Creating VHDL Descriptions of Asynchronous Dual-rail RSFQ Logic

Ivan Dimitrov Panayotov, Tania Ilieva Stoiadinova
Krasimira Vasileva Filipova, Valeri Markov Mladenov, Thomas Ortlepp

Abstract – **This paper describes a principle approach to implement VHDL descriptions of dual-rail rapid single flux quantum circuits (RSFQ). Since RSFQ is naturally pulse driven, the dual-rail data coding is prompting the event driven data processing at very high speed. We describe methods and challenges to create corresponding VHDL models for simulation. The implementation is done with respect to a later implementation of a parameter driven optimization of complex logic cells**

Keywords –**Dual-rail data coding, RSFQ, asynchronous logic, behavior-based simulation, VHDL description**

I. INTRODUCTION

RSFQ LOGIC

Rapid single flux quantum (RSFQ) is a digital electronics technology based on the flux quantization in superconducting materials [1]. It uses transfer, processing and storage of single flux quanta to realize arbitrary logic functionality. It is naturally a digital technology, were signal values are represented by pulses instead of static levels. The data exchange is characterized by transient voltage pulses, which occur at defined time slots. In comparison to transistors in CMOS technology, the Josephson junctions are the active switching element in RSFQ technology.

There typical data coding is based on a synchronous clocking scheme, were the logic the '1' is represented by the presence of a pulse within the clock-cycle while the '0' by its absence. In contras, the dual-rail logic utilizes two signal lines where a pulse on the one represents a logical '1' and a pulse on the other represents a logical '0'. This data coding is completely asynchronous and the timing information is contained in the appearance of signals themselves [3]. RSFQ circuitry is essentially self clocking, making the asynchronous circuit designs much more practical. This kind new logic family can operates at very high frequencies above 100 GHz [2]. In this case, the gate delay is similar to the pulse propagation delay between cells, which makes the design of complex circuits very challenging.

HDL LANGUAGES

Modern hardware description languages such as VHDL and Verilog-HDL allow the creation of different digital elements and systems. Initially intended for simulations and analysis, now they provide the ability for automatic synthesis of electric circuits from register transfer level and behavioral description level. As these languages are IEEE standards all circuit descriptions are reusable.

VHDL is used to build descriptions of both synchronous and asynchronous circuits and provides different language constructs for implementing delays and timing. Even though this could make the descriptions non-synthesizable, the language is very suitable for the creation of timed dual-rail logic descriptions for simulations only.

The VHDL descriptions with timing parameters extracted from analog circuit simulations of individual cells provides digital models of these basic logic elements that can be used to build more complex logic circuits in order to speed up circuit analysis, enable the timing analysis and future parameter optimization.

II. VHDL DESIGN

There are two ways to describe the behavior of the dual-rail RSFQ logic element. In first one, the incoming pulses can be used as triggering events to start the internal generation of the events and output signals. The description using this approach is coded and simulated for a dual-rail AND gate. The symbol of the element is given on figure 1.

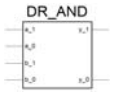

Figure 1 Dual-rail AND element symbol representation

Each one of the two inputs A and B are composed of two physical signal lines – one for the logical '1' and a second one for the logical '0'. A simplified truth table for the dual-rail AND gate is given in table 1.

TABLE 1. TRUTH TABLE OF DUAL-RAIL AND.

A_1	A_0	B_1	B_0	Y_1	Y_0
pulse	-	pulse	-	pulse	-
pulse	-	-	pulse	-	pulse
-	pulse	pulse	-	-	pulse
-	pulse	-	pulse	-	pulse
pulse	pulse	-	-	-	-
-	-	pulse	pulse	-	-

Ivan Panayotov - Faculty of Electronic Engineering and Technologies, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: idp@ecad.tu-sofia.bg

Tania Stoiadinova - Dept. Theoretical Electrical Engineering, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: tstoiadinova@abv.bg

Krasimira Filipova - Dept. Theoretical Electrical Engineering, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: KFilipova@abv.bg

Valeri Mladenov - Dept. Theoretical Electrical Engineering, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: valerim@tu-sofia.bg

Thomas Ortlepp – Institute of Information Technology, RSFQ design group, Ilmenau University of Technology, P.O. Box 100565, 98684 Ilmenau, Germany, e-mail: thomas.ortlepp@tu-ilmenau.de

The fragment for input signal capturing code is given in table 2. Here the "wait" operator is used to specify the delay and the width of each individual pulse of the output. In the simples case, the processing delay is common for all input signal combinations and relative timings.

TABLE 2. VHDL CODE PART DESCRIBING INPUT SIGNALS CAPTURING IN DUAL-RAIL AND.

```
  if(a_1'event and a_1 ='1') then  -- one on a
    if(flag_b_received = '1') then -- b already received
     assert (NOW - t_b_received) >= t_in report "Minnimum time between input pulses error!" severity ERROR;
      y_int <= value_b;
      flag_b_received <= '0';
      value_b <='0';
      gen_out <= not gen_out;
    else
      t_a_received <= NOW;
      value_a <= '1';
      flag_a_received <= '1';
    end if;
elsif(a_0'event and a_0 ='1') then-- zero on a
  …
elsif(b_0'event and b_0 ='1') then                -- zero on b
     if(flag_a_received = '1') then -- a already received
       assert (NOW - t_a_received) >= t_in report "Minnimum time between input pulses error!" severity ERROR;
        y_int <= '0';
        flag_a_received <= '0';
        value_a <='0';
        gen_out <= not gen_out;
     else
        t_b_received <= NOW;
        value_b <= '0';
        flag_b_received <= '1';
     end if;
end if;
```

A mechanism of checking the time between input pulses is provided by using the "assert" statement. In case of time violations an error message is displayed.

On each triggering event – rising edge of the input signal the check is made if a pulse on the other input was already received. The internal signal is used to store these information.

The connection between input signal conditions and the output generation is provided by internal flags that cause separate processes to be executed. The code fragment for these processes is presented in table 3. The process does not have a sensitivity list as it contains the "wait" statements. It is triggered on every change of the "gen_out" signal. The value of the output is generated depending on the value of the internal signal "y_int" – if it is '1' then a pulse on "y_1" line is formed, if it is '0' then the pulse is on "y_0" line. The pulse duration is fixed to 2 ps, but this can be be changed, depending in the circuit speed.

TABLE 3. VHDL CODE PART DESCRIBING THE OUTPUT SIGNAL GENERATION IN A DUAL-RAIL AND GATE.

```
process
begin
wait on  gen_out;
 if(y_int ='1') then
    wait for t_out;
    y_1 <= '1';  wait for 2 ps;  y_1 <= '0';
 elsif(y_int ='0') then   wait for t_out;
    y_0 <= '1';  wait for 2 ps;   y_0 <= '0';
 else
     y_0 <= '0';   y_1 <= '0';
 end if;
end process;
```

Here the delay and the pulse width are implemented using the "wait" statement. This makes the code completely non-synthesizable.

The simulation results of the AND gate are given on figure 2.

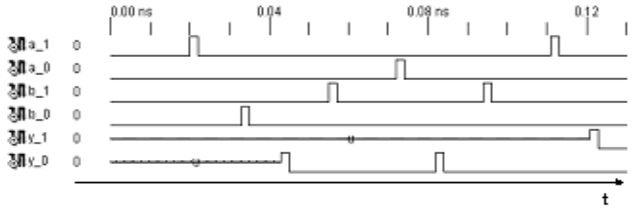

Figure 2 Simulation results for dual-rail AND. Time is in ns.

The output delay is measured from the rising edge of the last received input signal. In this version of the code there is no dependency on input signal order, but they are not allows to appear at the same time, as this will violate the "minimum time between pulses" condition and will result in an error message. Initially the output lines are in unknown state, while first value is assigned.

The second approach that can be used is to have a single process that implements all code logic and to use delay mechanisms provided by the language. This makes it easy to provide separate output delays for each input combination, thus allowing more sophisticated control and time tuning. The build-in the VHDL transport delay is used. This kind of delay is specified by using the key word "transport" in signal values assignments. It defines the time required for the signal to pass through a device or a transmission line. It is assigned upon every event on the signal. This allows describing the actual behavior of the circuit more accurate and close to real case. The delay mechanism in VHDL is illustrated on figure 3. The signal in the output repeats the input after a time required to pass through the device. On each event – rising edge or falling edge, delay time is 't'.

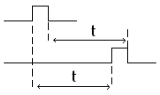

Figure 3 Transport delay mechanism in VHDL

In table 4 a part of the code that shows beginning of the process statement, the wait statement that captures changes of the input signals and output signals conditional assignments is given.

TABLE 4. VHDL CODE PART DESCRIBING BEGINNING OF THE PROCESS, INPUT SIGNALS CAPTURING AND OUTPUT SIGNALS ASSIGNMMENT IN DUAL-RAIL AND.

```
process
 begin
 wait on a_1, a_0, b_1, b_0;
if(a_1'event and b_1'event) then
   if (a_1 = '1' and b_1 = '1') then
      y_1 <= transport (a_1 and b_1) after t_out_ab_e_11;
   else
      y_1 <= transport '0' after t_out_ab_e_11;
   end if;
elsif (a_1'event  and b_0'event ) then -- one on a and zero on b at same time
…
elsif(a_0'event ) then
```

```
   if (a_0 ='1') then
      if(flag_b_received = "11") then
      assert (NOW - t_b_received) >= t_in report "Minnimum time
between input pulses error!" severity ERROR;
       y_0 <= transport a_0 after t_out_a_a_b_01;
       flag_b_received <= "01";
…
```

To implement the transport delay the assignment to the output signals should be done on every change of the input signal. For this reason the conditional "if" statements are used to check for the specific events on inputs. Using only "event" condition provides execution of the desired statements on every change of the signal. Thus when a pulse comes on an input and triggers a change of the output, the transport delay will be applied.

The results from simulations are presented on figure 4. Here again the outputs are in undefined state until first meaningful values are outputted.

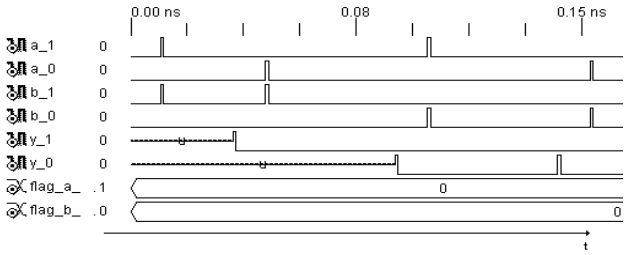

Figure 4 Simulation results for dual-rail AND gate by using the second code version. Time is in ns.

The logic conditions are implemented in the "if" statements which makes the description behavior. Even though the output pulse is formed by the input one, its presence on the line for logic '1' or '0' depends on the execution of the "if" statement.

Using the same approach and methods for code creation for a dual-rail XOR gate is described in VHDL and simulated. A code fragment for the case using transport delay is given in table 5.

TABLE 5. VHDL CODE PART OF DUAL-RAIL XOR.

```
process begin
 wait on a_1, a_0, b_1, b_0;
if(a_1'event and b_1'event) then
   if (a_1 = '1' and b_1 = '1') then
      y_0 <= transport (a_1 and b_1) after t_out_ab_e_11;
   else       y_0 <= transport '0' after              t_out_ab_e_11;
   end if;
elsif (a_1'event  and b_0'event ) then
  if (a_1 = '1' and b_0 = '1') then
       y_1 <= transport (a_1 and b_0) after t_out_ab_e_10;
  else       y_1 <= transport '0' after t_out_ab_e_10;
  end if;
```

The simulation results for XOR element are presented on figure 5.

The received results represent the actual behavior of dual-rail logic elements in a simplified configuration. The timing characteristics can easily be tuned to actually reflect those of practical RSFQ cells, as they are specified by using parameters.

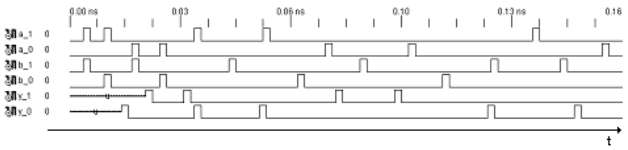

Figure 5 Simulation results for dual-rail XOR gate. Time is in ns.

III. CONCLUSIONS

The way of creating VHDL description of dual-rail asynchronous RSFQ logic cells is presented. Two different approaches are considered and compared. The results prove the ability of VHDL to be used for creating asynchronous descriptions for the RSFQ elements. The developed codes can be used for the analysis of more complex logic devices that can be simulated behaviorally and speed up the circuit analysis. Some of the considerations presented here can also be used in descriptions of asynchronous circuits in general. The practical impact of this method is related to behaviour-models including parameterized timing information. The practical processing delay depends on circuit parameters a optimization process of the overall circuit timing is essential to enable the correct circuit operation of complex systems [7]

IV. ACKNOWLEDGMENTS

The research described in this paper was carried out within the framework of contracts D002-106/15.12.2008 and D002-126/15.12.2008.

REFERENCES

[1] K.K. Likharev and V.K. Semenov, *"RSFQ Logic/Memory Family: A New Josephson-Junction Digital Technology for Sub-Terahertz Clock-Frequency Digital Systems"*, IEEE Trans. on Applied Superconductivity, vol. 1, pp. 3-28, 1991

[2] H. Akaike et al., *"Demonstration of a 120 GHz single-flux quantum shift register circuit based on a 10 kA/cm² Nb process"*, Superconductor Science and Technology, Vol. 19, pp. S320-S324, 2006

[3] B. Dimov, Th. Ortlepp, V. Mladenov, S. Terzieva, and F. H. Uhlmann. *The asynchronous rapid single-flux quantum electronics – a promising alternative for the development of high-performance digital circuits*, Advances in Radio Science, 2008

[4] V. Mladenov, V. Todorov, B. Dimov, Th. Ortlepp, F. H. Uhlmann,. *"High-Level Design of Asynchronous RSFQ Digital Circuits"*. in proceedings of the 51. International Scientific Colloquium, Ilmenau University of Technology, September 11-15, 2006

[5] Ashenden P.J. *"The Designer's Guide to VHDL"*, Second Edition, Morgan Kaufmann Publishers, San Francisco 2001.

[6] Nancheva-Philipova K., M.Hristov, I. Panayotov, V. Hristov, *"Use of (v)HDL for electronic devices synthesis" – reference book* , KING 2001, Sofia 2004

[7] A. Fujimaki et al., *"Demonstration of 2x3 regonfigurable data-path processors with 14.000 Josephson junctions"*, Superconducting SFQ VLSI workshop 2009, Fukuoka, Japan