APPLICATION OF HARDWARE DESCRIPTION LANGUAGES TO SPECIFICATION OF THE POINT MODULE INTERLOCKING LOGIC

Kawalec Piotr [1], Mocki Jacek [2]

[1] *Warsaw University of Technology, Faculty of Transport, Traffic Engineering & Control Division, Poland, 00-662 Warszawa, ul. Koszykowa 75, tel. 48-22-660 75 85, pka@ it.pw.edu.pl*
[2] *Scott Wilson Kirkpatrick Co & Ltd, Poland, 02-516 Warszawa, ul. Rejtana 17 p. I, tel. 48-22-542 41 80, 82-88, jacek.mocki @scottwilson.com*

Summary There is proposal to apply hardware description languages HDL in this paper to specify control movement devices in transport. Languages HDL, mainly used to design specialised programmable and reprogrammable digital devices, allow both simple and precise ways to describe functionality as well as a structure of both combine and sequence control devices. Concurrent processing instructions, built into languages, allow describing hardware realisations of control algorithm processes.

1. INTRODUCTION

Together with the increased development of the technique are growing up more both new and technologically advanced signalling systems - old relay systems are being replaced by microprocessor systems which allow an increase in functionality and reliability of the system and abate the costs of maintaining the devices service.

The newest signalling systems using microprocessors realise an interlocking logic in software [1,3]. The interlocking logic algorithm is realised by description language code and then that is most often tested by means of any additional programs on the basis of the algorithms for testing interlocking logic. Industry computer has to realise functions of a backup, route setting, display of the current situation on the station, diagnostic etc at the same time.

Here is proposal of apply hardware description languages HDL to specify control movement devices in transport. Languages HDL, mainly used to design specialised programmable and reprogrammable digital devices, allow both simple and precise ways describe functionality as well as a structure of both combine and sequence control devices. Concurrent processing instructions, built into languages, allow describing hardware realisations of control algorithm processes.

As it will be shown below, application of HDL languages to specify control devices in transport allow to both simplify specification process and it is possible to lead a verification of designed device correctness using methods functional simulation by means of appropriate software.

As it will be shown below, application of HDL languages to specify control devices in transport allow to both simplify specification process and it is possible to lead a verification of designed device correctness using methods functional simulation by means of appropriate software. Specification control device grows in a hardware description language like computer program in a programming language as well as digital device design simulation is realised like carrying out of computer program. One of the most widespread hardware description languages is VHDL language being one of two standard HDL languages.

Here will be shown a process of carrying out algorithm specification and verification in VHDL language as an example of one of the railway control movement devices (signalling devices) – point module. Here will be also shown a description source code and possibilities of hardware description in both final state machine editor FSM and block diagram editor BDE of Active-HDL package made by Aldec Co. Here will be presented a full process of correctness verification of signalling module by means of a logical simulator with a special attention on different ways of test vectors full collection.

2. ASSUMPTIONS OF THE SIGNALLING SYSTEM

The whole signalling system uses a concurrent processing as a fast method of route settings. It supposes from the route setting point of view to divide stations on three kinds of modules as it is shown on figure 1. Track circuit module (It) is responsible for trains' detection; point module (Iz) is responsible for a logic part of point machine running, checking of the point position etc and signal module (S) responsible for logic part of signal operations during signalling processes.

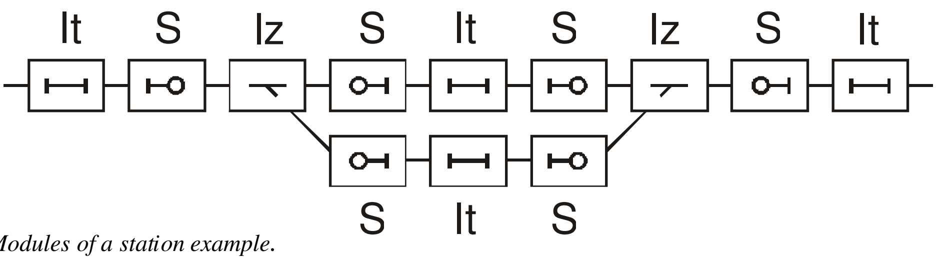


Fig.1. Modules of a station example.

The idea of the route setting is to use sequence operations inside modules blocking the concurrent processing in the route setting. Setting a route the signalman or computer system has to set an initial signal in the route then a final signal and approves the operation (control panel button). The route setting information is concurrent sending information into each module that the route has been chosen. Each module of the route starts its sequence process blocking operations prohibited by railway rules and realising route setting in accordance to railway standards. When the route has been set and initial signal on track is showing 'proceed' aspect the signalling system is waiting on a train passes. It is signalised by first track circuit occupation in the route.

System detects both correct trains passing releasing the route and failures appearing in the signalling system. The system interlocking logic of a point machine is supposed to be shown as an example of the railway interlocking function realisations.

3. ASSUMPTIONS OF THE POINT MACHINE MODULE

At the point module algorithm creation process the following assumptions have been established [2]:

- The logic of point module should realise all functions allowing set an appropriate route;
- Signals given into the point module should be clear that means they are safety;
- Immediately change state of point module to a failure state when a fault to be found;
- The current position of track is being checked with an interlocking table prepared separately for each station;
- The point logic inside the trackside point machine is responsible for the recognition of a point machine setting direction; The current position of track is being checked with an interlocking table prepared separately for each station;
- The interlocking table is required to set current situation of all signalling devices.

4. VARIABLES AND STATES TABLES

Setting of variables needed to specify logic point operations allows order kind of signals required a chart algorithm scheme (ASM) to be built. Variables and their description have been defined in the table below.

Symbol	Variable name	Kind of variable	Variable description
cur_pos	Current_position	input	Variable responsible for current position of the point. When 0 – the point is in its normal position, when 1 – it is in diverge position.
req_pos	Required_position	input	Interlocking table should send a signal of this variable to run the point module of interlocking. When 0 – the normal position required, when 1 – reverse position required
cor	Correct	input	If there has been noted a failure state of the point module of interlocking the correct variable should be responsible for sending into machine information that every fault has been repaired and checked. When 0 – there is still a fault noted, when 1 – the failure has been removed.
T1	Timer1	input	This variable gives a value informing system that required time was measured or not. When 0 – required time is being measured, when 1 – it has already been measured.
T2	Timer2	input	This variable gives a value informing system that required time was measured or not. When 0 – required time is being measured, when 1 – it has already been measured.

T1_cor	Timer1_correct	input	This variable is responsible for saying the timer works correctly or not and the time was measured correctly or not. When 0 – the timer doesn't work properly, when 1 – it measures time properly.
T2_cor	Timer2_correct	input	This variable is responsible for saying the timer works correctly or not and the time was measured correctly or not. When 0 – the timer doesn't work properly, when 1 – it measures time properly.
cha_pos	Change_position	input	When the position of point isn't agreed with the required position (point position of the setting route) there is necessary to change the position. The variable is responsible for do that. When 0 – don't change position, when 1 – change the point position.
poi_occ	Point_occupation	input	This variable gives information from track about detecting a train. When 0 – there is no any train detected on the point, when 1 – there is a train passing the point or there is a failure of the train detection system.
run_mac	Run_machine	output	It is an output signal directly sending information into the point machine. When 0 – there is no any action required, when 1 – it is require the point machine to be run.
run_T1	Run_timer1	output	This variable says that a signal of running point positioning timer has been run. When 0 – it isn't possible to run the timer, when 1 – timer is starting measuring time required the point to be set.
run_T2	Run_timer2	output	A variable starting timer responsible for measuring 120 second required to get back an initial state. When 0 - it isn't possible to run the timer, when 1 – timer is starting measuring the time of 120 second.

In accordance to assumptions of the signalling system and variables table it is possible to build a block diagram showing the point module

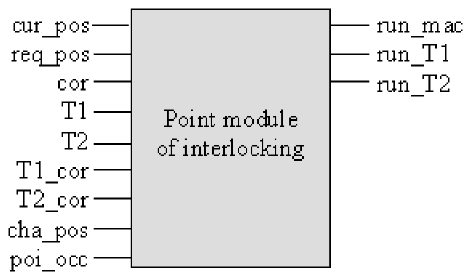


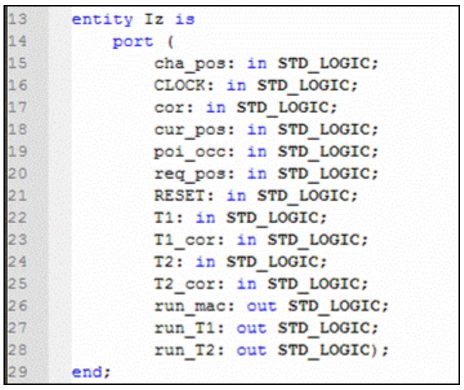
Fig.2. Block diagram of the point module.

5. FORMAL FUNCTIONS NOTATION

Algorithms of the signalling devices operation can be characterised in various way. The most popular ways of algorithms description are chart algorithm (ASM) and logic algorithm scheme (LSA). Figure 3 shows ASM of point module operation.

VHDL language, being a standard description language HDL allows specify control devices using behavioural, dataflow or structural state [1].

The specification consist in defined an entity. The entity characterises type, number and sort of pins required the specification to be realised.



```
entity Iz is
    port (
        cha_pos: in STD_LOGIC;
        CLOCK: in STD_LOGIC;
        cor: in STD_LOGIC;
        cur_pos: in STD_LOGIC;
        poi_occ: in STD_LOGIC;
        req_pos: in STD_LOGIC;
        RESET: in STD_LOGIC;
        T1: in STD_LOGIC;
        T1_cor: in STD_LOGIC;
        T2: in STD_LOGIC;
        T2_cor: in STD_LOGIC;
        run_mac: out STD_LOGIC;
        run_T1: out STD_LOGIC;
        run_T2: out STD_LOGIC);
end;
```

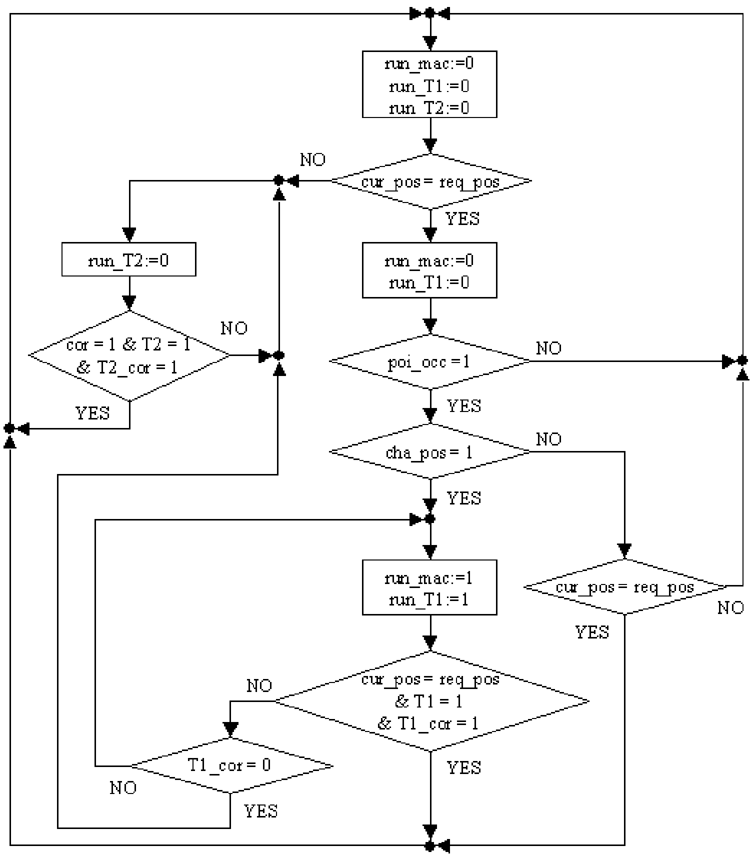
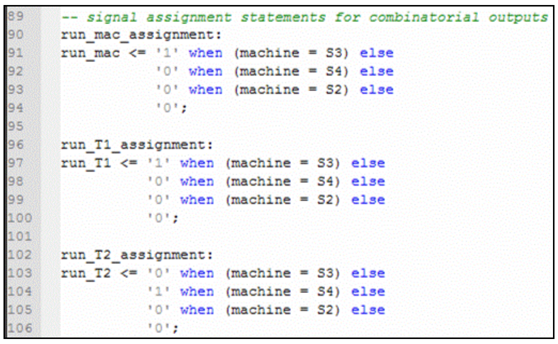


Fig.3. Chart ASM for the point module.

Then the entity makes device architecture inside one of the mentioned above styles. The best method of combination devices specification process leading is to use the dataflow style directly in a text editor of the language by means of concurrent processing signal instructions.

Because the outputs state is clearly given by the inputs state, for each of the outputs in the VHDL source code an attribute characteristic has been a specified using additive and multiplicity logical operator available in the language.

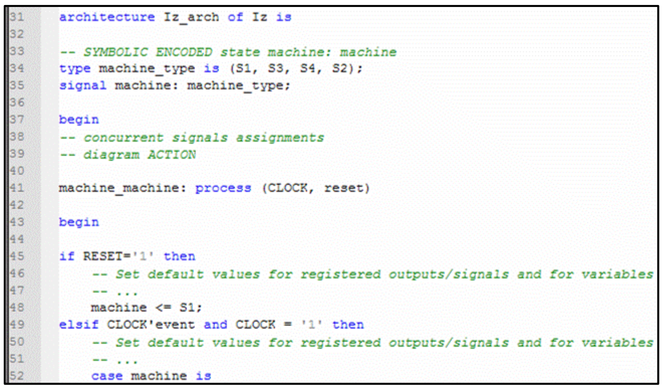


```
-- signal assignment statements for combinatorial outputs
run_mac_assignment:
run_mac <= '1' when (machine = S3) else
           '0' when (machine = S4) else
           '0' when (machine = S2) else
           '0';

run_T1_assignment:
run_T1 <= '1' when (machine = S3) else
          '0' when (machine = S4) else
          '0' when (machine = S2) else
          '0';

run_T2_assignment:
run_T2 <= '0' when (machine = S3) else
          '1' when (machine = S4) else
          '0' when (machine = S2) else
          '0';
```

However the best method to specify the sequence control devices in VHDL language is to use the behavioural state having the FSM editor and BDE editor (block diagram editor) as a base [3].



```
architecture Iz_arch of Iz is

-- SYMBOLIC ENCODED state machine: machine
type machine_type is (S1, S3, S4, S2);
signal machine: machine_type;

begin
-- concurrent signals assignments
-- diagram ACTION

machine_machine: process (CLOCK, reset)

begin

if RESET='1' then
    -- Set default values for registered outputs/signals and for variables
    -- ...
    machine <= S1;
elsif CLOCK'event and CLOCK = '1' then
    -- Set default values for registered outputs/signals and for variables
    -- ...
    case machine is
```

Especially, a possibility of a specification inside FSM (Finite State Machine) editor in the form of a state machine is very useful because description of the synchronous devices in the form of Moore or Mealy finite automaton is commonly using at traditional synthesis methods.

Thus the FSM editor is an environment of a control devices designer work [5].

Because the figure 3 algorithm can be realised by a sequence integrated circuit there is a Moore finite state machine used to its specification assuming 5 internal states (tab. 3).

State table ordering the state names used to the point machine be realised.

Tab. 3 States table

State name	State description
S1	Checking state
S2	Initial state
S3	Positioning state
S4	Failure state detection

Having ASM and state table it is possible to specify the signal module as a state machine which is a natural way of machines designing. There has been written in assumptions of the signalling system that the system uses a concurrent processing as a fast method of route settings. It means there is only one way to meet the requirement. It is necessary to use a computer aided design CAD software which is allowing specify the signalling system such way to use the specification to implement it into programmable logic devices PLD.

In this case Active-HDL software has been used to the point module be realised.

There are graphic model of the entity shown in the upper part of the figure 4 and architecture model placed in the form of the Moore finite automaton. In additional an input signal 'Reset' is provided allowing reset the machine (asynchronous machine transition into S1 state).

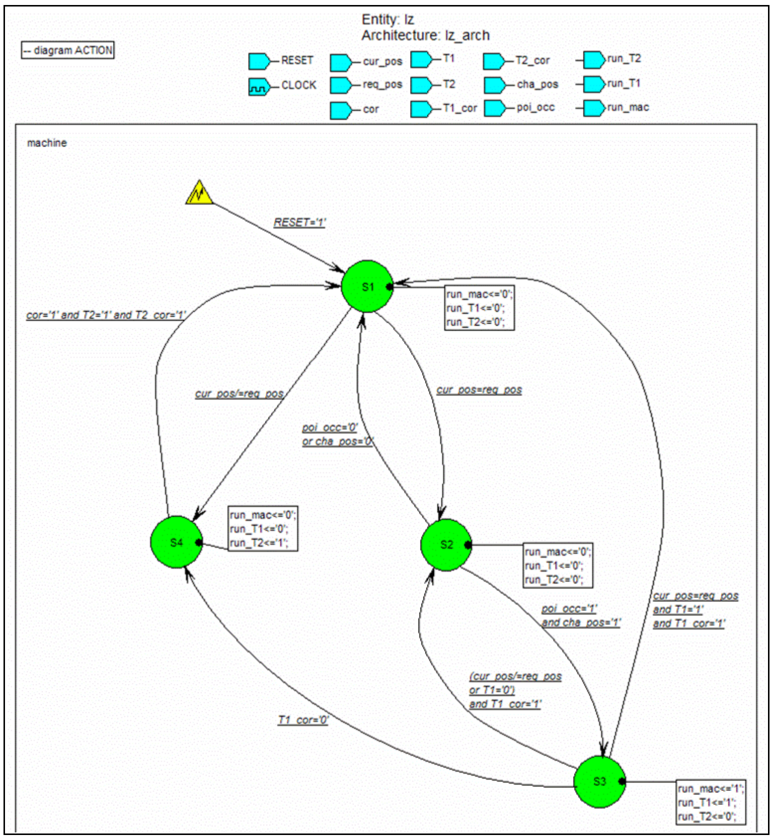

Entity: lz
Architecture: lz_arch
-- diagram ACTION
RESET
CLOCK
cur_pos
req_pos
cor
T1
T2
T1_cor
T2_cor
cha_pos
poi_occ
run_T2
run_T1
run_mac
machine
RESET='1'
S1
run_mac<='0'; run_T1<='0'; run_T2<='0';
cor='1' and T2='1' and T2_cor='1'
cur_pos/=req_pos
cur_pos=req_pos
poi_occ='0' or cha_pos='0'
S4
run_mac<='0'; run_T1<='0'; run_T2<='1';
S2
run_mac<='0'; run_T1<='0'; run_T2<='0';
poi_occ='1' and cha_pos='1'
cur_pos=req_pos and T1='1' and T1_cor='1'
(cur_pos/=req_pos or T1='0') and T1_cor='1'
T1_cor='0'
S3
run_mac<='1'; run_T1<='1'; run_T2<='0';

Fig.4. Finite State Machine for the signal module.

7. VERIFICATION OF THE STUDIED INTEGRATED CIRCUIT

Integrated circuit specified in any editor has been processed to the verification of which the algorithm work realisation has been checking for all combination of the input signals at all internal states of the integrated circuit.

The verification has been carrying out by means of right logic simulators. A logic correctness of the integrated circuit work can be checked using the simulators in functional simulation mode.

Input signals can be given manually aid of the stimulators or clocks with appropriate frequency can be assigned to each of the inputs.

The functional simulation results can be observed in the form of waveforms and there are changes of inputs and outputs showed without any delays existing in the integrated circuit (figure 5). It is possible to select any minute of the simulation on the waveforms that we would like to analyse values of the input/output signals.

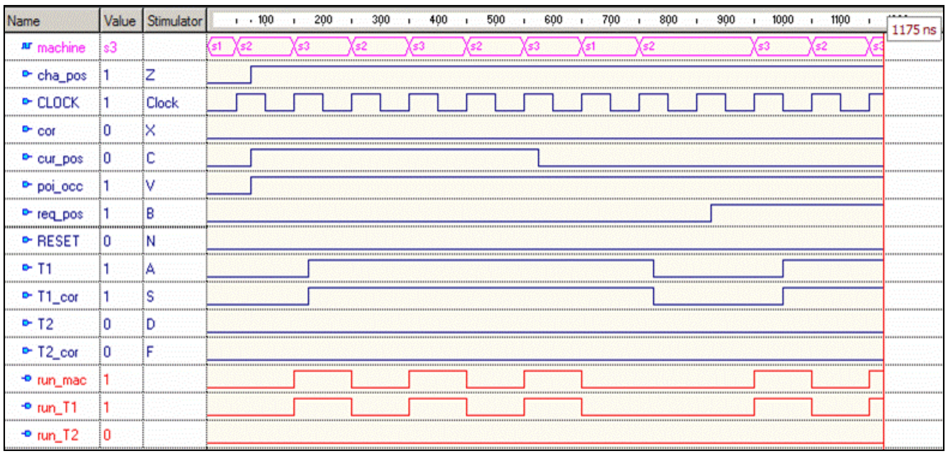


Fig.5. Simulation of the point module.

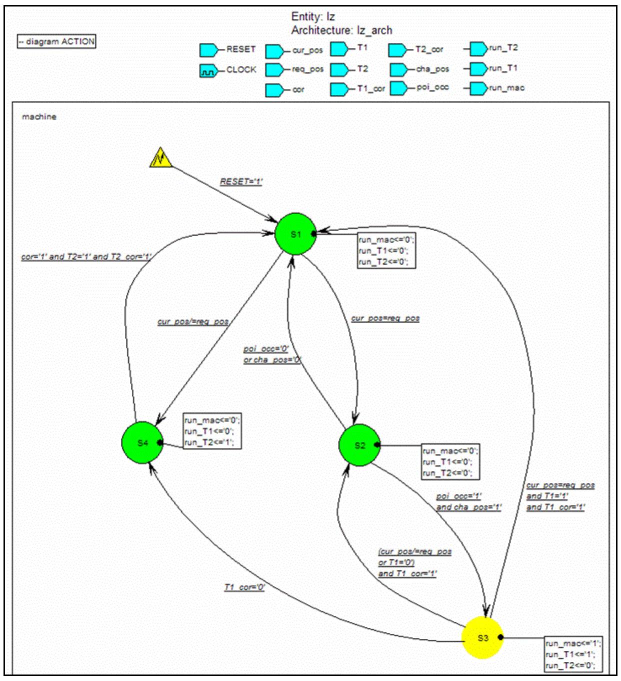

Entity: lz
Architecture: lz_arch
-- diagram ACTION
RESET
CLOCK
cur_pos
req_pos
cor
T1
T2
T1_cor
T2_cor
cha_pos
poi_occ
run_T2
run_T1
run_mac
machine
RESET='1'
S1
run_mac<='0'; run_T1<='0'; run_T2<='0';
cor='1' and T2='1' and T2_cor='1'
cur_pos/=req_pos
cur_pos=req_pos
poi_occ='0' or cha_pos='0'
S4
run_mac<='0'; run_T1<='0'; run_T2<='1';
S2
run_mac<='0'; run_T1<='0'; run_T2<='0';
poi_occ='1' and cha_pos='1'
cur_pos=req_pos and T1='1' and T1_cor='1'
(cur_pos/=req_pos or T1='0') and T1_cor='1'
T1_cor='0'
S3
run_mac<='1'; run_T1<='1'; run_T2<='0';

Fig.6. Finite State Machine for the point module during simulation.

The simulation process can be also observed directly on an automaton produced in the FSM editor because a colour of the active state of the automaton has been changed during simulation (state S4 on figure 6).

The characteristic point of the simulation showed on figure 5 and 6 presents that the state 3 is responsible for run the point machine when current position of point is in opposite to required point position.

8. BLOCK DIAGRAM EDITOR OF THE INTERLOCKING SYSTEM

Having designed two universal blocks - signal module [4] and point module, it is possible to set them on a block diagram editor as the part of railway station interlocking logic. To run the interlocking of railway station in programmable logic device it is necessary to design next universal blocks to fill up the gaps of interlocking logic modules shown on the picture below as empty blocks called Fub.

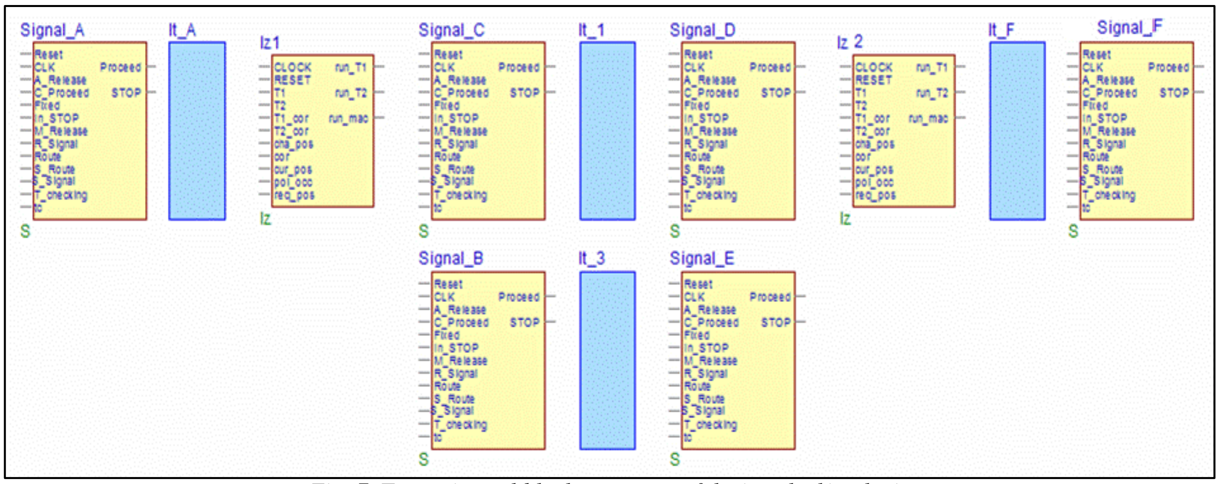


Fig. 7. Two universal blocks as a part of the interlocking logic.

REFERENCES

[1] Ashenden P.: *The designer's guide to VHDL*, Second Edition, Academic Press 2001

[2] Dabrowa-Bajon M.: *Basics of the signalling [in Polish]*. Publishing House of Warsaw University of Technology 2002

[3] Sobolewski K, Ossysek M.: *Getting Started Guide - Active-HDL TM Series*, Book #3, Aldec, Inc 1998

[4] Kawalec P., Mocki J.: *Application of hardware description languages to devices of controlling movement in transportation analyse and synthesis processes,* Proceedings of Symposium "Formal Methods for Automation and Safety in Railway and Automotive Systems FORMS/FORMAT 2004, Braunschweig, Germany, 2004, p 338 – 343

[5] Kawalec P., Mocki J.: *Specification and verification of the interlocking functions for signalling devices using hardware description languages HDL,* Proceedings of the 11th International Symposium "Zel 2004" Railways on the Edge of the 3rd Millennium "On the way towards the 'European' Railway- Harmonisation and IST", Zylina, Slovakia, 2004, p. 41 – 46