send a pulse at the frequency of 10 MHz and then put 16-b data on the bus. The cooperated FPGA processor samples the data from the pin connected to the GPIO of the ARM processor and reads them into the static random access memory (SRAM). The asynchronous communication of GPIO is modeled as a synchronous dataflow. Due to the limited space, all these models can be found in [26].

B. Verification of the System

The pseudocode descriptions of the system in the standard have been abstracted as a network of timed automata and synchronous dataflow. The kernels of message services are the data acknowledgment and retransmission procedures. We abstract four properties from the two procedures. All abstracted properties are verified, and the bugs in the standard corresponding to the violated properties are further analyzed, depicted, and modified. The kernels of device configuration are that all device information can be read into SRAM.

First, let us see two properties about the data retransmission procedure between the automaton FSM_SENDER and FSM_RECEIVER. When FSM_SENDER receives an "rcv_NKi?" signal, it will decide whether the sequence i lies in the legal interval or not. The decision is implemented according to the pseudocode descriptions of table 33 in the standard as follows: $(expected < NK\_number \leq send\_not\_yet)$. The value of this decision expression is assigned to a variable NK. When FSM_RECEIVER sends a "send_NKi!" signal and switches to the SEND_NK state, FSM_SENDER should be able to deliver the retransmission request. The decision expression should be evaluated as true. The property is described as

$$A[\,](FSM\_RECEIVER.SEND\_NK \rightarrow (NK == true)).$$

The property is not satisfied when it is verified by Uppaal. A counterexample is detected. When FSM_SENDER sends data packets for the first time, it will send "DT0!-DT6!" signals.

If the "DT0!" signal is lost in FSM_CHANNEL, FSM_RECEIVER will reply with a "send_NK0!" signal asking for the retransmission of DT0. When "rcv_NK0?" is triggered, FSM_SENDER will evaluate the expression $(expected < NK\_number \leq send\_not\_yet)$, where the values of *NK_number*, *expected*, and *send_not_yet* are equal 0, 0, and 7, respectively. The value of the decision expression is evaluated to be false, and the system cannot deliver this legal case. We modify the pseudocode description in the standard, and the original decision expression presented earlier should be changed to $(\boldsymbol{expected \leq NK\_number \leq send\_not\_yet})$. When the guard is implemented according to the modified expression, the property verification is satisfied.

Another property is about rolling back the sending window of the automation FSM_SENDER in the retransmission procedure. When the modified guard $(expected \leq NK\_number \leq send\_not\_yet)$ is evaluated to be true, FSM_SENDER will roll back the sending window and retransmit the data packets numbered from the sequence number *i*. The *next_send* data packet of FSM_SENDER must be equal to the value of *NK_number*. The property is described as

$$A[\,]\,((NK == true) \rightarrow (NK\_number == next\_send))\,.$$

The property is not satisfied when it is verified by Uppaal. A counterexample is detected. When FSM_SENDER receives an "rcv_NK0?" signal, it is supposed to retransmit data packets numbered from the sequence number 0. However, it sends data packets numbered from the sequence number 7, which means that the value of *next_send* equals 7 while the value of *NK_number* equals 0. Hence, FSM_SENDER fails to accomplish the retransmission service. We look up the pseudocode description in the standard and find that the system rolls back the sending window with the following expressions: $\{expected := NK\_seq\_nr;\ send\_not\_yet := (expected + credit)\ \%\ 8;\}$. The system does not update the value of *next_send*. It keeps the value before receiving the rcv_NKi? signal. We modify the pseudocode description in the standard, and the original update expression presented earlier should be changed to $(\{\boldsymbol{expected := NK\_seq\_nr; send_{not\_yet} := (expected + credit)\ \%\ 8; next\_send := expected;}\})$. When the update function $\{rollback(i)\}$ in FSM_SENDER is implemented according to the modified expressions, the property is satisfied, and the system finishes the retransmission procedure successfully.

Corresponding to the retransmission procedure, the data acknowledgment procedure has two similar properties: 1) When FSM_RECEIVER sends a "send_AKi!" signal, FSM_SENDER should be able to deliver the data acknowledgment requirement, and that means that the guard $(expected \leq AK\_number \leq send\_not\_yet)$ should be evaluated to be true; the value of the expression is assigned to the variable AK; and 2) when the guard $(expected \leq AK\_number \leq send\_not\_yet)$ is evaluated to be true, FSM_SENDER should forward the sending window and send data packets numbered from the sequence AK_number. The *next_send* data packet of FSM_SENDER must be equal to the value of *AK_number*. The two properties described hereinafter pass the verification

$$A[\,]\,(FSM\_RECEIVER.SEND\_AK \rightarrow (AK == true))$$
$$A[\,]\,((AK == true) \rightarrow (AK\_number == next\_send))\,.$$

Then, let us see the property about the device configuration. According to static calculation, the system needs 58 pulses to store all device configuration data. Because the clock frequency of GPIO on the main processor is 10 MHz, the clock frequency of SRAM should be higher than 10 MHz to sample all pulses. According to the power constraint, lower frequency is better, and we set the guard on FSM_SRAM with 12 MHz. Then, formal verification is performed, and the property described hereinafter is violated. The prototype implemented by SystemC is simulated correctly, but the proposed model based on this clock frequency fails to finish the initialization procedure at times during the simulation in Uppaal

$$A[\,]\,(FSM\_SRAM.FINISHE \rightarrow (COUNTER == 58))\,.$$

We find a counterexample after 200 initialization times. Only 33 pulses are sampled. When the clock of FSM_SRAM starts later than the main processor for half of the period, it will lose some pulses. We increase the frequency of FSM_SRAM to sample more pulses. The property is satisfied when the frequency of FSM_SRAM is set as 24 MHz. This is a very interesting phenomenon that motivates our work. In the implementation section hereinafter, we will give more descriptions.

C. Implementation

The model does not need special architecture for implementation. The behavioral heterogeneity of the model needs to be mapped to a compatible architecture. Timed automata contain complex decision logic and minor computations. Synchronous dataflow is on the contrary. Control-oriented behaviors modeled as timed automata can be executed by some processing units for logic execution, and data-oriented behaviors modeled as synchronous dataflow can be executed by other processing units for data computation. Hence, the processor for timed automata does not need powerful computation ability while the processor for synchronous dataflow should have powerful computation ability and some features such as registers and DSP components for digital signal processing. Based on these rules, we propose a platform including two kinds of processing units: ARM processor and FPGA processor. The former is for control-oriented behaviors modeled as timed automata and implemented by C. The latter is for data-oriented behaviors modeled as synchronous dataflow and implemented by VHDL. Their communications are realized by interrupts through the direct connection between the pin of FPGA and the GPIO of ARM. The conflicts of multiple interrupts can be captured by the embedded operating system running on the ARM processor. We choose eCos because it is an open source and can be easily configured for our application. We can write some interrupt handle program with the ISR and DSR of eCos to schedule the communication. Other embedding operating systems such as VxWorks and embedded Linux and the corresponding scheduling mechanism [24] can also be used for implementation.

Then, the train-control system can be mapped onto this architecture. As described in Fig. 5, there are mainly 18 compound function modules cooperating to accomplish communication services. Some of them are modeled as synchronous dataflow, implemented by VHDL, and can be synthesized into FPGA

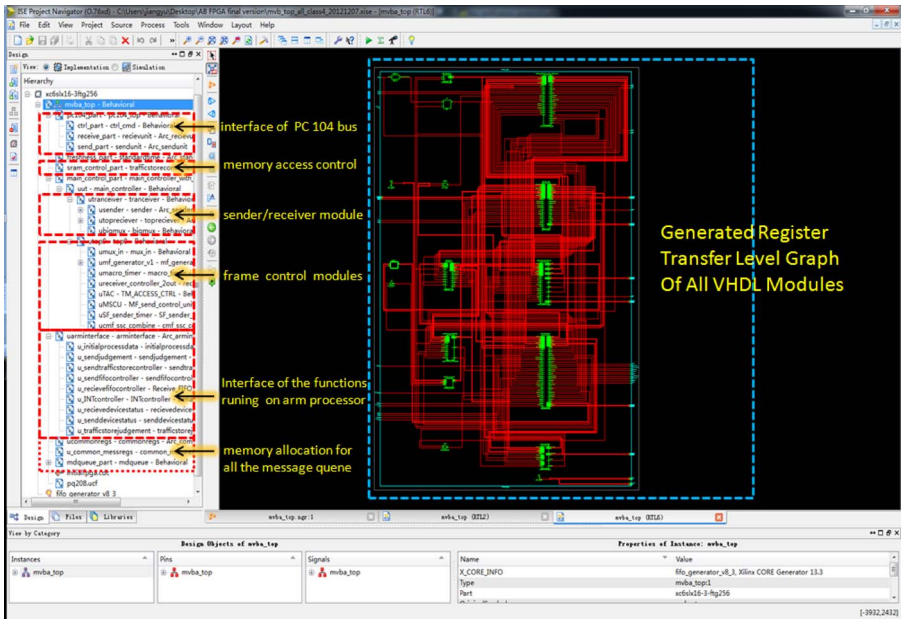


Fig. 6. All data-oriented functions are implemented by VHDL, including the interfaces that are used to communicate with the industrial computer and ARM processor. The memory access control corresponds to the traffic memory controller function in Fig. 5. The sender/receiver module corresponds to the receiver and sender functions in Fig. 5. The frame control modules correspond to all other functions contained in the red line of Fig. 5, such as send process data (sendfifocontroller) and send message data (sendtrafficstorecontroller). The memory allocation for all message queues operates on the SRAM contained in the blue line of Fig. 5. The interface of functions running on ARM processor sends interrupt triggers to the functions contained in the green line. All modules are synthesized in ISE development environment of Xilinx company. Then, the generated bit file can be loaded into the FPGA processor.

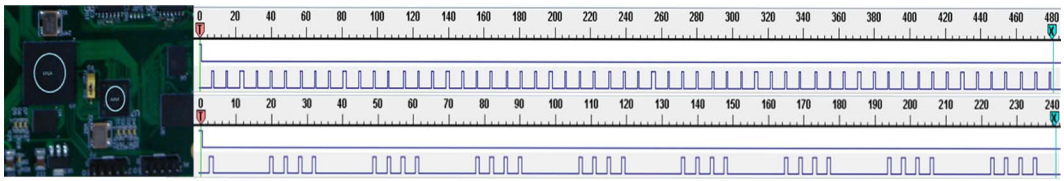


Fig. 7. Left side of the figure is our implemented experiment platform of the control system. The right side is the result run-time comparison for different designs, with 58 and 33 pulses sampled for 24 and 12 MHz, respectively.

for execution. Other modules are modeled as timed automata, implemented by C, and can be embedded into ARM for execution. The two processors and SRAM are composed as a network card as shown in the left side of Fig. 7. The two processors are controlled by two different clocks respectively and communicate with each other through the data bus in an asynchronous manner. The SRAM is controlled by the clock frequency of the FPGA processor. Our team implements the system according to the description of IEC 61 375 and the function partitioning of Fig. 5. Data-oriented behaviors such as sender and receiver functions are implemented by 11000 lines of VHDL codes. The framework is presented in Fig. 6. Moreover, the functions are presented in Fig. 5, and some interfaces for communications with the industrial computer and ARM processor are added. The interface parts receive data from the host industrial computer and decide when to send an interrupt trigger such as a master transfer to the ARM processor. Control-oriented behaviors such as the message service function are implemented with 65 000 lines of C codes. Due to the page limit, interested readers are referred to the standard IEC 61 375 for more details on the framework. For example, the 54 807 lines of C codes for the message service function are derived from the interface of functions and pseudocode descriptions in the standard. The detail functions and function call sequence descriptions for the transportation layer of the message service model verified earlier are located in pages 115–133, and the link layer is in pages 82–87. The event arbitration function is in page 555. The master transfer function is in page 263, etc.

Based on the implemented MVB controller, we simulate the verified properties presented in Section IV-B and set the running environment according to the counterexamples. The implemented system is embedded into the industrial computer to get some instructions from the keyboard, such as system initialization starts and communication starts. First, let us see the system initialization starts for a single controller. When the clock frequency of FPGA is set as 12 MHz, the waveforms sampled by ChipScope demonstrates that the FPGA pin only samples 33 pulses from 58 pulses of ARM GPIO because of the out of sync. This bug is not easy to trigger because the occurrence rate is low. According to the description in Section IV-B, we need to reduce the corresponding clock period, and the property is satisfied when the clock frequency of FPGA is set as 24 MHz. The waveforms sampled by ChipScope for the new design are also demonstrated, and the FPGA pin samples 58 pulses. The results are presented in Fig. 7.

Fig. 8. First card is for the model that is not verified. The second card is for the D113 system implemented by the Duagon company. The third card is for the model that is verified and fixed.

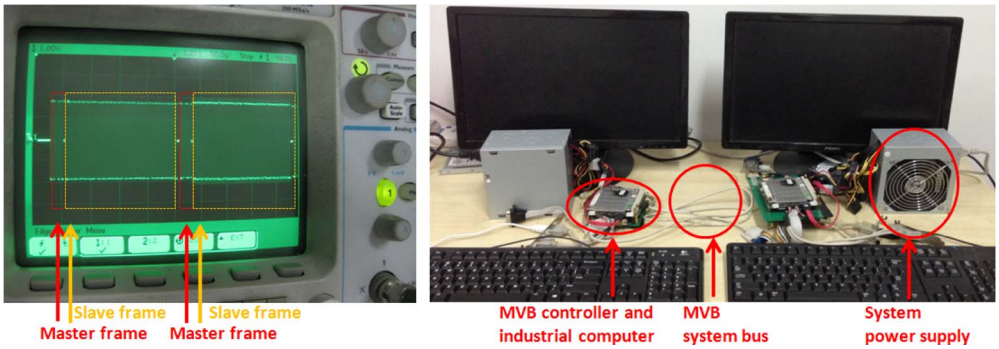


Fig. 9. Use the verified implemented system and the D113, and connect the two controllers with MVB bus. The system works well, and during each basic period, it can send seven master frames and receive slave frames correctly.

For message services between two MVB controllers, we test our implemented system with the world's most widely used MVB controller D113 from Duagon company. As shown in Fig. 8, the first platform is the MVB controller implemented based on the original standard IEC 61 375, the second platform is the MVB controller D113 of Duagon company whose codes are private, and the third platform is the MVB controller implemented based on the verified standard. The main difference between the first and the third platform is that the C codes of bold type as labeled in the property verification (see Section IV-B) are changed to the corrected ones. The difference is minor, but the effect is major.

We connect the first platform and the third platform to D113 with a vehicle bus in the manner of the right side of Fig. 9, respectively. When inserting some noise into the bus, the communication for the first platform crashes, but the communication for the third platform is correct, and the controller can retransmit the loss data caused by the frame loss on the vehicle bus. Furthermore, we use an oscilloscope to sample the data from the serial port that connected to the MVB bus. The sampled data are presented in the left side of Fig. 9. The master frame is sent correctly, and the system works well with a response slave frame. The implemented third platform is now used in a real-world subway, and the bugs found by our framework have been submitted to IEC. With the help of formal modeling, we make a contribution to avoid potential error of the train-control system and find the problem in the design stage.

V. Conclusion

In this paper, we have presented a timed automata and synchronous dataflow based design framework for modeling and validating the dynamic behaviors of the multiclock train-control system. Each local component of the system is modeled as a timed automaton with a local synchronous control clock or a synchronous dataflow module. The handshake asynchronous communication between two local components is realized by shared variables or CSP channels with input and output actions. After that, we present a mechanism to integrate the semantic of synchronous dataflow into the semantic of timed automata. Then, all timed automata can be executed concurrently, and various properties can be simulated and verified with Uppaal. A possible architecture consisting of two different behavior-oriented processors for the implementation of the train-control system from our model is proposed. Initial experiment results applied to the system design encourage us. Some safety-critical bugs in the original IEC standard 61 375 that cannot be detected by traditional techniques are detected within the proposed framework. The verified and implemented system has been used in real-world subway control in Beijing. We are carrying out more experiments to check the scalability of our framework and implementing a tool to generate VHDL and C codes from the model automatically.

References

[1] F. Boussinot and R. De Simone, "The Esterel language," *Proc. IEEE*, vol. 79, no. 9, pp. 1293–1304, Sep. 1991.
[2] N. Halbwachs, F. Lagnier, and C. Ratel, "Programming and verifying real-time systems by means of the synchronous data-flow language Lustre," *IEEE Trans. Softw. Eng.*, vol. 18, no. 9, pp. 785–793, Sep. 1992.
[3] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Trans. Comput.*, vol. C-36, no. 1, pp. 24–35, Jan. 1987.
[4] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Programm.*, vol. 8, no. 3, pp. 231–274, Jun. 1987.
[5] P. Caspi *et al.*, "From Simulink to SCADE/Lustre TTA: A layered approach for distributed embedded applications," *ACM Sigplan Notices*, vol. 38, no. 7, pp. 153–162, Jul. 2003.
[6] R. Alur and D. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, Apr. 1994.
[7] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. IEEE*, vol. 75, no. 9, pp. 1235–1245, Sep. 1987.

[8] K. Radecka and Z. Zilic, "Design verification by test vectors and arithmetic transform universal test set," *IEEE Trans. Comput.*, vol. 53, no. 5, pp. 628–640, May 2004.
[9] J. Bowen and V. Stavridou, "Safety-critical systems, formal methods and standards," *Softw. Eng. J.*, vol. 8, no. 4, pp. 189–209, Jul. 1993.
[10] A. Dahbura, K. Sabnani, and M. Uyar, "Formal methods for generating protocol conformance test sequences," *Proc. IEEE*, vol. 78, no. 8, pp. 1317–1326, Aug. 1990.
[11] A. Cimatti *et al.*, "NuSMV 2: An opensource tool for symbolic model checking," in *Computer Aided Verification*. Berlin, Germany: Springer-Verlag, 2002, pp. 359–364.
[12] J. Lee, J. Hwang, D. Shin, K. Lee, and S. Kim, "Development of verification and conformance testing tools for a railway signaling communication protocol," *Comput. Stand. Interfaces*, vol. 31, no. 2, pp. 362–371, Feb. 2009.
[13] G. Berry, S. Ramesh, and R. Shyamasundar, "Communicating reactive processes," in *Proc. ACM SIGPLAN-SIGACT Symp. POPL*, 1993, vol. 20, pp. 85–98.
[14] G. Berry and E. Sentovich, "Multiclock Esterel," in *Proc. Correct Hardware Des. Verification Methods*, 2001, pp. 110–125.
[15] L. Ju, B. Huynh, S. Chakraborty, and A. Roychoudhury, "Context-sensitive timing analysis of Esterel programs," in *Proc. 46th ACM/IEEE Des. Autom. Conf.*, 2009, pp. 870–873.
[16] C. Hoare, "Communicating sequential processes," *Commun. ACM*, vol. 21, no. 8, pp. 666–677, Aug. 1978.
[17] S. Ramesh, S. Sonalkar, V. Dsilva, R. Naveen Chandra, and B. Vijayalakshmi, "A toolset for modelling and verification of GALS systems," in *Proc. Intl. Conf. Comput. Aided Verification*, 2004, pp. 506–509.
[18] I. Viskic, L. Yu, and D. Gajski, "Design exploration and automatic generation of MPSoC platform TLMs from Kahn Process Network applications," *ACM Sigplan Notices*, vol. 45, no. 4, pp. 77–84, Apr. 2010.
[19] G. Holzmann, "The model checker spin," *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, pp. 279–295, Mar. 1997.
[20] F. Doucet, M. Menarini, I. H. Krüger, R. Gupta, and J.-P. Talpin, "A verification approach for GALS integration of synchronous components," *Theor. Comput. Sci.*, vol. 146, no. 2, pp. 105–131, Jan. 2006.
[21] G. Berry, "Circuit design and verification with Esterel v7 and Esterel Studio," in *Proc. IEEE Intl. HLVDT*, 2007, pp. 133–136.
[22] S. A. Edwards, CEC: The Colombia Esterel Compiler, 2003. [Online]. Available: http://www1.cs.columbia.edu/sedwards/cec
[23] J. C. Eidson, E. A. Lee, S. Matic, S. A. Seshia, and J. Zou, "Distributed real-time software for cyber-physical systems," *Proc. IEEE*, vol. 100, no. 1, pp. 45–59, Jan. 2012.
[24] R. B. Yehuda and Y. Wiseman., "The offline scheduler for embedded vehicular systems," *Int. J. Veh. Inf. Commun. Syst.*, vol. 3, no. 1, pp. 44–57, Jan. 2013.
[25] Duagon, Mvb Controller: D113. [Online]. Available: http://www.duagon.com/en/products/
[26] J. Yu, Detail Model for the Train Communication System, 2013. [Online]. Available: https://sites.google.com/site/jiangyu198964/home

Yu Jiang received the B.S. degree in software engineering from the Beijing University of Posts and Telecommunication, Beijing, China, in 2010. He is currently working toward the Ph.D. degree in computer science at Tsinghua University, Beijing.

His research interests include domain specific modeling, formal verification, and their applications in embedded systems.

Hehua Zhang received the B.S. and M.S. degrees in computer science from Jilin University, Changchun, China, in 2001 and 2004, respectively, and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2010.

She is currently a Lecturer with the School of Software, Tsinghua University. Her research interests include domain specific modeling, formal verification, and their applications in embedded systems.

Zonghui Li received the B.S. degree in computer science from Beijing Information Science and Technology University, Beijing, China, in 2010. He is currently working toward the M.S. degree in microelectronics at the Institute of Microelectronics, Tsinghua University, Beijing.

His research interests include high-performance graphics algorithms and embedded computing.

Yangdong Deng received the B.S. and M.S. degrees from the Electronics Department, Tsinghua University, Beijing, China, in 1998 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2006.

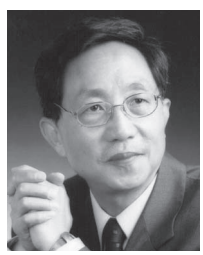
His research interests include parallel electronic-design-automation algorithms, electronic-system-level design, and parallel program optimization.

Xiaoyu Song received the Ph.D. degree in electronics from the University of Pisa, Pisa, Italy, in 1991.

In 1999, he joined the faculty at Portland State University, Portland, OR, USA, where he is currently a Professor in the Department of Electrical and Computer Engineering. His research interests include formal methods, design automation, embedded system design, and emerging technologies.

Ming Gu received the B.S. degree in computer science from the National University of Defense Technology, Changsha, China, in 1984, and the M.S. degree in computer science from the Chinese Academy of Science, Shengyang, China, in 1986.

Since 1993, she has been working as a Professor at Tsinghua University, Beijing, China. Her research interests include formal methods, middleware technology, and distributed applications.



Jiaguang Sun received the B.S. degree in automation science from Tsinghua University, Beijing, China, in 1970.

He is currently a Professor with Tsinghua University. He is currently the Director of the School of Information Science and Technology and the School of Software, Tsinghua University.