Synchronous and asynchronous structural implementation of Łukasiewicz norms in Spartan-6 FPGAs

Abstract

Fast time to market, high performance and low cost make new FPGAs a competition for dedicated VLSI device in many area. Their array architecture with lots of programmable resources and IO pins is attractive hardware platform for implementation a complex fuzzy systems. The article discusses the realization of fuzzy Łukasiewicz operations in Xilinx Spartan-6 FPGAs, which in addition to Zadeh operations, are basic elements in fuzzy systems. Safe behavioral description of these operations that define functionalities independent of the hardware platform are presented. Structural descriptions of both synchronous and asynchronous fuzzy operations are shown, to carry out their primitive level realization and the effective utilization of basic elements of the FPGA structure. As the result the area optimized implementation of Łukasiewicz operations are obtained.

Keywords: fuzzy hardware, fuzzy Łukasiewicz norms, FPGA.

1. Introduction

Fuzzy combinational circuits with semantics based on \([0, 1]\) interval of real number have been created by replacing the classical conjunction, disjunction and alternative by t-norm, t-conorm, and strong negation. To generalize the classical systems classical conjunction, disjunction and alternative by t-norm, interval of real number have been created by replacing the logic low state corresponded to the series of zeros and the logic high state to a series of ones. This allowed to obtain optimal utilization of hardware resources, compared with the use of behavioral descriptions.

2. Behavioral realization

FPGAs can be designed with a schematic or a hardware description language (VHDL, Verilog). Projects based on a HDL may be constructed in a behavioral or structural style. A behavioral description reflect functionality of the device. It has no direct reference to physical resources (e.g. a specific LUT or multiplexer) so that it can be transferred to any hardware platform with no changes. Necessary FPGA resources are established upon a code during a synthesis and implementation process.

Functional diagrams of fuzzy Łukasiewicz operations from equation (3) and (4) are shown in Fig. 1.

![Fig. 1. a) simplified diagram of Łukasiewicz sum, and b) product diagram](image-url)

The synthesizable VHDL code, describing the behavioral architecture of Łukasiewicz t-norm and t-conorm, programmed according to the diagrams from Fig. 1 is shown in Listing 1. The parameter \(n\) reflects the bit resolution.

Listing 1. The behavioral description of generic Łukasiewicz t-norm and t-conorm

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Tnorm is
    Generic(n: positive :=4);
    Port (A : in STD_LOGIC_VECTOR (n-1 downto 0);
          B : in STD_LOGIC_VECTOR (n-1 downto 0);
          Q : out STD_LOGIC_VECTOR (n-1 downto 0));
end Tnorm;

architecture Behavioral of Tnorm is
    constant Jeden:std_logic_vector(n-1 downto 0):=Others=>'1';
    constant Zero :std_logic_vector(n-1 downto 0):= Others=>'0';
begin
    process (A,B)
    variable suma: std_logic_vector(n downto 0);
    constant a: natural := 4;
    constant b: natural := 4;
    constant Q: std_logic_vector(n downto 0);
    constant A: std_logic_vector(n downto 0);
    constant B: std_logic_vector(n downto 0);
    constant Jeden:std_logic_vector(n-1 downto 0):=Others=>'1';
    constant Zero :std_logic_vector(n-1 downto 0):= Others=>'0';
    end process;
end Behavioral;
```

Modern FPGAs, with lots of programmable logic and attractive timing performance make an alternative to dedicated VLSI device in implementation of discrete fuzzy systems. Examples of schematic implementation of Łukasiewicz norms in the FPGA can be found in [7, 8, 9], where these norms were used for the construction of fuzzy flip-flops and fuzzy Petri nets.
implementation of CNorm is:

\begin{align*}
\text{constant } \text{Jeden:std_logic_vector(n-1 downto 0):=(others=>'1');}
\end{align*}

\begin{align*}
\begin{aligned}
\text{begin} & \\
\text{process (A,B)} & \\
\text{begin} & \\
\text{suma := conv_std_logic_vector((conv_integer(A) + conv_integer(B)),n+1);} & \\
\text{if suma >= Jeden then} & \\
\text{suma := conv_std_logic_vector((conv_integer(suma) – conv_integer(Jeden)),n+1);} & \\
\text{Q <= suma(n-1 downto 0);} & \\
\text{else} & \\
\text{Q <= (others => '0');} & \\
\text{end if;} & \\
\text{end process;} & \\
\text{end Behavioral;} & \\
\end{aligned}
\end{align*}

Fig. 2. Spartan-6 FPGA implementation of one bit full adder [10]

A propagation group \( p_i \) is generated in Look-Up Table (LUT) - the functions generator. A sum bit \( s_i \) is created in a dedicated XORCY gate from a propagation group and a carry of the previous bit. The carry bit \( c_i \) is formed in the MUXCY, that is part of carry logic and implements the function:

\begin{align*}
\text{c}_i = \text{a}_i \text{xor } \text{b}_i \text{c}_{i-1} + \text{a}_i \text{c}_{i-1} \text{b}_i. \quad (13)
\end{align*}

Full adders are connected to form multi-bit ripple-carry adders by fast carry chain (Fig. 2) being much faster than a switch matrix.

Using the carry chain, a fuzzy Łukasiewicz sum and a product can be constructed based on Eqs (9) and (10) in the form of an adder having at the outputs the OR gate (for t-conorm) or AND gate (for t-norm) which is presented in Fig. 4. The most significant bit of the sum \( s_n \), imposing appropriate limits, corresponds to a signal COUT from bit \( n-1 \).

Fig. 4. a) Logical diagram of 4-bit Łukasiewicz t-conorm, and b) t-norm diagram

<table>
<thead>
<tr>
<th>Formula</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a = a_{n-1}2^{n-1} + \cdots + a_22^2 + a_12^1 + a_02^0 )</td>
<td>Equation (6)</td>
</tr>
<tr>
<td>( b = b_{n-1}2^{n-1} + \cdots + b_22^2 + b_12^1 + b_02^0 )</td>
<td>Equation (7)</td>
</tr>
<tr>
<td>( s = a + b = s_n2^n + s_{n-1}2^{n-1} + \cdots + s_22^2 + s_12^1 + s_02^0 )</td>
<td>Equation (8)</td>
</tr>
<tr>
<td>( a \oplus b = a + b \quad \text{if } s &lt; 10000_2 = \begin{cases} a + b &amp; \text{if } s = 0_2 \ 1111_2 &amp; \text{if } s = 1_2 \end{cases} \quad (9) )</td>
<td></td>
</tr>
<tr>
<td>( a \otimes b = a + b - 1111_2 \quad \text{if } s \geq 10000_2 = \begin{cases} 0000_2 &amp; \text{if } s &lt; 10000_2 \ 1111_2 &amp; \text{if } s = 1_2 \end{cases} \quad (10) )</td>
<td></td>
</tr>
</tbody>
</table>

It can be seen that the result is a product of \( n-1 \) less significant bits of the sum \( s_{n-1} \ldots s_1, s_0 \), and the most significant bit \( s_n \) imposes appropriate limits.

Adders in Spartan-6 FPGAs are implemented in Configurable Logic Blocks (CLB) as a ripple-carry adders consisted of full adders (Fig. 2), whose operation is characterized by the equations:

\begin{align*}
\begin{aligned}
\text{si} & = \text{ci}_i \text{xor } \text{pi}_i, \quad \text{ci} = \text{gi}_i + \text{ci}_{i-1} \text{pi}_i, \quad (11) \\
\text{pi} & = \text{a}_i \text{xor } \text{b}_i, \quad \text{gi} = \text{a}_i \text{b}_i. \quad (12)
\end{aligned}
\end{align*}
In the fuzzy product, described by the equation (10), addition of $1_2$ and subtraction of $1000_2$ is realized in Fig. 3 by connecting the input CIN of the adder to a high logic state and omission the $\bar{s}_4$ bit. Diagram realizations of Łukasiewicz norms, build with adders and multiplexers can be found in [10]. A generic code of such a Łukasiewicz t-norm and t-conorm is presented in Listing 2.

Listing 2. A Behavioral architecture of generic Łukasiewicz t-norm and t-conorm based on Eqs (9) and (10)

```
architecture Behavioral of Tnorm is begin
    process (A,B)
        variable suma: std_logic_vector(n downto 0);
        begin
            suma := conv_std_logic_vector((conv_integer(A) +
            conv_integer(B) + conv_integer('1')),n+1);
            if suma(n) = '0' then
                suma(n-1 downto 0) := (others => '0');
                end if;
                Q <= suma(n-1 downto 0);
                end process;
    end Behavioral;
end Behavioral of Cnorm is begin
    P1: process (A,B)
        variable suma: std_logic_vector(n downto 0);
        begin
            suma := conv_std_logic_vector((conv_integer(A) +
            conv_integer(B)),n+1);
            if suma(n) = '1' then
                suma(n-1 downto 0) := (others => '1');
                end if;
                Q <= suma(n-1 downto 0);
                end process;
    end Behavioral;
```

Gates at the output of the adder may be replaced by flip-flops to receive the synchronous realization (Fig. 5).

```
3. Structural realization

A behavioral code is more intuitive, but not always produces an optimum and desired structure. For example, for synchronous norms shown in Fig. 4 flip-flops may be placed in the same slices as adders. This significantly reduces occupied resources - for $n$-bit Łukasiewicz norm to the area of adder ($n/4$ slice). The specific structure can be obtained with a structural style.

The structural description is composed of instances of components and connections between them reflecting the device’s diagram. Components can be functional and logical blocks or physical models of system elements supplied by the manufacturers in the form of a primitives library.

A structural code of $n$-bit t-conorm for Spartan-6 device is presented in Listing 4.

Listing 4. The structural description of generic synchronous Łukasiewicz t-conorm for Spartan-6 FPGA

```
LIBRARY IEEE;
LIBRARY UNISIM;
USE IEEE.STD_LOGIC_1164.ALL;
USE UNISIM.VCOMPONENTS.ALL;
entity CnormSynchS is
    Generic(n: positive := 8;
    lutInit: bit_vector := x"6");
    Port ( A : in  STD_LOGIC_VECTOR (n-1 downto 0);
    B : in  STD_LOGIC_VECTOR (n-1 downto 0);
    CLK_IN: in  STD_LOGIC;
    Q : out  STD_LOGIC_VECTOR (n-1 downto 0));
end CnormSynchS;
architecture Structural of CnormSynchS is
    signal carry : std_logic_vector(n downto 0);
    signal lutOut : std_logic_vector(n downto 0);
    signal xorOut : std_logic_vector(n-1 downto 0);
    attribute RPM_GRID: string;
    attribute RPM_GRID of CnormSynchS : entity is "GRID";
    end CnormSynchS;
```

Listing 3 describes behavioral code of a synchronous t-norm and t-conorm synchronized by a clock CLK_IN.

```
LISTING 3. A Behavioral architecture of generic synchronous Łukasiewicz t-norm and t-conorm

entity TnormSynch is
    Generic(n: positive := 4);
    Port ( A : in  STD_LOGIC_VECTOR (n-1 downto 0);
    B : in  STD_LOGIC_VECTOR (n-1 downto 0);
    CLK_IN: in  STD_LOGIC;
    Q : out  STD_LOGIC_VECTOR (n-1 downto 0));
end TnormSynch;
architecture Behavioral of TnormSynch is begin
    process (CLK_IN)
        variable suma: std_logic_vector(n downto 0);
        begin
            if rising_edge(CLK_IN) then
                suma := conv_std_logic_vector((conv_integer(A) +
            conv_integer(B) + conv_integer('1')),n+1);
            if suma(n) = '1' then
                Q <= suma(n-1 downto 0);
            else
                Q <= (others => '0');
            end if;
        end if;
    end process;
end Behavioral;
architecture Behavioral of CnormSynch is begin
    process (CLK_IN)
        variable suma: std_logic_vector(n downto 0);
        begin
            if rising_edge(CLK_IN) then
                suma := conv_std_logic_vector((conv_integer(A) +
            conv_integer(B)),n+1);
            if suma(n) = '1' then
                Q <= (others => '1');
            else
                Q <= suma(n-1 downto 0);
            end if;
        end if;
    end process;
end Behavioral;
```

Fig. 5. a) Logical diagram of 4-bit synchronous Łukasiewicz t-conorm, and b) t-norm
The main element of a structural t-conorm description is for... generate loop replicating \( n \)-times a one bit full adder shown in Fig. 2 with a flip-flop at the output to form a ripple-carry adder. Single-bit full adder is composed of elements labeled xorLut, xorSum, carryMux. For the creation of a propagation group \( \mathcal{P}_n \) (12) a LUT is initiated according to the truth table of Xor function with hexadecimal value x"6" (Tab. 1).

Tab. 1. The truth table of Xor function

<table>
<thead>
<tr>
<th>( a_1 )</th>
<th>( a_0 )</th>
<th>( q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Attributes RLOC (Relative Location) enforce proper distribution of instantiated primitives to each other. To locate objects the RPM GRID system is used. The Y value of attributes RLOC is integer(4*(\( i/4 \))) which deploys adders in a column and allows to use the carry chain. This value is a result of the assumed grid system and a structure of Spartan-6 carry chain in CLB. Outputs of the ripple-carry adder are terminated with flip-flops labeled flipFlopD as in Fig. 5. In the Spartan-6 there is no need to set the initial state of flip-flops (or latches), because it is closely related to the used set/reset signal. The area occupied by t-conorm is \( n/4 \) slices.

A structural code of \( n \)-bit t-norm for Spartan-6 FPGA is shown in Listing 5.

Listing 5. The structural architecture of generic synchronous \( \Lukasiewicz \) t-norm for Spartan-6 FPGA

```vhdl
architecture Structural of TnormSynchS is
  signal carry : std_logic_vector(n downto 0);
  signal lutOut : std_logic_vector(n-1 downto 0);
  signal xorOut : std_logic_vector(n-1 downto 0);
  attribute RLOC : string;
  attribute RLOC of inverter: label is "X0Y" & integer'image(integer(4*(n-1)/4));
begin
  carry(0) <= '1';
  L1:
    for i in 0 to n-1 generate
      attribute RLOC of xorLut: label is "X0Y" & integer'image(integer(4*(i/4)));
      attribute RLOC of xorSum: label is "X0Y" & integer'image(integer(4*(i/4)));
      attribute RLOC of carryMux: label is "X0Y" & integer'image(integer(4*(i/4)));
      begin
        xorLut : LUT2
          generic map (INIT => lutInit)
          port map (O => lutOut(i), I0 => A(i), I1 => B(i));
        xorSum : XORCY
          port map(O => xorOut(i), CI => carry(i), LI => lutOut(i));
        carryMux : MUXCY
          port map(O => carry(i+1), CI => carry(i), DI => A(i), S => lutOut(i));
        flipFlopD: FDS
          port map(Q => Q(i), C => CLK_IN, D => xorOut(i), S => carry(n));
      end generate;
    end Structural;
end Structural;
```

Flip-flops in Spartan-6 CLB have only high an active signal level on set/reset port. T-norm in Fig. 5 requires active low level on the flip-flop’s reset so that additional instance of the inverter for a carry(\( n \)) signal is necessary. This inverter can be placed in the same LUT as a xorLut instance because LUTs in Spartan-6 have two outputs. The last slice configuration of 8-bit synchronous t-norm is presented in Fig. 6.

An asynchronous realization of \( \Lukasiewicz \) norms can be obtained by replacing gates (Fig. 4) at the adder’s outputs with latches. Listing 6 shows a structural code of asynchronous t-conorm with latches for Spartan-6 device.

Listing 6. The structural architecture of generic asynchronous \( \Lukasiewicz \) t-norm for a Spartan-6 FPGA

```vhdl
architecture Structural of CnormS is
  signal carry : std_logic_vector(n downto 0);
  signal lutOut : std_logic_vector(n-1 downto 0);
  signal xorOut : std_logic_vector(n-1 downto 0);
  attribute RLOC : string;
begin
  carry(0) <= '0';
  L1:
    for i in 0 to n-1 generate
      attribute RLOC of xorLut: label is "X0Y" & integer'image(integer(4*(i/4)));
      attribute RLOC of xorSum: label is "X0Y" & integer'image(integer(4*(i/4)));
      attribute RLOC of carryMux: label is "X0Y" & integer'image(integer(4*(i/4)));
      attribute RLOC of flipFlopD: label is "X0Y" & integer'image(integer(4*(i/4)));
      xorLut : LUT2
        generic map (INIT => lutInit)
        port map (O => lutOut(i), I0 => A(i), I1 => B(i));
      xorSum : XORCY
        port map(O => xorOut(i), CI => carry(i), LI => lutOut(i));
      carryMux : MUXCY
        port map(O => carry(i+1), CI => carry(i), DI => A(i), S => lutOut(i));
      flipFlopD: FDS
        port map( Q => Q(i), C => CLK_IN, D => xorOut(i), R => notCarryN); 
    end generate;
  end Structural;
end Structural;
```
integer'image(integer(4*(i/4))); 
attribute RLOC of xorSum: label is "X0Y" &
integer'image(integer(4*(i/4))); 
attribute RLOC of carryMax: label is "X0Y" &
integer'image(integer(4*(i/4))); 
attribute RLOC of latch: label is "X0Y" &
integer'image(integer(4*(i/4))); 

begin 
xorlut : LUT2 
generic (INIT => lutInit) 
port map (O => xorOut(i), I0 => A(i), I1 => B(i));
xorSum : XORCY 
port map (O => xorOut(i), CI => carry(i), 
I1 => lutOut(i));
carryMax : MAXCY 
port map (O => carry(i+1), CI => carry(i), 
I1 => A(i), I2 => lutOut(i));
latch : LDP 
port map (Q => Q(i), D => xorOut(i), G => '1', 
PRE => carry(i));
end generate;
end Structural;

Latches should be used very carefully because for the Spartan-3A FPGA it was observed in the time simulation (after place and route - PAR) that short change at a set/reset signal, caused by propagation delays, could entail unstable condition (transient at their output). In Spartan-6 time simulations similar phenomenon was not detected.

A structural description allows to achieve the implementation that uses precisely planned area. However, it is much more difficult, less intuitive and therefore prone to errors. It is also platform-dependent. For example, the implementation of t-norm and t-conorm in a Spartan-3A requires a modification of the method of calculating the relative location of elements. In the case of t-norm this FPGA can absorb the inverter in flip-flop and change it to low-level triggered set/reset signal.

4. Test results

The correct operation of the implementation of fuzzy Łukasiewicz norms was confirmed during functional (behavioral) and timing (after PAR) simulation. Listing 7 illustrates testbench file used in a timing simulation of 4-bit synchronous t-conorm.

Listing 7. The testbench file for timing simulation of 4-bit synchronous t-conorm.

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.NUMERIC_STD.ALL;
USE IEEE.STD_LOGIC_TEXTIO.ALL;
USE IEEE.STD_TEXTIO.ALL;
entity TimingTestKonormySynch is
end TimingTestKonormySynch;
architecture behavioral of TimingTestKonormySynch is
component CnormSynchS is
  attribute RLOC of xorSum: label is "X0Y" &
  integer'image(integer(4*(i/4)));
  attribute RLOC of carryMax: label is "X0Y" &
  integer'image(integer(4*(i/4)));
  attribute RLOC of latch: label is "X0Y" &
  integer'image(integer(4*(i/4)));
begin
  xorSum : XORTUL
  generic (INIT => lutInit)
  port map (O => xorOut(i), I0 => A(i), I1 => B(i));
xorSum : XORCY
  port map (O => xorOut(i), CI => carry(i), 
  I2 => latchOut(i));
carryMax : MAXCY
  port map (O => carry(i+1), CI => carry(i), 
  I2 => A(i), I3 => latchOut(i));
latch : LDP
  port map (Q => Q(i), D => xorOut(i), G => '1', 
  PRE => carry(i));
end generate;
end Structural;

variable tx_out: line;
variable errorCounter: integer := 0;

begin
  wait for 5 * Period;
  for i in 0 to 2**n-1 loop
    for j in 0 to 2**n-1 loop
      wait for Delay;
      A <= conv_std_logic_vector(i,n);
      B <= conv_std_logic_vector(j,n);
      if (i+j) < (2**n-1) then
        result := conv_std_logic_vector(i+j,n);
      else
        result := conv_std_logic_vector(2**n-1,n);
      end if;
    end loop;
    if Q /= result then
      wait for 3 * Delay;
      if Q /= result then
        write(tx_out,string('"blad!"'));
        write(tx_out,string('" wyn: ");
        write(tx_out,string('" a: ");
        write(tx_out,string('" b: ");
        write(tx_out,string('" wyn: ");
        write(tx_out,result);
        errorCounter := errorCounter + 1;
        assert false report "blad!" severity error;
      end if;
    end loop;
    errorCounter := errorCounter + 1;
    if errorCounter > 0 then
      assert false report "Process ends with errors !!!" severity failure;
    else
      assert false report "No errors detected" severity note;
    end if;
    end process;
  end loop;
end;

In the prepared testbench an instance of t-conorm UUT (unit under test) was created, generated clock signal and test vectors, and formed a validation routine that checks the correctness of responses with appropriate time delays. During the test any detected errors are drawn up to the report text file.

In Tab. 2 and Tab. 3 the area occupied by asynchronous implementations of Łukasiewicz t-conorm and t-norm in a Spartan-6 is summarized. In all cases a structural description gave the best results.

Tab. 2. The area occupied by asynchronous Łukasiewicz t-conorm in Spartan-6

<table>
<thead>
<tr>
<th>T-conorm with architecture</th>
<th>4-bit</th>
<th>8-bit</th>
<th>12-bit</th>
<th>16-bit</th>
<th>24-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>behavioral</td>
<td>4 LUT/6 Slice</td>
<td>12 LUT/6 Slice</td>
<td>18 LUT/6 Slice</td>
<td>24 LUT/11 Slice</td>
<td>41 LUT/20 Slice</td>
</tr>
<tr>
<td>structural</td>
<td>4 LUT/1 Slice</td>
<td>8 LUT/2 Slice</td>
<td>16 LUT/3 Slice</td>
<td>16 LUT/4 Slice</td>
<td>24 LUT/6 Slice</td>
</tr>
<tr>
<td>behavioral Eqs (9) and (10)</td>
<td>4 LUT/2 Slice</td>
<td>12 LUT/6 Slice</td>
<td>18 LUT/6 Slice</td>
<td>24 LUT/11 Slice</td>
<td>36 LUT/15 Slice</td>
</tr>
<tr>
<td>structural</td>
<td>4 LUT/1 Slice</td>
<td>8 LUT/2 Slice</td>
<td>16 LUT/3 Slice</td>
<td>16 LUT/4 Slice</td>
<td>24 LUT/6 Slice</td>
</tr>
</tbody>
</table>

Tab. 3. The area occupied by asynchronous Łukasiewicz t-norm in Spartan-6

<table>
<thead>
<tr>
<th>T-norm with architecture</th>
<th>4-bit</th>
<th>8-bit</th>
<th>12-bit</th>
<th>16-bit</th>
<th>24-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>behavioral</td>
<td>7 LUT/2 Slice</td>
<td>15 LUT/6 Slice</td>
<td>38 LUT/18 Slice</td>
<td>44 LUT/24 Slice</td>
<td>65 LUT/24 Slice</td>
</tr>
<tr>
<td>behavioral Eqs (9) and (10)</td>
<td>4 LUT/1 Slice</td>
<td>12 LUT/8 Slice</td>
<td>18 LUT/8 Slice</td>
<td>24 LUT/11 Slice</td>
<td>36 LUT/15 Slice</td>
</tr>
<tr>
<td>structural</td>
<td>4 LUT/1 Slice</td>
<td>8 LUT/2 Slice</td>
<td>12 LUT/3 Slice</td>
<td>16 LUT/4 Slice</td>
<td>24 LUT/6 Slice</td>
</tr>
</tbody>
</table>

Post-PAR Static Timing Report indicated that presented structural descriptions can be implemented in a Spartan-6 with a frequency above 400 MHz.
5. Conclusions

FPGA devices enable the implementation of high-speed and complex fuzzy systems. Diagrams as well as behavioral and structural descriptions allowing to create synchronous and asynchronous Łukasiewicz t-norms and t-conorms have been presented. It has been discussed how to use effectively Spratan-6 FPGA programmable resources to construct these norms. The structural description has been shown to provide implementation of the presented method of FPGA device configuration and to lead to the best area utilization. In this case resources occupied by fuzzy Łukasiewicz t-norm and t-conorm are equal to the multi-bit adder. Tests results expose a big difference in area saving between structural and behavioral description, especially in higher bit resolution.

6. References


Łukasz SURDEJ, MSc, eng.
Łukasz Surdej received the MSc degree in electronics and telecommunications from the Military University of Technology, Warsaw, Poland, in 2011. Currently, he is a second year PhD student at the faculty of computer science at the Rzeszow University of Technology. The area of his interests includes FPGAs, microcontrollers and fuzzy logic hardware.

e-mail: lukasz.surdej@gmail.com

Lesław GNIEWEK, DSc, PhD, eng.
Lesław Gniewek received the PhD and DSc degrees in computer science from Wroclaw University of Technology, in 1999 and 2014, respectively. From 2014 he is Associate Professor at the Rzeszow University of Technology. His research interests are in area of fuzzy logic hardware, fuzzy Petri nets, and programmable logic controllers.

e-mail: lgniewek@prz-rzeszow.pl