Metrol. Meas. Syst., Vol. XVII (2010), No. 3, pp. 461–480

PAN POLSKA AKADEMIA NAUK

METROLOGY AND MEASUREMENT SYSTEMS

Index 330930, ISSN 0860-8229

www.metrology.pg.gda.pl

M&MS

VERIFICATION OF FLEXRAY START-UP MECHANISM BY TIMED AUTOMATA

Jan Malinský, Jiří Novák

Czech Technical University, Faculty of Electrical Engineering, Technická 2, 166 27 Prague 6, Czech Republic (✉ malinsj@fel.cvut.cz, +420 22 435 2201, jnovak@fel.cvut.cz)

Abstract

This contribution deals with the modelling of a selected part of a new automotive communication standard called FlexRay. In particular, it focuses on the mechanism ensuring the start-up of a FlexRay network. The model has been created with the use of timed automata and verified. For this purpose the UPPAAL software tool has been used that allows the modelling of discrete event systems with the use of timed automata, and subsequently the verification of the model with the use of suitable queries compiled in the so called computation tree logic. This model can be used to look for incorrect settings of time parameters of communication nodes in the network that prevent network start-up and subsequently the start of the car. The existence of this model also opens the way for finding possible errors in the standard. On the basis of the model, the work gives a case study of the start-up mechanism behaviour verification in a FlexRay network consisting of three communication nodes.

Keywords: FlexRay, timed automata, modelling.

1. Introduction

Together with the current trend of development of electronic systems in cars, the need has arisen for a new communication standard. The necessity of a new standard has been mainly due to x-by-wire technology [1], which makes it possible to remove some of existing mechanical and hydraulic parts from cars and to replace them with intelligent and reliable electronic systems. This is mainly the case of the following technologies:

- Steer-by-wire provides electronic transfer of the turning angle of the steering wheel to the angle of the wheels.
- Brake-by-wire means electronic transfer of the position of the brake pedal to the operational intervention of the braking system.
- Drive-by-wire allows electronic transfer of the position of the accelerator pedal.

The x-by-wire systems for cars have been taken over from aircrafts (fly-by-wire systems), where they have been being reliably used for years with communication protocol TTP/C [2]. These systems lay the foundations for new intelligent cars where the control system is informed about the driver's intentions (to turn, to brake, to accelerate, *etc*.) electronically and together with other electronic systems and sensors it makes a decision on the safety of the driver's request. Subsequently, to enhance the safety of the passengers, these systems can even intervene into the control of the vehicle. For deeper studies of this new standard the reference sources [3] and [4], dealing with the description of the link and the physical layer of the FlexRay (FR) communication standard of the latest version 2.1, are recommended.

As all x-by-wire applications require hard real-time communication, FR is based on the TDMA (Time Division Multiple Access) medium access method, where dedicated time slots are used for communication. Such a system should provide a mechanism enabling the start of the network; here it is called start-up mechanism (SUM). Its correct and reliable functionality

Article history: received on Jan. 1, 2009; received in revised form on Jun. 29, 2010; accepted on Aug. 31, 2010; available online on Sept. 6, 2010; DOI: 10.2478/v10178-010-0039-z.

is crucial for the FR network exploitation. This article describes the model of the SUM of the FR network based on timed automata [7] and its use for SUM verification. For this purpose the UPPAAL software tool [5, 6] has been used that allows the modelling of discrete event systems with the use of timed automata and the subsequent verification of the model by means of suitable queries compiled in the CTL (Computation Tree Logic) [8].

Related work has been made in the past [9, 10], which deals with the modelling and verification of automotive communication standard CAN (Controller Area Network) by use of timed automata. However, this work does not cover the FR at all.

2. Brief introduction to timed automata and UPPAAL tool

A timed automaton (TA) is an extended version of finite state machine (FSM). The extension is done by the addition of a clock. As well as a FSM, a TA consists of locations and transitions between them. In the case of TA, transitions are expanded with conditions based on a clock. The clock is a variable assuming a non-negative real value. The value is incremented uniformly and spontaneously.

The UPPAAL is a software tool created in cooperation between Uppsala University, Sweden and Aalborg University, Denmark. It enables the modelling of a system as a network of timed automata working concurrently. For each individual component of timed automaton the below mentioned features can be set:

For transitions:

- Guard Condition (GC) defines a condition for transition between two locations. The transition can be executed only if the GC is fulfilled.
- Sync serves for synchronization between timed automata. One of them (automaton A) is considered to be in a location leading to another one by means of transition with sync parameter equal to name? (there is a question mark placed after the name). When the transition with sync = name! (there is an exclamation mark placed after the name) is executed in another concurrently running timed automaton (automaton B), the transition with sync = name? will be executed in automaton A too.
- Update allows initialization of variables including the clock. The updating is performed when transition is executed.

For locations:

- Invariant Condition (IC) defines a condition expressing an urgency of leaving the location. If the IC is fulfilled, the system can stay in or leave the location. The location has to be abandoned before the IC can be said to be unfulfilled. If the system is not able to do it because of GCs for all possible transitions and IC in appropriate location are unfulfilled at the same time, there is a deadlock in the system.

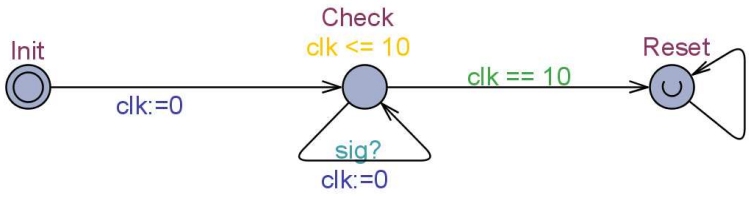

Fig. 1. Example of timed automaton – watchdog timer.

Let us consider the state of a system as a combination of locations in which the timed automata are situated, including a combination of variables. All possible states create the state space of the system. The system can be verified by queries compiled in CTL. UPPAAL gives the possibility to answer the queries by searching in the state space. Each query consists of

two parts. The first part is a path formula (A[], E<>, y→x, A<>, E[]) and the second is state formula *x*. Five kinds of queries are supported according to the kind of path formula:

Invariantly: A[] *x*

The query is satisfied when the state formula *x* is true in every possible state of system. This path formula is graphically depicted in Fig. 2.

Possibility: E<> *x*

The query is satisfied when there exists at least one state in the system where formula *x* is true. In other words there exists a possible path in the system how to get from initial state to the state where formula *x* is true. This path formula is graphically depicted in Fig. 3.

Leads to: *y*→ *x*

The query is satisfied when every path from a state where formula *y* is true leads to states where formula *x* is true. This path formula is graphically depicted in Fig. 4.

Inevitable: A<> *x*

The query is satisfied when every path from initial state leads to states where formula *x* is true. This path formula is graphically depicted in Fig. 5.

Potentially always: E[] *x*

The query is satisfied when there exists at least one path from initial state leading to *Sn* state in the system, where *Sn* is a state which is placed in the infinite or is circular (see Fig. 6). In addition, the formula *x* must be true in every gone-through state within the scope of the path including initial and *Sn* state. This path formula is graphically depicted in Fig. 6.

An easy example of a TA is shown in Fig. 1 depicting a model of a watchdog timer. The model consists of three locations where the first of them is init. The TA begins in this location and the transition to check is executed by sync parameter equal to start?. This action sets the variable clk (clock variable) to zero. In location check, the clk is automatically incremented with time. In this example the conditions for going to the reset are set as follows:

– The system is allowed to remain in check location only if value of clk is smaller than 10 time units or equal (IC in the location check). In other cases the check must be left.
– The transition from check to reset is allowed only if value of clk is equal to 10 time units (GC in the transition).

The transition to reset is always realized when the value of the clk reaches 10. There is not any another possibility in the system. However, if the request (sig?) is received and clk has not yet achieved 10, the clk is set to zero. In other words, if the difference of clk between two successive requests is always lower than 10, the system will never go to reset location.

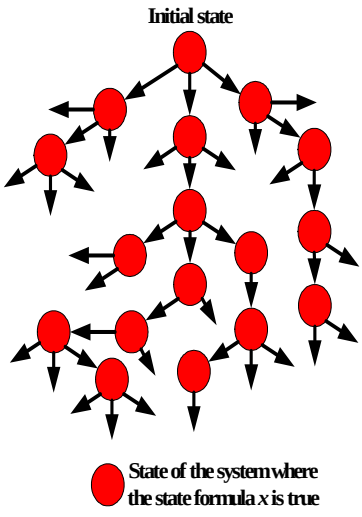

Fig. 2. Invariantly: A[] *x*.

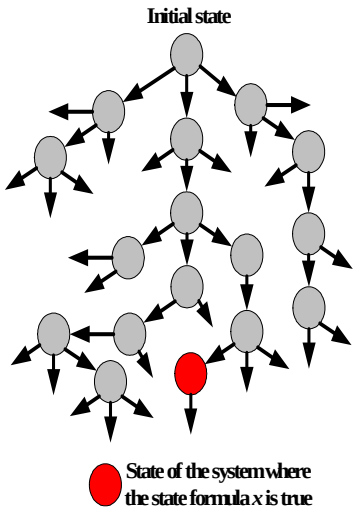

Fig. 3. Possibility: E<> *x*.

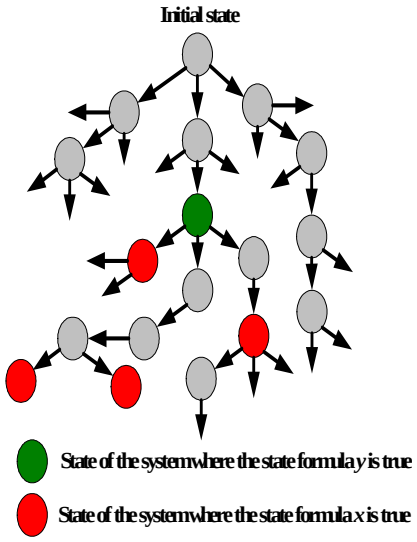

Fig. 4. Leads to: *y*→ *x*.

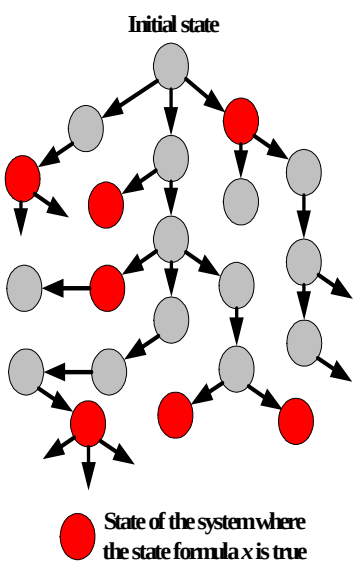

Fig. 5. Inevitable: A<> *x*.

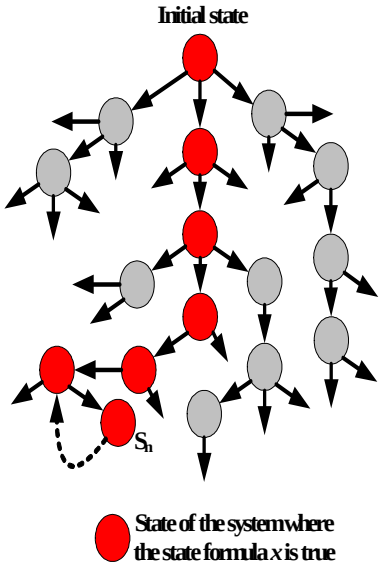

Fig. 6. Potentially always: E[] *x*.

3. Start-up mechanism of the FlexRay network

This chapter will describe the SUM (Start-up Mechanism) of the FR network from the moment of initiation of individual nodes, through their integration in the network, to the achievement of the normal active stage. The SUM means that the nodes must define a common start of the communication cycle (CC) and check the same pattern of the time schedule because of TDMA. A complete description can be found in [3].

Fig. 7 presents a simplified view of a FR network consisting of so-called cold-start nodes (CSNs) and non-cold-start nodes (NCSNs). In each FR network, there must be at least two CSNs. These two nodes (1. node and 2. node in Fig. 7) first select the CSN leader (CSNL) among them. This will be the node that first sends so-called CAS (Collision Avoidance Symbol). The other CSN automatically becomes so-called following CSN (CSNF). The CSNL will then initiate communication by sending start-up frames (SUFs) in each CC. By this way the beginning of CC is defined. The CSNF, on the basis of these SUFs, will also start sending its own SUFs to the corresponding time slot.

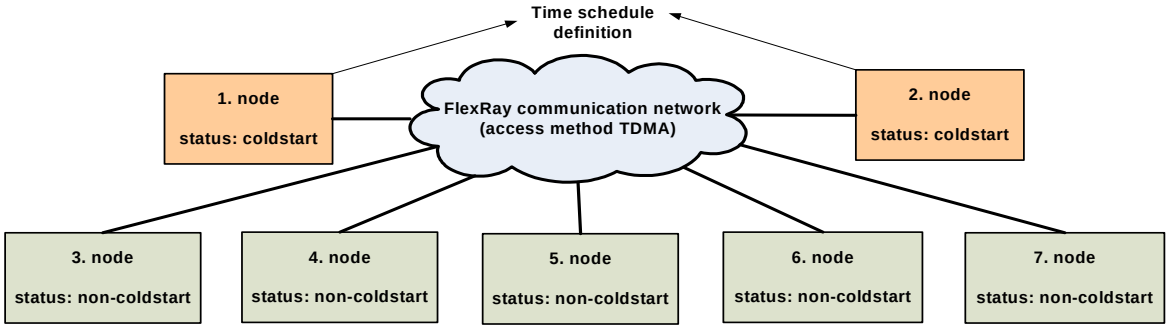

Fig. 7. Simplified view of the start-up mechanism.

This way both CSNs define the time schedule and the rest of the network, consisting of NCSNs, can then be integrated into this schedule. This integration will only be successful for a NCSN if it has the same time schedule as has been set by the CSNs.

If everything proceeds correctly, all the nodes in the network will pass into the normal active state when the network is started and ready for operation. However, if a NCSN is

configured for a different timing from the timing set by the CSNs, the NCSN will never be able to integrate to the network. If the time schedules of the CSNs do not correspond to each other, the network will not be started at all, which means that the car will not start either.

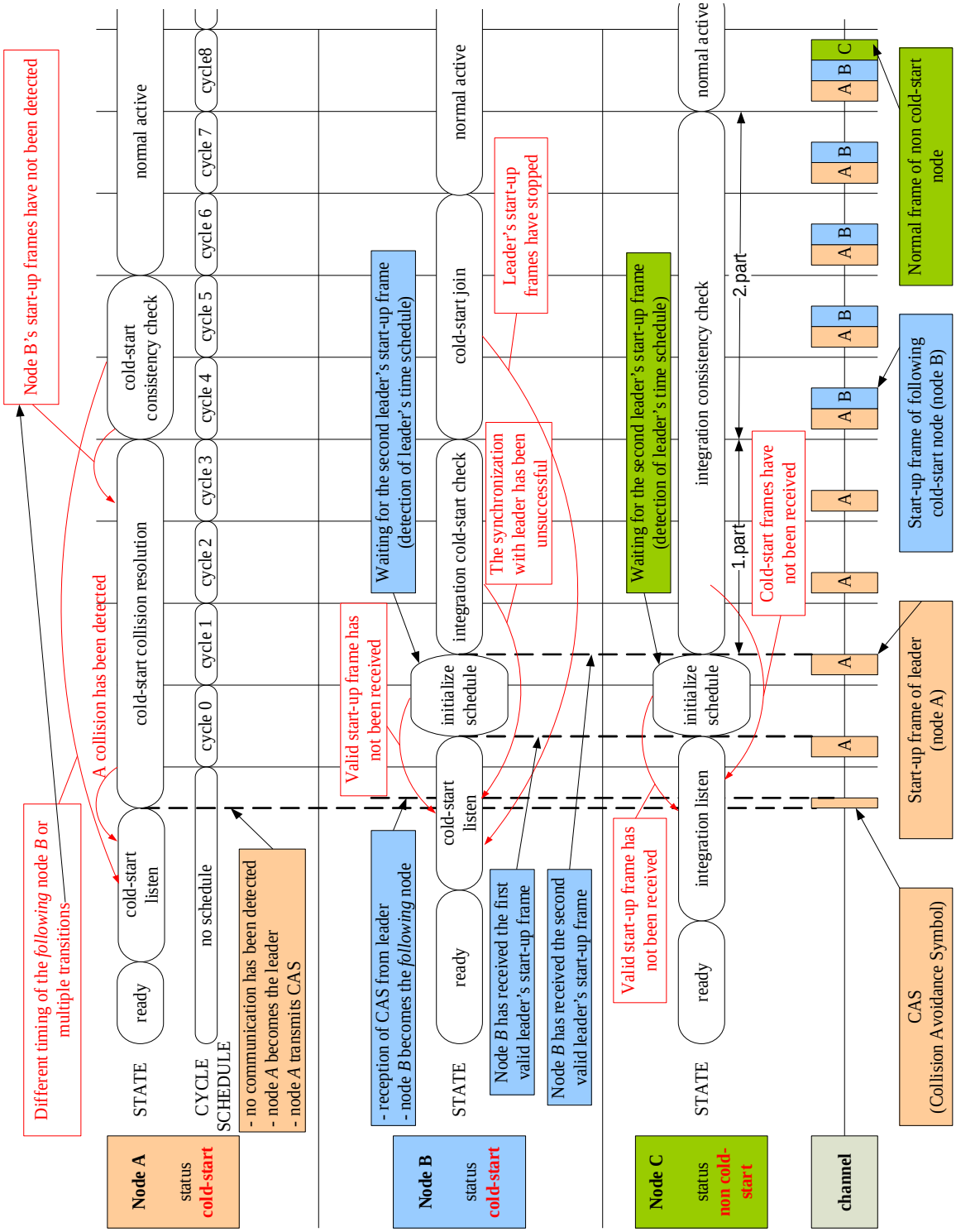

Fig. 8. Example of successful start-up procedure.

Fig. 8 shows an example of a successful SUM in a FR network consisting of three nodes A, B, C where both A and B are CSNs and the C is NCSN. The relevant start-up stages, which are gone through by nodes during the SU procedure, are depicted in sequence in Fig. 8. In this example all three nodes are considered to be successfully woken-up and located in the ready stage.

First of all, node A has gone to cold-start listen stage. In this stage, if the node does not catch any indication of communication coming from other nodes at least over one CC period, it sends the CAS. In this way the node A becomes a leader of the SU process and node B is requested to be the CSNF, which is subordinated to the leader. If node B had sent the CAS first, the situation would have been reversed. Leader election is unsuccessful when the CAS is transmitted by both CSNs simultaneously. To eliminate this unwanted case the leader goes to collision resolution stage immediately after the CAS transmission.

In the collision resolution stage, taking at least four CCs, the CSNL repeats sending the start-up frame (SUF) in the appropriate communication slot. If the CSNL catches another communication other than its own, it recognizes another CSN trying to become a leader as well. They both then return to the cold-start listen stage. To decrease the collision probability, the duration of the cold-start listen stage should be selected randomly. In the case of no collision, the CSNL successively sends at least four SUFs during the collision resolution stage in the appropriate time slot. These frames serve for initialization of the time schedule in the CSNF and NCSN. If the CSNF has been set incorrectly, it is not able to have the same view of time schedule as the CSNL has. In this wrong case, the network cannot be started at all. However, if the setting is correct, the CSNF goes into integration cold-start check stage where it performs synchronization of its own time base by means of the leader's start-up frames.

The leader then goes to the cold-start consistency check stage where the start-up frames from CSNF are expected in the appropriate slot. If the expectation is unsatisfied for the leader, the network cannot be started. However, if the CSNF has gone to cold-start join stage and started sending its own SUFs, there is a couple of SUFs (one originates from CSNL and second from CSNF) in each CC. The described process ensures that the time schedule for the rest of the NCSNs is established.

4. The model of the start-up mechanism

This section deals with the modelling of the SUM of a FR network. For this purpose the UPPAAL software tool has been used, which makes it possible to create a model with the use of timed automata and its subsequent verification. Verification queries for testing the correctness and robustness of the design of the modelled part of the standard have been compiled.

The SUM of each node is modelled by one timed automaton. Individual automata work in parallel within the system (FR network). Two types of a timed automaton have been created. The first type models a CSN, the other one a NCSN. Fig. 9 indicates interfaces of timed automaton modelling the CSN (orange colour) and NCSN nodes (grey colour) and their mutual interconnection. These TA are interconnected via three synchronization channels (SCH) start_frame_A, start_frame_B and CAS.

The behaviour of a SCH has been explained in chapter 2 (see the description of feature *sync* adjustable in transitions). In case of our model, SCHs are used for modelling of sending and receiving of start-up frames and CAS. In the CSN ***A***, "start_frame_A!" represents the transmitted SUFA and by means of "start_frame_B?" the SUFB is received from CSN **B**. Hence, if CSN **A** is set in a location leading to other by means of transition with sync parameter equal to "start_frame_B?", this transition will be executed when a transition with *sync* parameter equal to "start_frame_B!" is performed in CSN **B**. The similar reverse

situation takes place in the CSN **B**. The synchronization channel is also used for the modelling of transmitting and receiving the CAS. NCSNs receive all three SCHs by means of "start_frame_A!", "start_frame_B!", and "CAS!". However, they are not able to send anything to SCH.

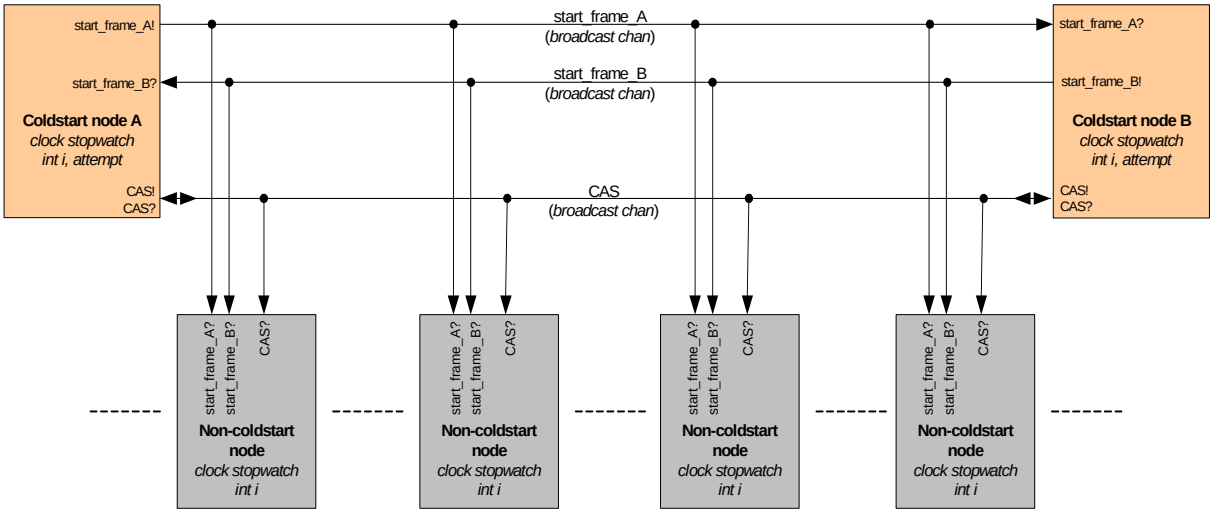

Fig. 9. Interfaces of timed automaton modelling the cold-start and non cold-start nodes.

Appropriate TA modelling the SUM is implemented inside each node depicted in Fig. 9. The model type used depends on whether the appropriate node is a CSN or NCSN. Fig. 10 presents the TA modelling SUM in a NCSN and fig.11 shows TA for SUM in a CSN node. Both of them were created using the UPPAAL tool in accordance with behavior of the SUM described in Section 3.

Models include guard conditions, sync parameters, update expression in transitions and invariant conditions in locations as well as location's names. All this is distinguished by means of colours:

- Location's name – red colour;
- Invariant condition – pink colour;
- Guard condition – green colour;
- Sync parameter – bright blue colour;
- Update expression – deep blue colour.

Before describing the TAs depicted in Fig. 10 and Fig. 11 in detail it is necessary to explain terms and variables used in these models.

μT. So-called "microtick" is defined within the FR standard and it is the smallest indivisible time unit used in standard. It is derived directly from the time base clock signal period. Typical values of one μT are 12.5; 25; 50; 100; 200 ns. Which of them is used depends on the clock signal frequency.

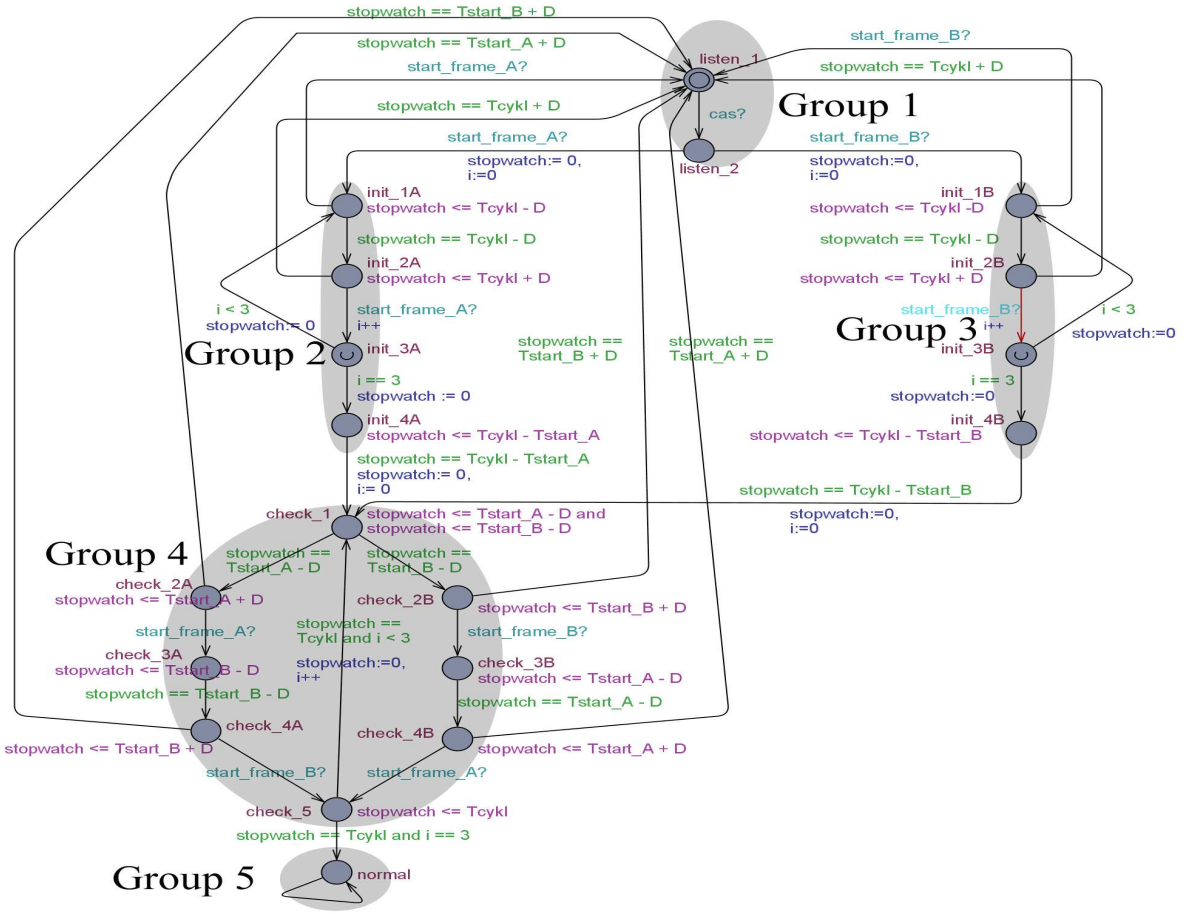

Fig. 10. Start-up mechanism in a non cold-start node modelled as a TA.

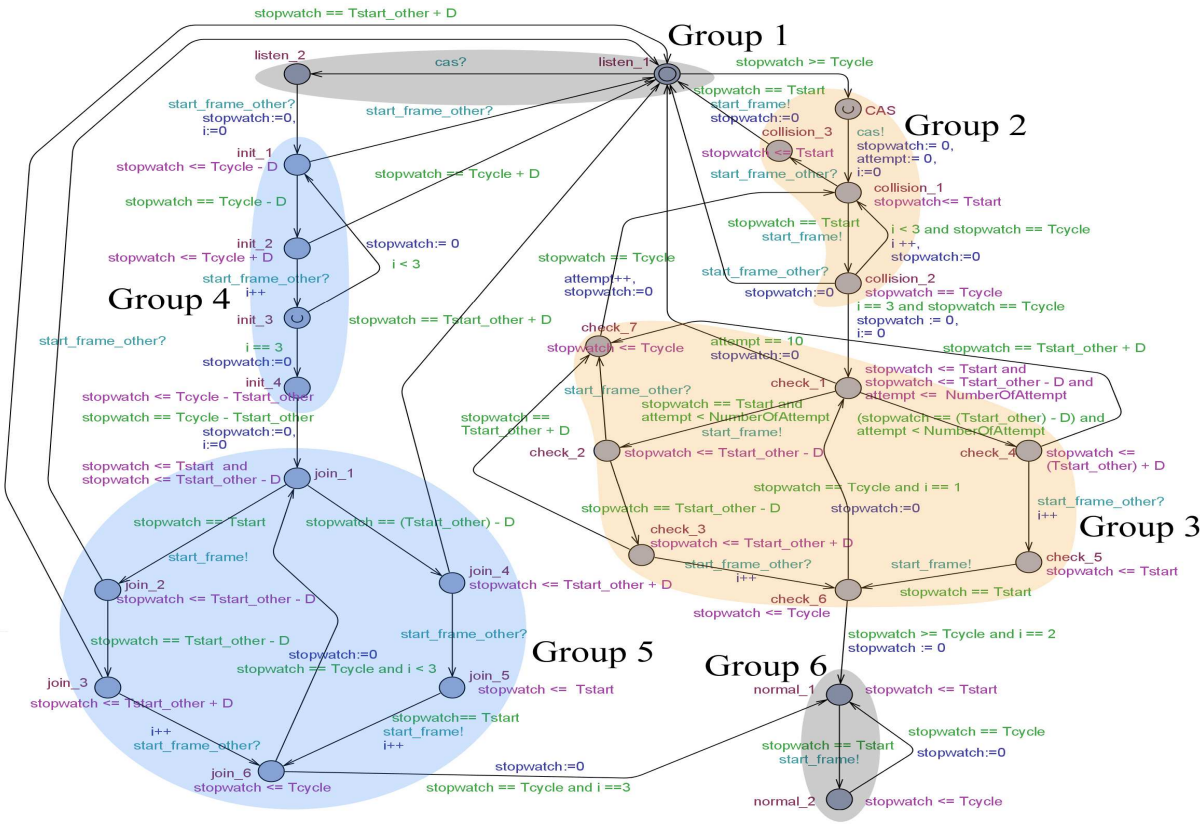

Fig. 11. Start-up mechanism in a cold-start node modelled as a TA.

Stopwatch [clock] – Freely running time base of a node. Each node has its own assigned time base in the form of the stopwatch variable of the clock type. In real applications due to some influences like a crystal ageing or temperature variation there appear discrepancies between time base frequencies. Unfortunately, the UPPAAL tool is not able to define more variables of clock type running with different speed. However, the frequency deviations between selected time bases can be modelled to a certain degree by time settings of the model. These deviations involve a different view of an accurate global time base. In other words, regarding the view of a global time base, there will be a difference between the departure time of SUF from a CSN and the arrival time of this SUF from the other node.

In fact, there is a so-called synchronization mechanism [3] in FR standard. This mechanism reduces the time differences. However, it is not possible to zero them totally. The model presented in this work does not consider the synchronization mechanism. Hence, this work covers the worst-case, where the synchronization mechanism does not work at all or works incorrectly.

i [int]. This variable stores the number of executed internal loops in the system.

Attempt [int]. Variable is used to count the number of unsuccessful SU attempts.

T_{start} [int]. Defines the number of μT from the start of the CC after which the CSN will send its SUF to the network.

T_{start_other} [int]. Defines the number of μT from the start of the CC after which the CSN will expect the arrival of the SUF from the other CSN.

T_{start_A} [int]. Defines the number of μT from the start of the CC after which a NCSN will expect the arrival of the SUF from the CSN A.

T_{start_B} [int]. Defines the number of μT from the start of the CC after which a NCSN will expect the arrival of the SUF from the CSN B.

T_{cycle} [int]. Defines the number of μT of the nominal length of the CC in the particular node.

D [int]. Permitted maximum deviation (in μT) between the expected and actual time of a SUF arrival. The CSN accepts the arrival of the SUF in the interval $<T_{start\_other} - D, T_{start\_other} + D>$. A NCSN accepts the arrival of start-up frames in the interval $<T_{start}\_A - D, T_{start}\_A + D>$ and $<T_{start}\_B - D, T_{start}\_B + D>$.

Description of the TA modelling start-up mechanism in a CSN. The TA modelling SUM in a CSN is shown in Fig. 11. The TA covers the model for the CSNL as well as the CSNF cases. In this subsection, the TA is explained in more detail. For easier comprehensibility, the diagram is divided into six groups of the TA's locations. In addition, the TA is separated into three main parts. The first of them contains groups highlighted by blue colour (group 4, group 5). They serve for modelling the SUM in the case of a CSNF. The second part marked in orange colour (group 2, group 3) is used for modelling of the SUM in a CSNL. The third part comprises groups highlighted by a grey colour on the background (group 1, group 6) and is used both for modelling of the SUM in a CSNF and in a CSNL. The following text describes each group in more detail.

Group 1. In fact, this group corresponds to the cold-start listen stage in a CSN according to Fig. 8. The group serves for listening to the communication bus during one CC. If the CAS from another CSN is received during this time, the TA will model a CSNF. In other cases, if no CAS is noticed, the TA can model a CSN trying to be leader.

Location listen_1. In listen_1 location, the TA waits until the duration of one CC passes. In other words, until the stopwatch variable reaches a value equal to T_{cycle}. After reaching it, the TA is allowed to go to group 2 where it will try to become CSNL by means of CAS sending. However, according to GC in transition listen_1 – CAS and missing IC in listen_1 location, the TA does not have to change the listen_1 location to CAS in group 2 when the time is equal to $T_{cycle.}$ In this way, the time of CAS sending can be chosen randomly. This is

necessary for reducing the number of situations when two CSNs want to be leader at the same time.

Location listen_2. If a CAS symbol, coming from another CSN, is received before the listen_1 has been left to the *CAS* location, the TA changes location to listen_2. In this case the node becomes CSNF. In the listen_2 location, the TA waits until SUFO from CSNL is received.

Summary of group 1. The group 1 fully corresponds to cold-start listen stage in a CSN (see Fig. 8). The main task of the group is the decision whether the node will become a CSNF or will get a chance to be CSNL.

Group 2. This group corresponds to the CS collision resolution stage in a CSNL (see Fig. 8 – node A). The main task is a collision recognition. A collision occurs when two CSNs want to become a leader at the same time. In other words, two CASs are sent by means of two CSNs simultaneously because they have reached group 2 concurrently. The collision recognition is performed on the basis of detection of a communication originated from other nodes. In the group 2, the CSNL sends its own SUF in appropriate time slots in every CC.

Location CAS. In fact, the CAS location itself has no actual function. The location serves only as a bridge between the listen_1 and collision_1 locations to achieve better lucidity. The location is always left immediately after entering and this takes no time (the location is urgent type). However, the transition between CAS location and collision_1 location is very important. By execution of this transition, the CAS is sent by means of sync channel with the parameter set as “cas!”. The CAS can be received in other CSNs as well as in other NCSNs. Strictly speaking, the CAS is received in every TA located in a location leading to another by means of a transition with the sync parameter set as “cas?”.

Location collision_1. According to ICs and GC in collision_1 location, the TA waits in this location until the stopwatch reaches T_{start} value. This is the time when the SUF must be sent. The sending is executed on the basis of the transition from the collision_1 to the collision_2 location. However, if a SUFO comes before the collision_1 has been left, the TA goes to the collision_3. In other words, there is another CSN in the network trying to be the leader as well. It means that the leader election is unsuccessful in this case.

Location collision_2. In the collision_2 location, the TA waits for the end of CC. In other words, until the stopwatch variable reaches a value equal to T_{cycle}. At this point, on the basis of IC and GC, the transition collision_2 to collision_1 is executed. The number of the transition repetitions is stored in the *i* variable. The transition execution causes the stopwatch variable to be set to zero and the variable *i* to be incremented. This way the transition from collision_1 to collision_2 location is executed four times. It means that the SUF is sent in time T_{start} in each of four successive CCs. If any SUFO does not arrive during these four CCs, the successful leader election is completed and the TA can continue to the group 3 when the end of the fourth CC is reached. However, if a SUFO comes from another CSN node while the TA is in the collision_2 location, the TA returns back to listen_1 location immediately and the leader election is not successful.

Location collision_3. The TA gets to the collision_3 location immediately after a SUFO is received during waiting in the collision_1 location. In addition, TA waits in the collision_3 until the stopwatch variable reaches a value equal to T_{start}. The SUFO comes before the sending of SUF. Hence, the TA sends its own SUF by transition to listen_1 location in the group 1. On the basis of this SUF, the other CSN, being situated in the collision_2 location, is forced to transit to the listen_1 location in group 1 and the leader election is unsuccessful.

Summary of group 2. The group 2 corresponds to the CS collision resolution stage (see Fig. 8 – node A). The main task is in recognising whether two CSNs have decided to be leader at the same time. If not, the leader election is successful and TA goes to the group 3. If

yes, the collision of two CSNs is recognized and the TA is forced to go back to the listen_1 in the group 1.

Group 3. This group corresponds to the CS consistency check stage in a CSNL (see Fig. 8 – node A). In this group, the TA continues with sending of SUFs, and checks whether SUFOs are received within an allowed range.

Location check_1. In the check_1 location, the decision whether the TA should continue to the check_2 or the check_4 is made on the basis of a SUF order given by time schedule. In addition, the number of executed transitions from the check_7 to the collision_1 location is checked to be 10. This transition corresponds with the reverse route from the CS consistency check stage to the CS collision resolution stage (see Fig. 8 – node A). The number of these transitions is stored in the variable attempt. When the variable attempt reaches 10, the TA goes from check_1 to the listen_1 location. This transition corresponds with the reverse route from the CS consistency check stage to the CS listen stage (see Fig. 8 – node A).

The transition leading from the check_1 to the check_2 location is executed when the variable stopwatch reaches value T_{start} earlier than T_{start_other} ($T_{start} < T_{start\_other}$). In other words, the time schedule is set in such a way that SUF is placed before SUFO in a CC. In addition, the variable attempt is smaller than 10. By executing this transition, the TA sends SUF at time T_{start}.

In the case that $T_{start} > T_{start\_other}$, and of course attempt < 10, the TA transits from the check_1 to the check_4 location after the stopwatch reaches value T_{start_other}, where SUFO is expected.

Location check_2. The TA is found in the check_2 location when the SUF has been transmitted first. Hence, the coming of SUFO is expected here. In fact, the location checks whether the arrival time of the SUFO is greater than ($T_{start\_other} - D$). If not, the SUFO comes too early and is unacceptable. In this case the TA comes to check_7 location. If the stopwatch variable reaches a value equal to ($T_{start\_other} - D$) without any SUFO coming, the TA continues to check_3 location.

Location check_3. In the check_3 location, the TA checks that the arrival time of a SUFO does not exceed the value ($T_{start\_other} + D$). In summary, the check_2 and check_3 locations supervise whether the SUFO comes within the range $<T_{start\_other} - D, T_{start\_other} + D>$. If so, the transition leading to check_6 location is executed. In other cases, the TA is forced to go to the check_7 location.

Location check_4. The TA gets to the check_4 location from the check_1 location when the time schedule is set in such a way that $T_{start\_other} < T_{start}$ (SUFO is placed before SUF in the CC), the stopwatch variable has reached a value equal to ($T_{start\_other} - D$) and the variable attempt has not yet reached 10. In this location, the TA waits until the SUFO comes. However, if the stopwatch variable reaches ($T_{start\_other} + D$) without any SUFO coming while being in the check_4, the TA comes to the check_7 location. In other words, the SUFO is checked whether its arrival time is in the range $<T_{start\_other} - D, T_{start\_other} + D>$. If so, the received SUFO is acceptable and the TA continues to check_5 location.

Location check_5. In the check_5 location, the TA waits until the stopwatch variable reaches a value equal to T_{start}. After reaching it, the TA sends SUF by means of transition to check_6 location.

Location check_6. In the check_6 location, the TA waits for the end of CC. In other words, until the stopwatch variable reaches a value equal to T_{cycle}. After reaching it, the TA either goes back to check_1 or to the normal_1 location situated in the group 6. Which transition is chosen depends on how many times the group 3 has been executed. This number of executions is stored in the variable *i*. For leaving the group 3 and entering to the group 6, it is necessary to execute the group 3 two times consecutively. It corresponds with the cold-start

consistency check stage (see Fig. 8 – node A), where two SUFs are sent and two SUFOs are received in two consecutive CCs.

Location check_7. The TA is found in the check_7 location when a SUFO has not come within the allowed range $<T_{start\_other} - D, T_{start\_other} + D>$ in the group 3. By means of this location the feedback to group 2 is realized. This transition corresponds with reverse route from the cold-start consistency check to cold-start collision resolution stage (see Fig. 8 – node A). The TA waits in the check_7 location for the end of CC. In other words, until the stopwatch variable reaches value T_{cycle}. When this transition to group 2 is executed, the number of unsuccessful attempts is incremented in the attempt variable.

Summary of group 3. The group 3 fully corresponds to the cold-start consistency check stage in a CSNL (see Fig. 8 – node A). SUFs are sent by the CSNL and SUFOs, coming from CSNF, are expected here. For this purpose the sequence of locations check_1 – check_2 – check_3 – check_6 is designated in the case that the SUF is situated in the CC before the SUFO. If the time schedule is set in such a way that the order of SUF and SUFO is reversed, the sequence check_1 – check_4 – check_5 – check_6 is considered.

The chosen sequence is performed two times. It covers two successive CCs (see Fig. 8 – node A – cold-start consistency check stage). If a SUFO comes out of allowed range $<T_{start\_other} - D, T_{start\_other} + D>$ during the group 3, the TA goes to check_7 location and subsequently, after reaching the end of CC, the transition to the group 2 is executed. This transition corresponds with the reverse route from cold-start consistency check to cold-start collision resolution stage (see Fig. 8 – node A).

Group 4. This group belongs to the area of TA destined for the modelling of a CSNF. In fact, the group corresponds to the initialize schedule and integration cold-start check stage (see Fig. 8 – node B). The main task is in checking whether the time duration between two successive SUFOs coming from CSNL is within the range $<T_{cycle} - D, T_{cycle} + D>$. If not, TA is forced to go back to group 1. Notice that in contrast to the group 3 in leader, the group 4 checks the duration between two SUFOs from CSNL. The group 3 checks the arrival time of SUFOs coming from CSNF. This difference comes from the fact that CSNL starts the time schedule generation and CSNF has to follow it.

Location init_1. In the init_1 location, the time between two successive SUFOs coming from CSNL is tested for the lower bounds of a range where the coming SUFOs are acceptable. The TA changes location from listen_2 to init_1 immediately after a SUFO is received. Here, the TA waits until the stopwatch variable reaches a value equal to $(T_{cycle} - D)$. After reaching it, the transition to init_2 location is executed. However, if a SUFO is received earlier than $(T_{cycle} - D)$ is reached, the SUFO is unacceptable and the TA goes back to listen_1 location in the group 1 and this is an unsuccessful attempt of SU for the CSNF.

Location init_2. The main task of init_2 location is to check whether the time between the receptions of two successive SUFOs does not exceed the higher limit of the range. If none of the SUFOs is received by the time the stopwatch variable reaches a value equal to $(T_{cycle} + D)$, the TA goes back to listen_1 location in the group 1 and this is an unsuccessful attempt of SU for the CSNF. In other cases the transition to init_3 is executed immediately after the SUFO is received.

Location init_3. In the init_3 location, a decision is made whether the sequence init_1 – init_2 will be repeated again or TA will transit to the group 5. The number of repetitions is stored in the *i* variable. According to GC and IC, the sequence init_1 – init_2 is executed three times. It corresponds with the checking of three periods between four successive SUFs (see Fig. 8 – node B –initialize schedule and integration CS check).

Location init_4. After the last SUF is successfully received in the sequence init_1 – init_2 – init_3, the stopwatch variable has a value equal to T_{start_other}. The only task for the init_4

location is to wait for the time remaining to the end of CC. In other words, the TA waits until the time ($T_{cycle} - T_{start\_other}$) elapses. After that, the TA goes to the group 5.

Summary of group 4. In fact, the group 4 corresponds to the init schedule and integration CS check stage (see Fig. 8 – node B). The main task is to check whether the time between two successive SUFOs coming from CSNL is within the range $<T_{cycle} - D, T_{cycle} + D>$. In other words, whether the view of CC duration in the CSNF is similar to that in the CSNL. If not, the TA goes back to the group 1, it corresponds with return to cold-start listen stage (see Fig. 8 – node B).

Group 5. This group belongs to the area of TA destined for the modelling of CSNF. This group is entered when the group 4 is left by means of init_4 location. The group fully corresponds to cold-start join stage (see. Fig. 8 – node B). This group is in charge of two main tasks. Firstly, checking whether the arrival time of SUFO received from the CSNL is within the acceptable range. Secondly, the TA (CSNF) sends its own SUFs in the appropriate time slot. The function of this group is similar to group 3 in leader's area.

Location join_1. From the join_1 location the TA can continue either to join_2 or to join_4 locations. Which of them is chosen depends on the time schedule. If the time schedule is set in such a way that $T_{start\_other} > T_{start,}$ in other words the CSNF sends the SUF earlier than SUFO from CSNL is received, the transition leading to join_2 location is chosen after the stopwatch variable reaches a value equal to T_{start}. By means of this transition a SUF from CSNF is sent.

However, if $T_{start\_other} < T_{start}$, the reception of a SUFO from the CSNL is expected before sending a SUF by means of CSNF. In this case the transition leading to join_4 is considered and is executed when the stopwatch variable reaches a value equal to lower bound ($T_{start\_other} - D$) of the range where arrival of the SUFO from CSNL is acceptable.

Location join_2. In the location join_2 the TA waits until the stopwatch variable reaches ($T_{start\_other} - D$). After reaching it the TA continues to join_3 location. However, if a SUF from leader is received before the lower bound is reached, it means that the SUFO has come too early. In this case, the TA moves back to listen_1 location. In other words, joining to the leader is unsuccessful because the time schedule between CSNL and CSNF is too different.

Location join_3. The location join_3 checks whether the arrival time of SUFO does not exceed ($T_{start\_other} + D$). If a SUFO is received before the stopwatch variable reaches ($T_{start\_other} + D$), the TA continues to join_6 location. However, if the value ($T_{start\_other} + D$) is reached without the SUFO reception, the SUFO is too late. In this case, the TA is forced to go back to listen_1 location in the group 1. It means that joining to the CSNL is unsuccessful because the time schedule between CSNL and CSNF is too different.

Location join_4. The TA location join_4 is entered from join_1 location when $T_{start\_other} < T_{start}$ and the stopwatch variable reaches ($T_{start\_other} - D$). The main task for join_4 is the same as for join_3. It is the checking whether the arrival time of the leader's SUF does not exceed ($T_{start\_other} + D$).

Location join_5. The TA waits in the location join_5 until the stopwatch variable reaches T_{start}. After reaching it the SUF is sent by execution of the transition from join_5 to join_6 location.

Location join_6. The TA waits for the end of CC in the join_6 location. In other words, it waits until the stopwatch variable reaches T_{cycle}. After reaching it the TA can go either to join_1 or to normal_1 location. Which of them is chosen depends on how many times the group 5 has been repeated. The number of these repetitions is stored in the variable *i*. This variable is incremented by executing the transition either from join_3 to join_6 or from join_5 to join_6. According to GC, the group 3 is executed three times. It covers the sending of three SUFs by CSNF and checking whether the received SUFOs come in the acceptable time range. It corresponds with cold-start join stage (see Fig. 8 – node B).

Summary of group 5. The group 5 fully corresponds with cold-start stage (see Fig. 8 – node B). The group is called to do two main tasks. Firstly, transmitting of SUFs by the CSNF is performed in the appropriate communication time slot. Secondly, the TA checks whether the received SUFOs are in range $<T_{start\_other} - D, T_{start\_other} + D>$. Tasks take place either in first sequence of location join_1, join_2, join_3, join_6 or in second sequence join_1, join_4, join_5, join_6. Which one of them is chosen depends on the SUF order. If, according to the time schedule, the condition $T_{start\_other} > T_{start}$ is held true, the first sequence is used. In other cases, the second sequence is considered. The group is executed three times. The number of executions is the same as in cold-start join stage (see Fig. 8 – node B).

Group 6. This group is the same for both the CSNL and CSNF. Entering this group means that SU procedure has been successful for the node. The group corresponds to normal active stage (see Fig. 8 – node A, B).

In the normal_1 location, the TA waits for the time when the SUF should be transmitted. In other words, it waits until the stopwatch variable reaches a value equal to T_{start}. After reaching it, the TA goes to normal_2 location. By this transition, the SUF is sent.

In the normal_2 location, the TA waits until the end of CC is reached. It means until the stopwatch variable reaches value T_{cycle}. After reaching it, the TA goes to normal_1 location. By this transition, the stopwatch variable is set to zero.

Summary of group 6. After entering the group 6, the modelled CSNL or CSNF has successfully finished its own SU procedure. The node continues sending SUFs in the appropriate time slots. These frames are necessary for other nodes where successful performing of SU is still in progress. In the group 6, the node no longer checks SUF from other nodes.

Description of the TA modelling start-up mechanism in a NCSN. The TA, modelling SUM in a NCSN is shown in Fig. 10. In this subsection the TA is explained in detail. For better comprehensibility the diagram is divided into 5 groups of TA's locations.

Group 1. In fact, this group corresponds to integration listen stage in a NCSN according to Fig. 8. The TA waits in this group until a CSN sends CAS and first SUF.

Location listen_1. The listen_1 is the initial location where TA waits for the arrival of a CAS sent by a CSNL. After the CAS is received, the TA changes location from listen_1 to listen_2 location. This change is provided by the transition between them with the sync parameter "cas?".

Location listen_2. In the listen_2 location, the TA waits for SUF sent by the CSNL. From listen_2 location, the TA can continue either to group 2 or group 3. The chosen direction depends on which one of two CSNs (A, B) has become leader. If the CSN A is the leader, the TA goes to group 2 on the basis of SUFs sent by CSN A. This transition is executed by sync parameter "start_frame_A?". In the case that the CSN B is the leader, the sync parameter "start_frame_B?" is considered and TA transfers to group 3.

Summary of group 1. The group 1 completely corresponds to the integration listen stage in a NCSN (see Fig. 8 – NODE C). The CAS and first subsequent SUF from CSNL are expected here. After receiving it, the TA can go either to group 2 (CSN A is expected to be leader) or group 3 (CSN B is expected to be leader).

Group 2. The group models the initialize schedule stage in NCSN and the first part of integration consistency check stage (see Fig. 8 – NODE C). The main task is checking the duration between two received SUFAs.

Location init_1A. According to IC and GC, the TA waits in init_1A until the clock variable stopwatch reaches a value equal to $(T_{cycle} - D)$. After reaching it, the TA is forced to change the location to init_2A. However, if a SUFA is received before init_1A is left, the SUFA is unacceptable. It means that this location checks whether SUFA has been received too early (under the lower limit of acceptance). If so, the TA goes back to the listen_1

location in the group 1. This corresponds to the reverse route from initialize schedule to integration listen stage (see Fig. 8 – NODE C).

Location init_2A. In init_2A location the TA waits for the arrival of SUFA. On the basis of this frame, The TA goes to init_3A and the variable i is incremented, where the variable i is a counter of the number of received SUFAs. However, if a SUFA is not received until the stopwatch reaches a value equal to (T_{cycle} + D), the SUFA is unacceptable. In summary, the init_2A location checks whether the SUFA has been received too late (over the higher limit of acceptance). If so, the TA goes back to the listen_1 location in the group 1. This corresponds to the reverse route from initialize schedule to integration listen stage (see Fig. 8 – NODE C).

Location init_3A. One of two possible transitions from init_3A is to the init_1A location. On the basis of this transition, the sequence init_1A – init_2A – init_3A can be repeated more times. When this transition is executed, the variable stopwatch is set to zero. The number of executions of the sequence is stored in the variable *i*. According to transition GC, the sequence is executed three times. It covers the checking of three cycle durations between four SUFAs. The first checked cycle belongs to initialize schedule stage and others to the first part of integration consistency check stage in a NCSN (see Fig. 8 – NODE C).

Location init_4A. Immediately before leaving the init_3A, the TA receives the SUFA and the stopwatch had the value equal to T_{start_A}. By transition to init_4A, the stopwatch is set to zero. According to IC in init_4A and GC, the TA waits in init_4A location until the stopwatch reaches a value equal to ($T_{cycle} - T_{start\_A}$). In other words, the TA waits for the rest of the time remaining to the end of CC. In summary, the init_4A serves in waiting for the end of CC after the last SUFA is received in the first part of the integration consistency check stage in NCSN (see Fig. 8 – NODE C).

Summary of group 2. This group checks whether the time between two successive frames is within the range $<T_{cycle} - D, T_{cycle} + D>$. When four successive SUFAs are received and three time intervals between them have been verified to belong to the range $<T_{cycle} - D, T_{cycle} + D>$, the TA will continue to the group 4. However, if the time interval between any two SUFAs is out of the range, the TA will move back to listen_1 in the group 1. This activity accords with the initialization schedule stage in NCSN (see Fig. 8 – NODE C). It also covers the first part of the integration consistency check stage in NCSN, where the first two SUFAs are received (see Fig. 8 – NODE C) including waiting for the end of the last CC.

Group 3. In fact, the group 3 is basically the same as the group 2. The only difference is in the origin of SUF. In contrast to the group 2, group 3 serves when CSN B has become leader. Hence, instead of a SUFA, a SUFB is considered in the description of group 2.

Group 4. This group corresponds to the second part of integration consistency check stage where four pairs of SUFs (SUFA + SUFB) are expected (see Fig. 8 – NODE C). The main task is checking the time interval between SUFAs received from CSN A and SUFBs received from CSN B. In fact the basic idea of this group is very similar to group 2. In contrast to the group 2, the group 4 checks not only SUFs from CSNL but also from CSNF.

Location check_1. In the check_1 location the TA waits until the stopwatch variable reaches value ($T_{start\_A} - D$) or ($T_{start\_B} - D$). In other words, it waits until one of two lower bounds of ranges, where reception of SUFA or SUFB is acceptable, is reached. After reaching it, the TA goes to either check_2A location or check_2B location. Which of these two transitions is chosen depends on the order of received SUFA and SUFB. The order is defined by the time schedule. Now, let us consider the time schedule to be set in such a way that SUFA is sent first. In this case, the transition on the left side leading to check_2A location is chosen.

Location check_2A. In the check_2A location, the TA waits for SUFA reception. After the reception, the TA goes to check_3A location. However, when the variable stopwatch reaches value (T_{start_A} + D) without SUFA reception, the SUFA is unacceptable and the TA is

sent back to listen_1 in the group 1. In summary, the sequence of locations check_1 – check_2A checks whether a SUFA is received within the range $<T_{start\_A} - D, T_{start\_A} + D>$.

Location check_3A and check_4A. In the check_3A location, TA waits until the stopwatch variable reaches value ($T_{start\_B} - D$). After reaching it, with regard to IC and GC, the TA is forced to change location from check_3A to check_4A, where SUFB is expected. After receiving it, the transition to check_5 is executed. However, if the stopwatch variable reaches value equal to ($T_{start\_B} + D$) without SUFB reception, the SUFB is unacceptable and the TA goes back to the listen_1 location in the group 1. In summary, the sequence of locations check_3A - check_4A checks whether a SUFB is received within the range $<T_{start\_B} - D, T_{start\_B} + D>$.

Notice that the function of sequence of locations check_1 – check_2B – check_3B – check_4B is the same as check_1 – check_2A – check_3A – check_4A. Only the SUF order is reversed.

Location check_5. After successful reception of successive SUFA and SUFB, the TA is located in check_5 location. Here it is waiting for the end of the CC. In other words, the TA waits until the stopwatch reaches a value equal to T_{cycle}. After reaching it, the group 4 is repeated again. The number of repetitions is stored in the *i* variable. This variable is incremented when the transition from location check_5 to check_1 is executed. According to the GC in this transition, the group 4 is executed four times. It corresponds to the second part of integration consistency check stage in NCSN (see Fig. 8 NODE C). When group 4 is executed four times in sequence without coming back to group 1, the SU procedure is successful for the NCSN and the TA is forced to leave group 4 by going to normal location in the group 5.

Summary. The group 4 corresponds to the second part of integration consistency check stage in a NCSN (see Fig. 8 – node C). The purpose of the group is to capture a pair of SUFs (SUFA + SUFB) in four successive CC. All these frames are checked whether their arrival time suits the range $<T_{start\_A} - D, T_{start\_A} + D>$ for SUFA and the range $<T_{start\_B} - D, T_{start\_B} + D>$ for SUFB. If one of frames is out of range, the SU procedure is unsuccessful for the modelled NCSN and the TA moves back to listen_1 location in the group 1. In other cases the TA moves to normal location situated in the group 5.

Group 5. This group contains only one location called normal. Entering this location means a successful SU procedure for the modelled NCSN. According to normal active stage in the NCSN (see Fig. 8 – NODE C), the node sends its own data frame C in an appropriate time slot. However, the TA does not include it because the sending of data frame from NCSN is insignificant for the SUM modelling.

5. A case study

A case study was made on a system (FR network) consisting of two CSNs A and B, and one NCSN C interconnected in accordance with Fig. 9. Nominal values of times parameters T_{start_A}, T_{start_B}, T_{cycle} in appropriate nodes CSN A, CSN B, NCSN C in the network have been set according to Table 1, Table 2 and Table 3.

Table 1. Nominal values of time parameter T_{start_A} in CSN A, CSN B and NCSN C.

NODE	**T_{start_A} [μT]**	
CSN A	10 000 μT	Number of μT from start of CC where CSN A sends SUFA
CSN B	10 000 μT	Number of μT from start of CC where CSN B expects arriving of SUFA. For CSN B the T_{start_A} is labelled as the T_{start_other}.
NCSN C	10 000 μT	Number of μT from start of CC where NCSN C expects arriving of SUFA

Table 2. Nominal values of time parameter Tstart_B in CSN A, CSN B and NCSN C.

NODE	**T_{start_B} [μT]**	
CSN A	20 000 μT	Number of μT from start of CC where CSN A expects arriving of SUFB. For CSN A the T_{start_B} is labelled as the T_{start_other}.
CSN B	20 000 μT	Number of μT from start of CC where CSN B sends SUFB
NCSN C	20 000 μT	Number of μT from start of CC where NCSN C expects arriving of SUFB

Table 3. Nominal values of time parameter Tcycle in CSN A, CSN B and NCSN C.

NODE	**T_{cycle} [μT]**	
CSN A	200 000 μT	Length of CC in CSN A
CSN B	200 000 μT	Length of CC in CSN B
NCSN C	200 000 μT	Length of CC in NCSN C

The set of above-mentioned parameters is an example of a typical time setting adopted from a real system.

5.1. Verification queries

For verification of the SUM in the network the following verification queries were applied.

Q1: **Is the system without a deadlock?**

Formula: A[] not deadlock

Fulfilled only if for all possible states in the system it is guaranteed that no deadlock status will occur.

Q2: Is it possible to start the FR nodes A and B?

Formula: E<> (cs_node_A.normal_2) and (cs_node_B.normal_2)

Fulfilled if there is a sequence in the system how to get from the initial state of the system to the state where both the processes modelling the CSNs cs_node_A and cs_node_B have achieved normal_2 location.

Q3: Is it possible to start the NCSN?

Formula: E<> (non_coldstart_node_C.normal)

Fulfilled if there is a sequence in the system how to get from the initial state of the system to the state where the process modelling the NCSN non_coldstart_node_A has achieved the *normal* location. The process strongly depends on the processes of the CSNs. If for some reason the CSNs are not started correctly, this query will never be fulfilled.

Q4: Is the network always started after the proper selection of one leader? In other words every path from the proper selection of one leader leads to the full start-up of the network.

Formula: (cs_node_A.listen_2 or cs_node_B.listen_2) --> (cs_node_B.normal_2 and cs_node_A.normal_2 and non_cs_node_C.normal)

Only fulfilled if after the successful selection of one leader (one node has become the CSNL and the other one the CSNF) all the other possible paths of the system lead to the state where both CSNs are in the normal_2 location and all the NCSNs in the normal location.

5.2. Evaluation of the results of verification queries

Each query Q1 to Q4 was executed several times, always with different time parameters in comparison with the nominal settings. In accordance with different time parameter settings the verification of SUM has different results. In this subsection, five results of the verification are introduced. For each case (1–5) the relevant time setting is mentioned. Notice that instead of parameters T_{start_A}, T_{start_B}, T_{cycle}, values ΔT_{start_A}, ΔT_{start_B}, ΔT_{cycle}, *i.e.* deviations from nominal values (Table 1, Table 2, Table 3), are given in cases 1 –5.

1. The network can be completely started without complications and the system will not get to the deadlock status. Slightly different time parameters in comparison with the nominal settings have not prevented the network to be fully started.

Example of time settings for this case:

	ΔT_{start_A} [μT]	ΔT_{start_B} [μT]	ΔT_{cycle} [μT]	D [μT]
CSN A	0	0	- 20	200
CSN B	- 20	- 20	+ 20	200
NCSN C	+ 20	+ 20	0	200

2. While the CSNs are started together, the NCSN node is not. The network cannot be started completely. The system will not get to the deadlock status.

Example of time settings for this case:

	ΔT_{start_A} [μT]	ΔT_{start_B} [μT]	ΔT_{cycle} [μT]	D [μT]
CSN A	0	0	- 20	200
CSN B	- 20	- 20	+ 20	200
NCSN C	+ 20	+ 220	0	200

3. The Q2 and Q3 queries, finding out whether there is a path to mutually start the CSNs and the NCSN, are fulfilled (*i.e.* there is a path enabling the complete SU of the whole network). However, the Q4 query is not fulfilled as for its fulfilment it requires that each path must lead to the full SU of the network. A set of timing parameters has been found where the success of the start-up procedure depends on which of the CSNs became leader. There is no deadlock in the system.

Example of time settings for this case:

	ΔT_{start_A} [μT]	ΔT_{start_B} [μT]	ΔT_{cycle} [μT]	D [μT]
CSN A	+ 120	+ 40	- 20	200
CSN B	- 40	- 80	+ 40	200
NCSN C	+ 40	+ 80	+ 80	200

In this case, if the CSN B is the leader, the network will be successful started. However, if the CSN A becomes leader, the network will not be able to finish its start.

4. The CSNs are never started and subsequently the NCSN node is not started either. The reason is given by a too high difference between the departure time and arrival time of a SUF. There is no deadlock in the system.

Example of time settings for this case:

	ΔT_{start_A} [μT]	ΔT_{start_B} [μT]	ΔT_{cycle} [μT]	D [μT]
CSN A	0	+ 130	0	140
CSN B	+ 80	- 120	0	140
NCSN C	+ 40	+ 80	+ 80	220

5. In the case of a problem in the timing parameters, for example the length of the CC is shorter than the arrival times of the start-up frames, not only can the network not be started, even in parts, but there is also a deadlock in the system.

Example of time setting for this case:

	ΔT_{start_A} [μT]	ΔT_{start_B} [μT]	ΔT_{cycle} [μT]	D [μT]
CSN A	+ 140	+ 140	- 199 800	140
CSN B	- 80	- 120	- 199 800	140
NCSN C	+ 40	+ 140	- 199 800	220

Note that this setting is nonsense because T_{start_A} or T_{start_B} is higher than T_{cycle}. Hence, a deadlock has occurred in the system.

6. Conclusions

This article describes a model of the start-up mechanism of the FlexRay communication standard, including the results of its partial verification by using timed automata and CTL. The results show that there may exist some incorrect settings in the system when the FlexRay network is not able to start and subsequently the car would not be started either. These incorrect settings can be caused by an error made by a designer during the network scheduling stage, by a fault in the synchronization mechanism or in the time base of a node. In future, the behavior of the model will be compared with a real system. All simulations in this work have been made for the worst-case, where the synchronization mechanism does not work at all.

Acknowledgment

This research is supported by the Czech Ministry of Education project No. 1M0568 - Josef Božek Research Center of Engines and Automotive Engineering.

References

[1] T. Hiaroka, S. Eto, O. Nishihara, H.Kumamoto: “Fault Tolerant Design for X-by-wire Vehicle”. *SICE 2004 Annual Conference*, Hokkaido Institute of Technology, Sapporo, Japan, vol. 3, Aug. 2004, pp. 1940–1945.

[2] H. Kopetz: *A Comparison of TTP/C and FlexRay*. Technical report 10/2001 Vienna University of Technology, Real-Time System Group.

[3] FlexRay Consortium, “FlexRay Protocol Specification v.2.1 Rev. A”, FlexRay Consortium, 2005.

[4] FlexRay Consortium, “FlexRay Electrical Physical Layer Specification v.2.1 Rev. A”, Flex Ray Consortium, 2005.

[5] UPPSALA University, AALBORG University: UPPAAL – integrated tool environment for modelling, validation and verification of real-time systems modelled as networks of timed automata. Available at: *www.uppaal.com*.

[6] G. Behrmann, A. David, K.G. Larsen: *A tutorial on Uppaal*. Department of Computer Science. Aalborg University, Denmark.

[7] R. Alur, D.L. Dill: “A theory of timed automata”. *Theoretical Computer Science*, vol. 126, no. 2, Apr. 1994, pp. 183–235.

[8] R. Alur, C. Courcoubetis, D.L. Dill: “Model-checking for realtime systems”. *5th Symposium on Logic in Computer Science*, 1990, pp. 414–425.

[9] J. Krakora, L. Waszniowski, Z. Hanzalek: “Timed Automata Approach to Distributed and Fault Tolerant System Verification”. *1st NeCTS Workshop, Networked Control Systems & Fault Tolerant Control*, 2005, pp. 45–50.

[10] J. Krakora, Z. Hanzalek: “Checking Real-Time Properties of CAN Bus by Timed automata”. *Advanced Control Theory and Applications*, 2003, pp. 130–134.

[11] J. Krakora, Z. Hanzalek: “Testing of Hybrid Real – time System Using FPGA Platform”. *IEEE Symposium on Industrial Embedded Systems* – IES 2006, Lyon, CNRS-ENS, 2006. CD-ROM

[12] K. Godary, P. Parrend, I. Augé-Blum: “Comparison and Temporal Validation of Automotive Real-Time Architectures”. *International Conference on Industrial Technology*, Hammamet, Tunisia, Dec. 2004.

[13] J. Malinský: “The application of the timed automata for FlexRay start-up testing”. *16th IMEKO TC4 Symposium*, Florence, pp. 425–430. CD-ROM

List of abbreviations used in this contribution

CAN – Controller Area Network
CAS – Collision Avoidance Symbol
CC – Communication Cycle
CSN – Cold-Start Node (can be CSNL as well as CSNF)
CSNF – Following Cold-Start Node
CSNL – Leader Cold-Start Node
CTL – Computation Tree Logic
FR – FlexRay
GC – Guard Condition
IC – Invariant Condition
NCSN – Non Cold-Start Node
SUF – Start-Up Frame
SUFA – Start-Up Frame from cold-start node A
SUFB – Start-Up Frame from cold-start node B
SUFO – Start-Up Frame from Other cold-start node
SU – Start-Up
SUM – Start-Up Mechanism
TA – Time Automaton
TDMA – Time Division Multiple Access
TTP/C – Timed Triggered Protocol
UPPAAL – Integrated tool environment for modelling, validation and verification of real-time systems modelled as networks of timed automata.