Int. J. Fuzzy Syst. (2018) 20(4):1130–1144
https://doi.org/10.1007/s40815-017-0399-6

Fuzzy Hold Flip-Flop and Flip-Flops Hardware Realizations

Łukasz Surdej[1] · **Lesław Gniewek**[1]

Received: 13 April 2017 / Revised: 24 September 2017 / Accepted: 26 September 2017 / Published online: 30 October 2017
© The Author(s) 2017. This article is an open access publication

Abstract This paper presents a new type Hold (H) of two-input fuzzy flip-flops. The definition of fuzzy H flip-flop for different fuzzy operations is given, the characteristics presented and selected properties pointed out. The new flip-flop is compared with other two-input fuzzy flip-flops described in the literature. Moreover, the potential for hardware implementation of fuzzy flip-flops is analyzed and diagrams, with the use of standard digital blocks, given. Special attention is paid to Lukasiewicz fuzzy flip-flops which allow efficient hardware realization. After the identification of the essential features of FPGAs, the presented diagrams were implemented in a Spartan-6 device. This allows the assessment of the suitability of FPGAs for fuzzy flip-flop implementations, as well as reception of fast and area optimized fuzzy flip-flops.

Keywords Fuzzy H (Hold) flip-flop · Flip-flop diagrams · FPGA implementation

1 Introduction

For many years, fuzzy flip-flops have been considered a replacement for binary flip-flops in fuzzy systems, acting as primary storage elements. The concept of a fuzzy flip-flop was presented by Hirota and Ozawa in [7]. Generalizing a binary JK flip-flop by replacing the logical conjunction, disjunction and negation, respectively, with t-norm, s-norm (t-conorm) and fuzzy negation they received the fuzzy flip-flop. Fuzzy operations do not hold distributive lows, the non-contradiction low and the excluded middle low, so that minterm and maxterm forms of fuzzy flip-flops characteristic equations, derived from binary flip-flops, are not the same (in contrast to binary flip-flops). This in turn initiated the search for characteristic equations of fuzzy flip-flops which ensure the identity of characteristics and enable convenient implementation in integrated circuits.

The equation of the first fuzzy JK flip-flop, based on Zadeh (min-max) operations, and its implementation were shown in [7]. An algebraic JK flip-flop was described in [11]. In [3], the authors made an attempt to specify a bounded JK flip-flop. In [5], a lack of analogy of this flip-flop to binary flip-flop for $K = 0$ and $J = 0$ was stated, and a family of four JK flip-flops $\mathrm{JK}_{\mathrm{SA}}$, $\mathrm{JK}_{\mathrm{AA}}$, $\mathrm{JK}_{\mathrm{AB}}$ and $\mathrm{JK}_{\mathrm{SB}}$ was proposed.

After the JK flip-flop, other flip-flops were also fuzzified. The definition of fuzzy SR flip-flop can be found, among others, in [1, 6, 15]. Due to the forbidden state when $S = R = 1$, fuzzy SR flip-flop was developed in two types, Set and Reset [15].

This article briefly describes the characteristic equations of binary and fuzzy flip-flops, covered so far in the literature. Against this background, we introduce a new fuzzy H (Hold) flip-flop. The characteristic table and the definition with characteristic equations will be given, selected properties discussed and characteristics illustrated. Diagrams of Zadeh and bounded fuzzy flip-flops using standard digital blocks will also be depicted. Next, the implementation of the diagrams in the Spartan-6 FPGA XC6SLX16 device will be presented. The implementation was preceded by the indication of properties of FPGA structure, which will have

✉ Łukasz Surdej
lukasz.surdej@gmail.com

Lesław Gniewek
lgniewek@prz-rzeszow.pl

[1] Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, W. Pola 2, 35-959 Rzeszow, Poland

a significant impact on the final results. This, in conjunction with an analysis of characteristic equations of Lukasiewicz flip-flops, allowed us to get fuzzy flip-flops with very good timing performance and also occupied a very small area.

2 Binary Flip-Flops

In most cases, binary flip-flops are the basis for development of fuzzy flip-flops. In digital systems, commonly used two-input binary flip-flops are JK and SR flip-flops. The characteristic table of these flip-flops is presented in Table 1.

Q is the current state of the flip-flop and Q^+ is the next state. In order to compare the characteristic table of different flip-flops, the inputs in Table 1 are marked twice.

The JK flip-flop has inputs called J and K. A high state at J while K is held low sets the output high, and a high state at K while J is held low resets the output. When $J = K = 0$, the flip-flop remembers a previous state, and when $J = K = 1$ it changes its state to opposite. JK flip-flop characteristic equations, describing its next output state Q^+, in the form of sum of products give Eq. (1) and in the form of product of sums Eq. (2)

$$Q^+_{\mathrm{JK}} = J\overline{Q} + \overline{K}Q, \tag{1}$$

$$Q^+_{\mathrm{JK}} = (J + Q)\left(\overline{K} + \overline{Q}\right). \tag{2}$$

SR flip-flop inputs are marked with S and R. They are, respectively, equivalent to inputs J and K of JK flip-flop. When $S = R = 1$, the SR flip-flop operation differs from the JK. This input configuration is forbidden for SR flip-flops. The characteristic equations of SR flip-flop define Eqs. (3) and (4)

$$Q^+_{\mathrm{SR}} = S + \overline{R}Q, \tag{3}$$

$$Q^+_{\mathrm{SR}} = \overline{R}(S + Q). \tag{4}$$

3 Fuzzy Flip-Flops

A common practice for developing fuzzy flip-flops is to generalize binary flip-flop characteristic equations by fuzzy norms. Popular operations used as fuzzy norms are Zadeh—min-max ($\wedge$, $\vee$), Lukasiewicz—bounded ($\otimes$, $\oplus$), algebraic and drastic operations with a complement

Table 1 Characteristic table of JK and SR flip-flops

S/J	R/K	Q^+_{JK}	Q^+_{SR}
0	0	Q	Q
0	1	0	0
1	0	1	1
1	1	$\overline{Q}$	Not allowed

($\overline{A} = 1 - A$) as a fuzzy negation [7, 16]. More complex operations, e.g., Yager, Hamacher, Dombi, Dubois-Prade and Frank, cause a lot of problems in hardware implementation and are rarely used (especially in simulations). Despite the fact that some of fuzzy triples hold De Morgan laws, none of them holds either distributive lows, the non-contradiction low and the excluded middle low. This causes that the minterm and maxterm forms of generalized Eqs. (1–4) are not the same. Moreover, a generalized JK flip-flop does not meet the boundary value specified in Table 1 [7]. To fulfill boundary demands and integrate minterm and maxterm characteristic, the authors modified the characteristics equations of fuzzy flip-flops. The first fuzzy flip-flop was Zadeh (min-max) JK flip-flop with the characteristic equation (5)

$$Q^+_{\mathrm{JK}} = \left(J \vee \overline{K}\right) \wedge (J \vee Q) \wedge \left(\overline{K} \vee \overline{Q}\right) \tag{5}$$

defined in [7]. The same flip-flop with algebraic operations was described in [11] by the characteristic equation (6)

$$Q^+_{\mathrm{JK}} = J + Q - JQ - KQ. \tag{6}$$

The bounded fuzzy JK flip-flop with Eq. (7)

$$Q^+_{\mathrm{JK}} = \left(J \oplus \overline{K}\right) \otimes (J \oplus Q) \otimes \left(\overline{K} \oplus \overline{Q}\right) \tag{7}$$

was presented in [3]. While $K = 0$ or $J = 0$, the bounded value of this flip-flop does not retain the analogy with a binary JK flop-flop. In [5], a family of four flip-flops $\mathrm{JK_{SA}}$ (8), $\mathrm{JK_{AA}}$ (9), $\mathrm{JK_{AB}}$ (10) and $\mathrm{JK_{SB}}$ (11) which provides such an analogy was proposed.

$$Q^+_{\mathrm{JKSA}} = \left[\left(J \otimes \overline{Q}\right) \oplus Q\right] \otimes \left[\overline{K} \oplus \overline{Q}\right]. \tag{8}$$

$$Q^+_{\mathrm{JKAA}} = \left[(J \oplus Q) \otimes \overline{Q}\right] \oplus \left[\overline{K} \otimes Q\right]. \tag{9}$$

$$Q^+_{\mathrm{JKAB}} = \left\{\left[\left(Z_1 \oplus \overline{Z_2}\right) \otimes Z_2\right] \oplus \overline{Z_3}\right\} \otimes Z_3, \tag{10}$$

where

$$\begin{aligned} Z_1 &= (J \otimes K) \oplus \overline{K}, \\ Z_2 &= \left(\overline{K} \otimes Q\right) \oplus J, \\ Z_3 &= \left(J \otimes \overline{Q}\right) \oplus \overline{K}. \end{aligned} \tag{11}$$

$$Q^+_{\mathrm{JKSB}} = \left\{\left[\left(V_1 \oplus \overline{V_2}\right) \otimes V_2\right] \oplus \overline{V_3}\right\} \otimes V_3,$$

where

$$\begin{aligned} V_1 &= (J \otimes K) \oplus \overline{K}, \\ V_2 &= \left(J \otimes \overline{Q}\right) \oplus Q, \\ V_3 &= \left(\overline{K} \otimes Q\right) \oplus \overline{Q}. \end{aligned}$$

The Lukasiewicz JK flip-flop can also be described by Eq. (12)

$$Q^+_{\mathrm{JK}} = r1 \otimes (r2 \oplus r3), \tag{12}$$

where

$$\begin{aligned} r_1 &= \left[\left(\overline{K} \oplus \overline{Q}\right) \oplus \left(K \otimes \overline{Q}\right)\right] \\ &\quad \oplus \left\{\left(K \oplus \overline{Q}\right) \otimes \left[(Q \otimes Q) \otimes \overline{J}\right]\right\}, \\ r_2 &= \left(\overline{K} \oplus Q\right) \otimes \left[J \oplus \left(\overline{K} \otimes Q\right)\right], \\ r_3 &= \left\{\left[\overline{K} \oplus \left(\overline{Q} \otimes \overline{Q}\right)\right] \otimes K\right\} \\ &\quad \otimes \left\{\left[\overline{K} \otimes \left(\overline{Q} \otimes \overline{Q}\right)\right] \oplus (J \oplus Q)\right\}. \end{aligned}$$

This equation provides the most similar characteristic to Lukasiewicz Set-type and Reset-type JK flip-flops, which comes directly from generalized minterm and maxterm forms of a binary JK flip-flop.

JK$_{SA}$ (8) and JK$_{AA}$ (9) flip-flops do not achieve a symmetry of characteristics which have Zadeh (5) and algebraic (6) flip-flop. Equations (10–12) allow the achievement of such a symmetry; however, they are complicated and therefore inconvenient for hardware implementation.

The forbidden state, occurring in a binary SR flip-flop when $S = R = 1$, is not desirable for fuzzy flip-flops. It will eliminate a large range of input values, which simultaneously partially belong to a high state. Therefore, fuzzy SR flip-flops are available in two types: Set and Reset. Table 2 lists the characteristic tables of two-input fuzzy flip-flops.If $S = R = 1$, the next state of a Set-type fuzzy SR flip-flop is high and a Reset-type is low. In this inputs configuration, the JK flip-flop takes the next state of $\overline{Q}$.

Set-type and Reset-type fuzzy SR flip-flops with various fuzzy operations were discussed in [6, 15]. The characteristic equation of the Set-type fuzzy flip-flop was defined as

$$Q^+_{\mathrm{SRSet}} = S \textcircled{s} \left(\overline{R} \textcircled{t} Q\right), \tag{13}$$

and the Reset-type as

$$Q^+_{\mathrm{SRReset}} = \overline{R} \textcircled{t} (S \textcircled{s} Q). \tag{14}$$

4 The Fuzzy H Flip-Flop

The general definition of a fuzzy H flip-flop is given in Definition 1.

Definition 1 The fuzzy H (Hold) flip-flop has two fuzzy inputs S and R and a fuzzy output Q, reflecting its actual inner state. A high state at the S input sets the output Q high and a high at R resets the output. If inputs are $S = R = 0$ or $S = R = 1$, this flip-flop holds the output at the previous state Q.

In Table 2, the characteristic table of a fuzzy H flip-flop is marked in bold. Definition 2 specifies the characteristic equations of Zadeh and Lukasiewicz H flip-flop.

Definition 2 Zadeh and Lukasiewicz H flip-flops define characteristic equations (15) and (16)

Table 2 Characteristic table of two-input fuzzy flip-flops

S/J	R/K	Q^+_{JK}	Q^+_{SRSet}	Q^+_{SRReset}	$\mathbf{Q_H}^+$
0	0	Q	Q	Q	**Q**
0	1	0	0	0	**0**
1	0	1	1	1	**1**
1	1	$\overline{Q}$	1	0	**Q**

$$Q^+_{\mathrm{Hs}} = \left(\overline{R} \textcircled{s} Q\right) \textcircled{t} \left[S \textcircled{s} \left(\overline{R} \textcircled{t} Q\right)\right], \tag{15}$$

$$Q^+_{\mathrm{Hr}} = (S \textcircled{t} Q) \textcircled{s} \left[\overline{R} \textcircled{t} (S \textcircled{s} Q)\right]. \tag{16}$$

Equations (15) and (16) are derived, respectively, from Set-type and Reset-type fuzzy SR flip-flops, but in contrast to them, they satisfied Theorem 1.

Theorem 1 *Equations (15) and (16) for Zadeh and Lukasiewicz operations are identical.*

Proof (for Zadeh H flip-flop)

$$\begin{aligned} Q^+_{\mathrm{Hs}} &= \left(\overline{R} \vee Q\right) \wedge \left[S \vee \left(\overline{R} \wedge Q\right)\right] \\ &= \left[\left(\overline{R} \vee Q\right) \wedge S\right] \vee \left[\left(\overline{R} \vee Q\right) \wedge \left(\overline{R} \wedge Q\right)\right] \\ &= (S \wedge Q) \vee \left(S \wedge \overline{R}\right) \vee \left(\overline{R} \wedge Q\right) \\ &= (S \wedge Q) \vee \left[\overline{R} \wedge (S \vee Q)\right] = Q^+_{\mathrm{Hr}} \end{aligned} \tag{17}$$

Proof (for Lukasiewicz H flip-flop) Consider equations from (18) to (25).

$$\overline{R} \oplus Q = \begin{cases} \overline{R} + Q & \text{if } \ \overline{R} + Q \le \mathbf{1} \\ \mathbf{1} & \text{if } \ \overline{R} + Q > \mathbf{1} \end{cases} \tag{18}$$

$$\overline{R} \otimes Q = \begin{cases} \overline{R} + Q - \mathbf{1} & \text{if } \ \overline{R} + Q > \mathbf{1} \\ \mathbf{0} & \text{if } \ \overline{R} + Q \le \mathbf{1} \end{cases} \tag{19}$$

$$\begin{aligned} & S \oplus \left(\overline{R} \otimes Q\right) \\ &= \begin{cases} S & \text{if } \ \overline{R} + Q \le \mathbf{1} \\ S + \overline{R} + Q - \mathbf{1} & \text{if } \ \overline{R} + Q > \mathbf{1} \,\&\, S + \overline{R} + Q - \mathbf{1} \le \mathbf{1} \\ \mathbf{1} & \text{if } \ \overline{R} + Q > \mathbf{1} \,\&\, S + \overline{R} + Q - \mathbf{1} > \mathbf{1} \end{cases} \end{aligned} \tag{20}$$

$$\begin{aligned} Q^+_{\mathrm{Hs}} &= \left(\overline{R} \oplus Q\right) \otimes \left[S \oplus \left(\overline{R} \otimes Q\right)\right] \\ &= \begin{cases} \mathbf{1} & \text{if } \ \overline{R} + Q > \mathbf{1} \,\&\, S + \overline{R} + Q - \mathbf{1} > \mathbf{1} \\ S + \overline{R} + Q - \mathbf{1} & \text{if } \ \overline{R} + Q > \mathbf{1} \,\&\, S + \overline{R} + Q - \mathbf{1} \le \mathbf{1} \\ S + \overline{R} + Q - \mathbf{1} & \text{if } \ \overline{R} + Q \le \mathbf{1} \,\&\, S + \overline{R} + Q - \mathbf{1} > \mathbf{0} \\ \mathbf{0} & \text{if } \ \overline{R} + Q \le \mathbf{1} \,\&\, S + \overline{R} + Q - \mathbf{1} \le \mathbf{0} \end{cases} \\ &= \begin{cases} \mathbf{1} & \text{if } \ S + \overline{R} + Q - \mathbf{1} > \mathbf{1} \\ S + \overline{R} + Q - \mathbf{1} & \text{if } \ \mathbf{1} \geqslant S + \overline{R} + Q - \mathbf{1} > \mathbf{0} \\ \mathbf{0} & \text{if } \ S + \overline{R} + Q - \mathbf{1} \le \mathbf{0} \end{cases} \end{aligned} \tag{21}$$

$$S \oplus Q = \begin{cases} S+Q & \text{if} \quad S+Q \leq \mathbf{1} \\ \mathbf{1} & \text{if} \quad S+Q > \mathbf{1} \end{cases} \tag{22}$$

$$S \otimes Q = \begin{cases} S+Q-\mathbf{1} & \text{if} \quad S+Q > \mathbf{1} \\ \mathbf{0} & \text{if} \quad S+Q \leq \mathbf{1} \end{cases} \tag{23}$$

$$\overline{R} \otimes (S \oplus Q) = \begin{cases} \overline{R} & \text{if} \quad S+Q > \mathbf{1} \\ S+\overline{R}+Q-\mathbf{1} & \text{if} \quad S+Q \leq \mathbf{1} \,\&\, S+\overline{R}+Q-\mathbf{1} > \mathbf{0} \\ \mathbf{0} & \text{if} \quad S+Q \leq \mathbf{1} \,\&\, S+\overline{R}+Q-\mathbf{1} \leq \mathbf{0} \end{cases} \tag{24}$$

$$\begin{aligned} Q^+_{\mathrm{Hr}} &= (S \otimes Q) \oplus \left[\overline{R} \otimes (S \oplus Q)\right] \\ &= \begin{cases} \mathbf{1} & \text{if} \quad S+Q > \mathbf{1} \,\&\, S+\overline{R}+Q-\mathbf{1} > \mathbf{1} \\ S+\overline{R}+Q-\mathbf{1} & \text{if} \quad S+Q > \mathbf{1} \,\&\, S+\overline{R}+Q-\mathbf{1} \leq \mathbf{1} \\ S+\overline{R}+Q-\mathbf{1} & \text{if} \quad S+Q \leq \mathbf{1} \,\&\, S+\overline{R}+Q-\mathbf{1} > \mathbf{0} \\ \mathbf{0} & \text{if} \quad S+Q \leq \mathbf{1} \,\&\, S+\overline{R}+Q-\mathbf{1} \leq \mathbf{0} \end{cases} \\ &= \begin{cases} \mathbf{1} & \text{if} \quad S+\overline{R}+Q-\mathbf{1} > \mathbf{1} \\ S+\overline{R}+Q-\mathbf{1} & \text{if} \quad \mathbf{1} \geqslant S+\overline{R}+Q-\mathbf{1} > \mathbf{0} \\ \mathbf{0} & \text{if} \quad S+\overline{R}+Q-\mathbf{1} \leq \mathbf{0} \end{cases} \end{aligned} \tag{25}$$

Equations (21) and (25) result in (26)

$$Q^+_{\mathrm{Hs}} = Q^+_{\mathrm{Hr}}, \tag{26}$$

which ends the proof. □

The equation of algebraic H flip-flop determines Definition 3.

Definition 3 The algebraic fuzzy H flip-flop is described by the characteristic equation (27)

$$Q^+_{\mathrm{H}} = SRQ + S\overline{R}\,\overline{Q} + \overline{R}Q. \tag{27}$$

Characteristics of the next state Q^+_{H} of a Zadeh H flip-flop for different values of previous state Q are shown in Fig. 1. The characteristics confirm that for all value of Q Zadeh H flip-flop meets the requirements specified in Table 2 (red dots in Fig. 1). The inner state Q can be reset if the values of $\overline{R}$ and S fall below Q or set if S and $\overline{R}$ exceed Q. In other cases, Q is held unchanged. If $R = 0$, this flip-flop can be used to store a maximum value of S. Moreover, Zadeh H flip-flop characteristics retain excellent symmetry which provides the same flip-flop behavior during the Set and Reset processes. Similar symmetry also maintains characteristics of Lukasiewicz and algebraic H flip-flops presented, respectively, in Figs. 2 and 3.

The characteristic of the Lukasiewicz H flip-flop is a plain with a vertical bias equal to the state Q. Bounded flip-flops presented so far in the literature have not had uniform characteristics. In the case of a bounded JK flip-flop, it is hard to even clearly determine a proper characteristic. The Lukasiewicz H flip-flop holds its state only when the value of S is equal to R. When inputs are not the same, the inner state changes by the value of the difference of signals S and R. Considering computational properties, the Lukasiewicz H flip-flop, according to Eqs. (21) and (25), conducts the operation $Q^+ = Q + S - R$ in the entire fuzzy range $[\mathbf{0}, \mathbf{1}]$. SR flip-flops also perform this operation, but a scope of Q^+ variation is limited. According to Eqs. (20) and (24), the output range for Set-type fuzzy flip-flop is $[S, \mathbf{1}]$ and for Reset-type is $\left[\mathbf{0}, \overline{\mathbf{R}}\right]$. A smaller range of variation is a result of priority of one SR flip-flop input over the other. In H flip-flop, inputs have the same priority. This property can be used in many applications. An example may be a fuzzy Petri net [4]. If the H flip-flop is used for modeling a fuzzy place p, even with input and output transition activated and a large value of S and R, the place will properly establish the marker. The SR flip-flop, due to the limited scope of calculation $\left[\mathbf{0}, \overline{\mathbf{R}}\right]$, may lose the marker. On the other hand, if $R = 0$, a fuzzy H flip-flop may function as an accumulator of S. The algebraic H flip-flop characteristic (Fig. 3) is a second-order polynomial surface which is curved according to the actual value of Q. As shown in Fig. 4, the characteristic at a diagonal $R = \overline{S}$ is linear only for $Q = 0.5$. For other values of Q, the state of algebraic H flip-flop is changing nonlinearly.Since the characteristic is varying with the change of Q, Q itself is also changing in subsequent time steps. Figure 5 presents the output value Q of fuzzy H flip-flops in subsequent simulation steps with initial $Q_0 = 0.2$ and inputs $R = 0.1$, $S = 0.3$. The inner state Q of algebraic H flip-flop comes hyperbolically to the state in Eq. (28)

$$Q^+_{\mathrm{H}} \to \frac{S\overline{R}}{S\overline{R} + \overline{S}R}, \tag{28}$$

which is independent of initial or actual state and depends only on the inputs S and R. Previous Q is held when $S = R = 0$ or $S = R = 1$. If inputs $S = R$ and they are not equal to 0 or 1, the state Q finally comes to the value 0.5. In a simulated case, the state of Lukasiewicz H flip-flop is increasing by the difference $S - R$ and Zadeh H flip-flop holds the value of S. It is worth to notice that all of fuzzy H flip-flops are stable (in contrast to fuzzy JK flip-flop [2]).

Fuzzy flip-flops were considered to be a part of a neural network used to produce transfer function [10]. Unique properties of fuzzy H flip-flops can be observed when input $R = \overline{S}$. In this case, transfer functions of fuzzy H flip-flops are presented in Fig. 6. The algebraic flip-flop produces a sigmoidal function, Zadeh—linear and Lukasiewicz forms

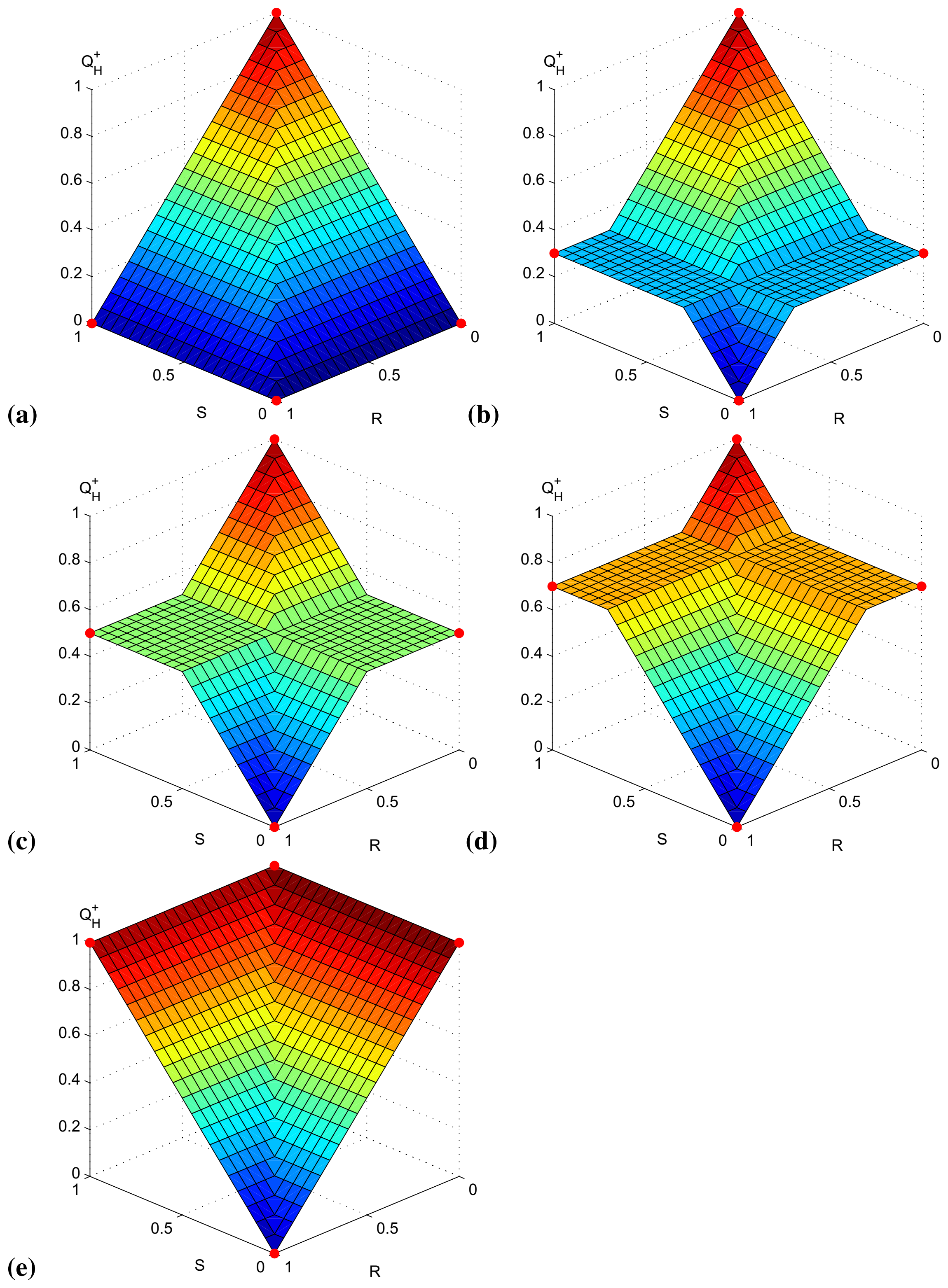


Fig. 1 Characteristics of a Zadeh H flip-flop for **a** $Q = 0.0$, **b** $Q = 0.3$, **c** $Q = 0.5$, **d** $Q = 0.7$, **e** $Q = 1.0$

a function like signum. These functions are the most popular transfer functions used in neural nets. Due to the limited speed of change of Q (Lukasiewicz H—proportional to $|S - R|$, algebraic H—hyperbolic, Fig. 5) the hysteresis is observed. The hysteresis size depends on the input dynamics.

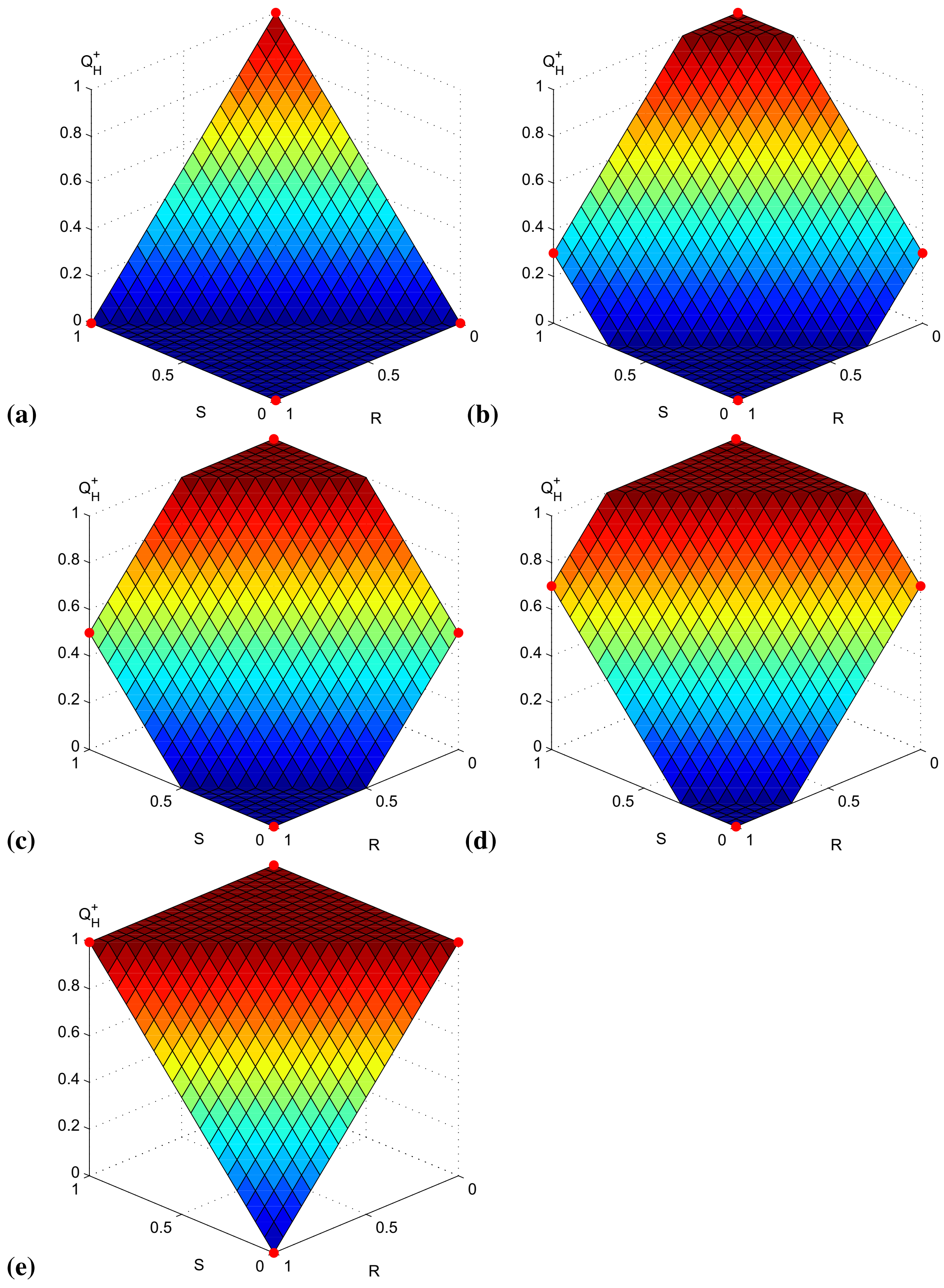


Fig. 2 Characteristics of a Lukasiewicz H flip-flop for **a** $Q = 0.0$, **b** $Q = 0.3$, **c** $Q = 0.5$, **d** $Q = 0.7$, **e** $Q = 1.0$

A fuzzy H flip-flops allow the building of fuzzy register. A Zadeh and a Lukasiewicz H flip-flop also permit to easily build fuzzy shift register (Fig. 7). In a Zadeh shift register, H flip-flops catch actual input state when $R = \overline{S}$ (Fig. 6). Lukasiewicz H flip-flops transfer input S to the output when $R = Q$ as a result of calculation

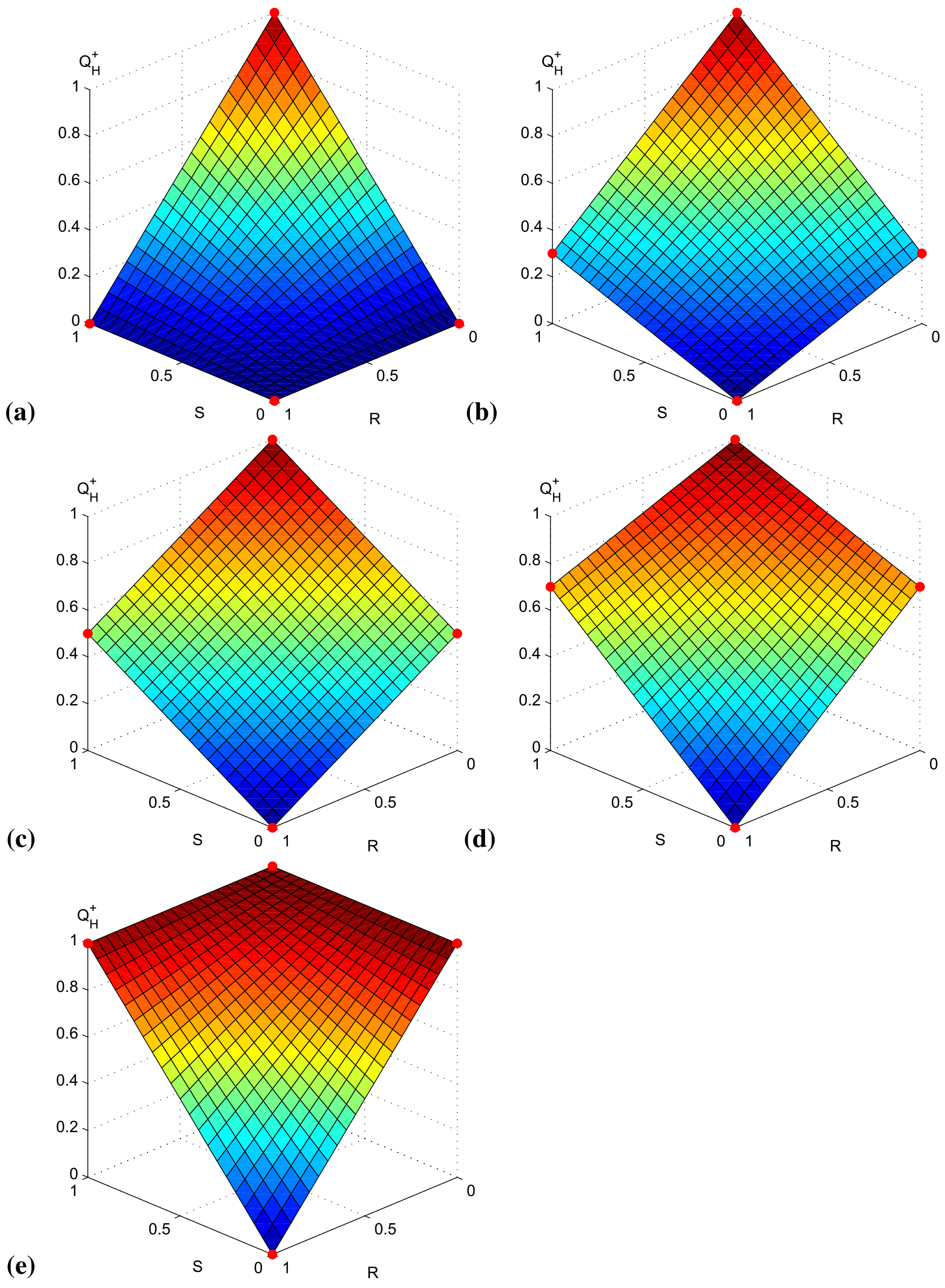


Fig. 3 Characteristics of an algebraic H flip-flop for **a** $Q = 0.0$, **b** $Q = 0.3$, **c** $Q = 0.5$, **d** $Q = 0.7$, **e** $Q = 1.0$

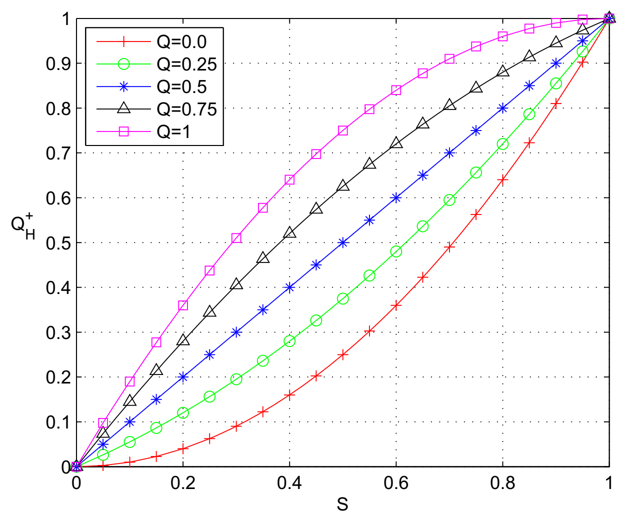


Fig. 4 Characteristics of an algebraic H flip-flop at a diagonal $R = \overline{S}$

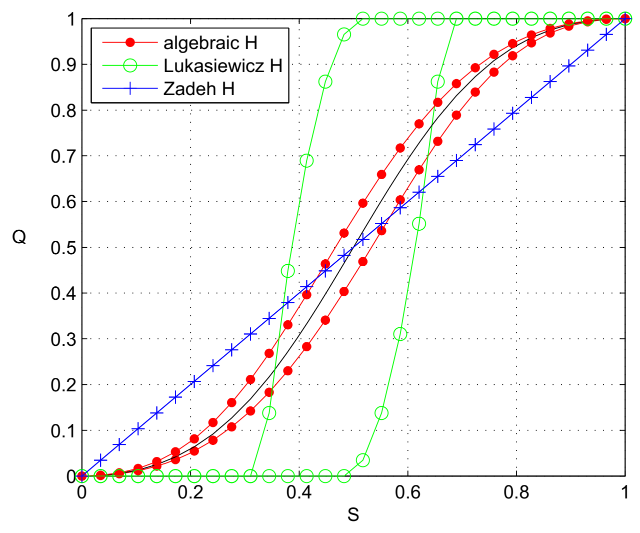


Fig. 6 Transfer functions of fuzzy H flip-flops for $R = \overline{S}$

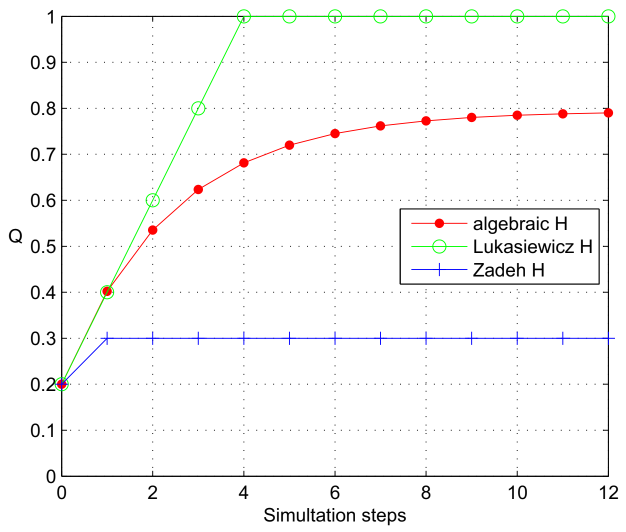


Fig. 5 Subsequent states of fuzzy H flip-flops ($Q_0 = 0.2$, $R = 0.1, S = 0.3$)

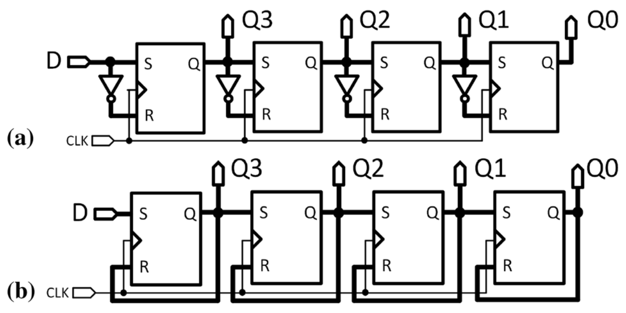


Fig. 7 Fuzzy shift right register with **a** Zadeh H flip-flops, **b** Lukasiewicz H flip-flops

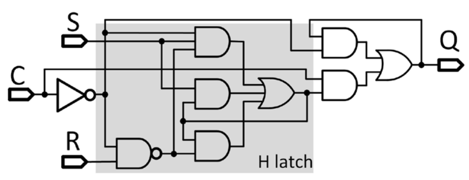


Fig. 8 Diagram of a binary H flip-flop

$$Q_H^+ = S + \overline{Q} + Q - 1 = S. \tag{29}$$

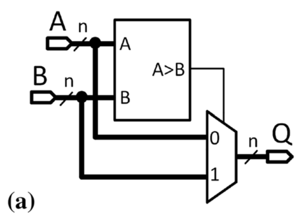
A fuzzy H flip-flop, like other fuzzy flip-flops, may have an equivalent binary flip-flop with Eq. (30)

$$Q_{Hb}^+ = SQ + S\overline{R} + \overline{R}Q. \tag{30}$$

An example of binary H flip-flop diagram is shown in Fig. 8. The gray area is an H latch. The rising edge of clock C blocks the latch and transfers its state to the output Q. The falling edge of C unblocks the latch and actives the output loop. Fuzzy H flip-flops, together with a binary H, allow the construction of a mixed fuzzy-binary system.

5 Fuzzy Flip-Flops Diagrams

A fuzzy flip-flop should store one fuzzy variable. Fuzzy variables can take continuous or discrete values from the range of [0, 1]. In hardware implementations of fuzzy flip-flops, continuous variables are usually substituted for voltage between [$0V$, $5V$]. In contrast, discrete variables are stored on n bits and take one of 2^n values from the discrete set $[0\ldots00, 0\ldots01, \ldots, 2^n - 1]$. In this case, the logic low state corresponds to the series of zeros $\mathbf{0} = [0\ldots00]_2$, and the logic high state to the series of ones $\mathbf{1} = 2^n - 1 = [1\ldots11]_2$.



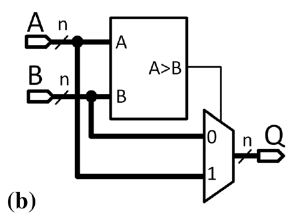


Fig. 9 Diagram of Zadeh **a** t-norm, **b** s-norm

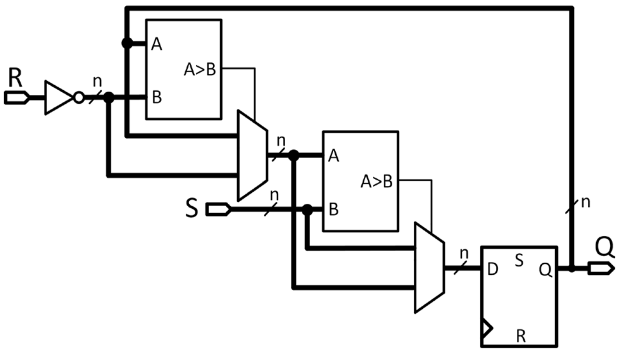


Fig. 10 Diagram of a Zadeh Set-type fuzzy SR flip-flop

The following part of the paper discusses the realization of Zadeh and Lukasiewicz discrete flip-flops. Flip-flops are ended with synchronous elements.

5.1 Zadeh Fuzzy Flip-Flops

Zadeh operations can be made using a comparator and a multiplexer. In Fig. 9, diagrams of Zadeh (a) t-norm and (b) s-norm are presented . The diagrams differ in connections of multiplexer inputs. For the construction of these operations, the n-bit comparator and n of 2:1 multiplexers are used. Similar realizations of Zadeh operations, using comparators and multiplexers, are presented, e.g., in [8, 12].

With the use of discussed Zadeh operations and characteristic equations [for JK flip-flop Eq. (5), for Set-type SR Eq. (13), Reset-type SR Eq. (14) and for H flip-flop Eqs. (15) or (16)], diagrams of Zadeh fuzzy flip-flops can be obtained. In fuzzy flip-flops diagrams, the number of comparators and multiplexers will be proportional to the number of fuzzy operations. The output of a fuzzy flip-flop is ended with a synchronous element (a binary D flip-flop). In Fig. 10, an example of a Set-type fuzzy SR flip-flop realization is depicted. In the equation of a fuzzy H flip-flop, there are t-norm and s-norm for the same signals. As a result, the diagram of an H flip-flop may be reduced by one comparator. Such realization, based on Eq. (16), is presented in Fig. 11. The number of digital blocks required to build Zadeh fuzzy flip-flops is collected in Table 3. The

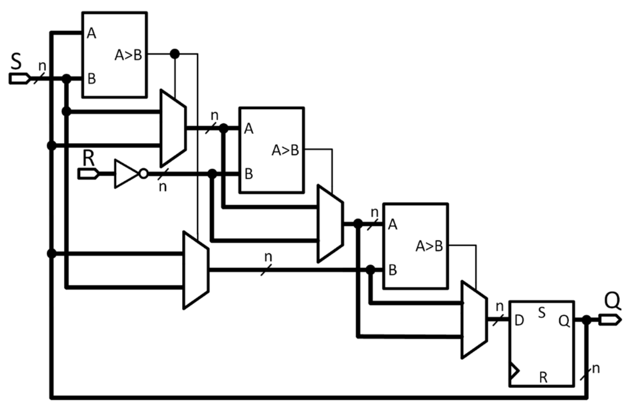


Fig. 11 Diagram of a Zadeh fuzzy H flip-flop

Table 3 Number of digital blocks required to construct Zadeh fuzzy flip-flops

Type	JK	SR Set	SR Reset	H
n-bit comp	5	2	2	3
mux 2:1	$5n$	$2n$	$2n$	$4n$
D flip-flop	n	n	n	n
inv	$2n$	n	n	n

least amount of elements requires SR flip-flops and the most JK flip-flops.

5.2 Lukasiewicz Fuzzy Flip-Flops

Lukasiewicz operation diagrams are usually based on a carry adder with an additional element in the output, imposing an appropriate limitation on its value. To restrict the output value, we can use logic gates [3], multiplexers [9, 16] or *Set*/*Reset* input of synchronous elements [14].

Diagrams of Lukasiewicz (a) t-norm and (b) s-norm with gates at the output are shown in Fig. 12. In the case of t-norm, the *CI* input of the adder is connected to a high state. If the *CO* output of the adder is low ($A + B + 1_2 \leq \mathbf{1}$), then the output of the circuit is also held low by AND gates. A high state in the *CO* output ($A + B + 1_2 > \mathbf{1}$) causes the circuit to perform operation

$$A + B - \mathbf{1} = A + B - 2_2^n + 1_2 \quad (31)$$

by omitting the most significant bit of the sum (carry bit *CO*).

In the case of s-norm, a sum $A + B$ is transmitted to the output until a high state of *CO* output of the adder ($A + B > \mathbf{1}$). A high state at the *CO*, through OR gates, holds the output in a high state.

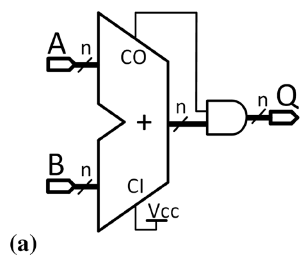


Fig. 12 Diagram of Lukasiewicz **a** t-norm, **b** s-norm with logic gates

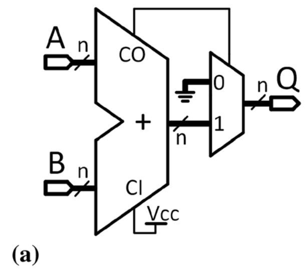


Fig. 13 Diagram of Lukasiewicz **a** t-norm, **b** s-norm with multiplexers

Lukasiewicz operations with multiplexers work in the same way, and their diagrams are presented in Fig. 13. Multiplexers, controlled by the CO output of the adder, switch t-norm state between $\mathbf{0}$ and $A+B-\mathbf{1}$, and s-norm state between $A+B$ and $\mathbf{1}$. Diagrams of Lukasiewicz operations with synchronous elements (D flip-flops) are illustrated in Fig. 14. Considering the s-norm, a high state of the carry CO, connected to the S (Set) flip-flop input, sets the output high. In the case of t-norm, the carry CO is connected through an inverter to the input R ($Reset$). As a result, a low state of CO resets the flip-flop.

The number of digital components required for construction of presented diagrams of fuzzy Lukasiewicz operations is summarized in Table 4.

Diagrams of Set-type and Reset-type fuzzy SR flip-flops can be obtained using discussed realizations of Lukasiewicz operations and the characteristic equations (13) and (14). Equation (13) is reflected by the diagram presented in Fig. 15. The operation $\overline{R} \otimes Q$ is composed of the adder with AND gates (Fig. 12a), while the Lukasiewicz sum, due to the requirement of synchronous fuzzy flip-flop output, consists of the adder and binary flip-flops (Fig. 14b). By analogy, the diagram of a Reset-type flip-flop [Eq. (14)] is shown in Fig. 16.

Equation (20), which is the extension of the characteristic equation of a Set-type fuzzy SR flip-flop (Eq. (13)), can also be written as

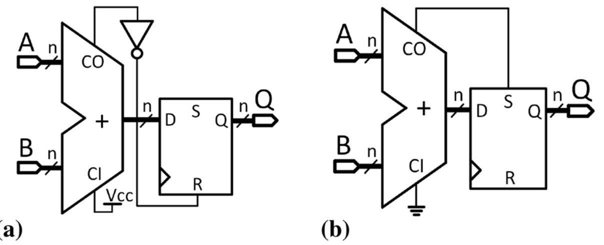


Fig. 14 Diagram of Lukasiewicz **a** t-norm, **b** s-norm with D flip-flops

Table 4 Number of digital blocks required for construction Lukasiewicz fuzzy flip-flops

Type	Figure 12	Figure 13	Figure 14
n-bit adder	1	1	1
AND/OR gate	n	–	–
mux 2:1	–	n	–
D flip-flop	–	–	n
inv	–	–	1*

* only for t-norm

$$Q^{+}_{\mathrm{SRS}} = \begin{cases} S & \text{if} \quad Y_r \le \mathbf{1} \\ (Y_r - \mathbf{1}) + S & \text{if} \quad Y_r > \mathbf{1} \,\&\, (Y_r - \mathbf{1}) + S \le \mathbf{1} \\ \mathbf{1} & \text{if} \quad Y_r > \mathbf{1} \,\&\, (Y_r - \mathbf{1}) + S > \mathbf{1} \end{cases}$$

where $Y_r = \overline{R} + Q$.

(32)

Equation (32) corresponds to the diagram presented in Fig. 17. Multiplexers are driven by a signal CO_1 of the first adder. When $CO_1 = 0$, multiplexers provide the input of D flip-flops with a signal S. If $CO_1 = 1$, the first adder produces a signal $(Y_r - \mathbf{1})$, and the sum $(Y_r - \mathbf{1}) + S$ from the second adder feeds D flip-flops. If carry signals CO_1 and CO_2 are held high then the AND gate, connected to S input of flip-flops, sets the state $\mathbf{1}$ at its output.

Similarly, from Eq. (24), describing a Reset-type fuzzy SR flip-flop, written in the form of

$$Q^{+}_{\mathrm{SRR}} = \begin{cases} \overline{R} & \text{if} \quad Y_S > \mathbf{1} \\ Y_s + \overline{R} - \mathbf{1} & \text{if} \quad Y_s \le \mathbf{1} \,\&\, Y_s + \overline{R} > \mathbf{1} \\ \mathbf{0} & \text{if} \quad Y_s \le \mathbf{1} \,\&\, Y_s + \overline{R} \le \mathbf{1} \end{cases} \qquad (33)$$

where $Y_s = S + Q$,

we can obtain the diagram presented in Fig. 18. In this case, D flip-flops are reset at low state of CO_1 and CO_2. If $CO_1 = 0$ and $CO_2 = 1$, the first adder produces a sum Y_s and the second adder performs operation (31). While $CO_1 = 1$, the signal $\overline{R}$ feeds D flip-flops. Based on the characteristic equations (15) and (16) of the H fuzzy flip-

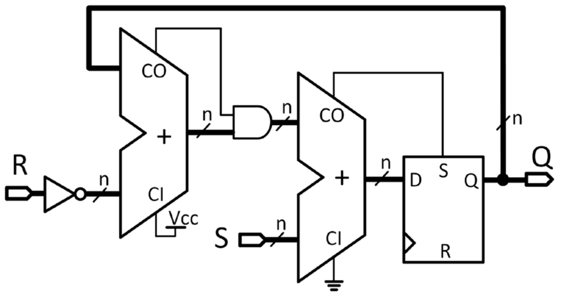


Fig. 15 Diagram of a Lukasiewicz Set-type SR flip-flop with AND gates

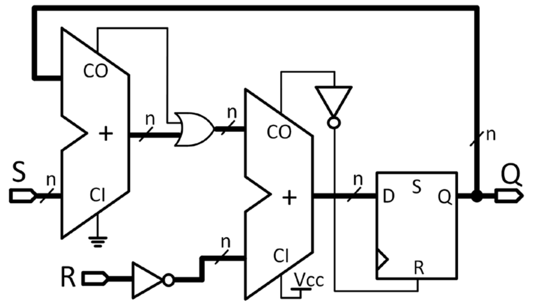


Fig. 16 Diagram of a Lukasiewicz Reset-type SR flip-flop with OR gates

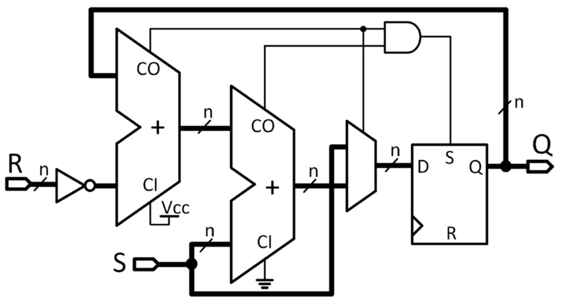


Fig. 17 Diagram of a Lukasiewicz Set-type SR flip-flop with multiplexers

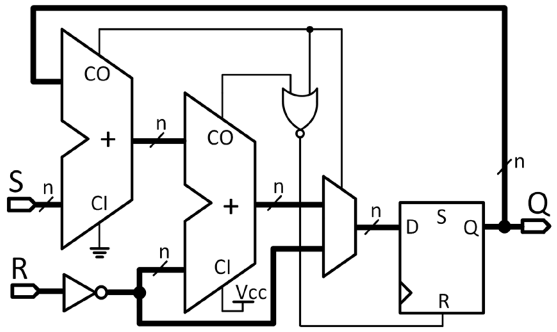


Fig. 18 Diagram of a Lukasiewicz Reset-type SR flip-flop with multiplexers

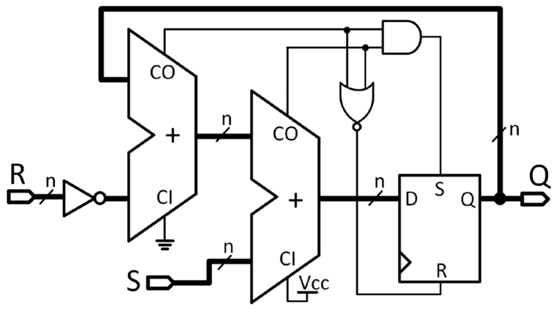


Fig. 19 Diagram of a Lukasiewicz H flip-flop from Eq. (34)

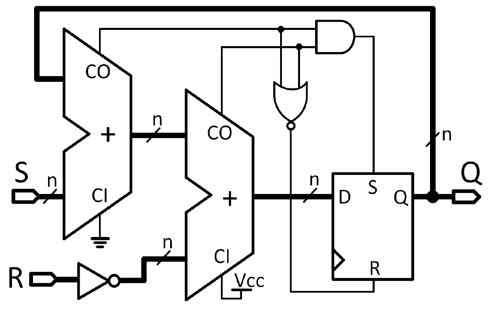


Fig. 20 Diagram of Lukasiewicz H flip-flop from Eq. (35)

flop, which contain four fuzzy operations, it could be deduced that its diagram will be twice the size of the SR flip-flop. However, analyzing the values taken by the flip-flop, its diagram can be simplified. Equation (21) can be written as

$$Q^{+}_{\mathrm{Hs}} = \begin{cases} 1 & \text{if} \quad Y_r > \mathbf{1} \,\&\, (Y_r - \mathbf{1}) + S > \mathbf{1} \\ (Y_r - \mathbf{1}) + S & \text{if} \quad Y_r > \mathbf{1} \,\&\, (Y_r - \mathbf{1}) + S \le \mathbf{1} \\ Y_r + S - \mathbf{1} & \text{if} \quad Y_r \le \mathbf{1} \,\&\, Y_r + S > \mathbf{1} \\ \mathbf{0} & \text{if} \quad Y_r \le \mathbf{1} \,\&\, Y_r + S \le \mathbf{1}. \end{cases} \tag{34}$$

Equation (34) allows to obtain the diagram of the H fuzzy flip-flop presented in Fig. 19.Depending on the condition $Y_r > \mathbf{1}$, operation (31) is performed in the first or in the second adder. The boundary of a state $\mathbf{1}$ imposes the AND gate at $CO_1 = CO_2 = 1$, and relating to the state $\mathbf{0}$ the NOR gate at $CO_1 = 0$ and $CO_2 = 0$. Similarly, Eq. (25) in the form

$$Q^{+}_{\mathrm{Hr}} = \begin{cases} 1 & \text{if} \quad Y_s > \mathbf{1} \,\&\, (Y_s - \mathbf{1}) + \overline{R} > \mathbf{1} \\ (Y_s - \mathbf{1}) + \overline{R} & \text{if} \quad Y_s > \mathbf{1} \,\&\, (Y_s - \mathbf{1}) + \overline{R} \le \mathbf{1} \\ Y_s + \overline{R} - \mathbf{1} & \text{if} \quad Y_s \le \mathbf{1} \,\&\, Y_s + \overline{R} > \mathbf{1} \\ \mathbf{0} & \text{if} \quad Y_s \le \mathbf{1} \,\&\, Y_s + \overline{R} \le \mathbf{1} \end{cases} \tag{35}$$

leads to the diagram of a Lukasiewicz H fuzzy flip-flop depicted in Fig. 20. As shown in Figs. 19 and 20, in the diagram of the H flip-flop, there is no difference which

Table 5 Number of digital blocks required to construct Lukasiewicz fuzzy flip-flops

Type	Figure 15	Figure 16	Figure 17	Figure 18	Figures 19/20
n-bit adder	2	2	2	2	2
mux 2:1	–	–	*n*	*n*	–
AND gate	*n*	–	1	–	1
OR gate	–	*n*	–	–	–
NOR gate	–	–	–	1	1
D flip-flop	*n*	*n*	*n*	*n*	*n*
inv	*n*	$n+1$	*n*	*n*	*n*

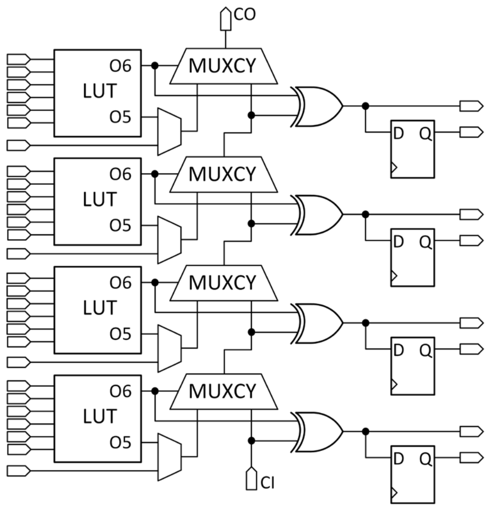


Fig. 21 Components of FPGA slice used for implementation of an adder

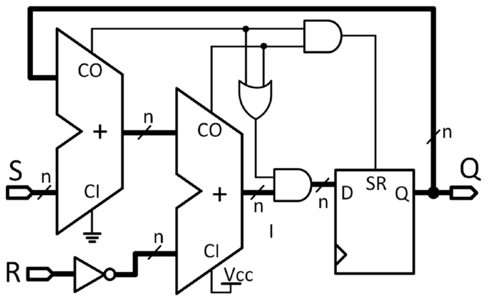


Fig. 22 Diagram of Lukasiewicz H flip-flop with Reset performed by OR and AND gates

adder will be fed by a signal S and which by $\overline{R}$. There is also no difference whether $CI_1 = 0$ and $CI_2 = 1$, or $CI_1 = 1$ and $CI_2 = 0$.

Table 5 summarizes the number of components used to build presented Lukasiewicz flip-flops.The least amount of digital elements requires an H flip-flop. Its construction, in addition to two n-bit adders, inverters and flip-flops, also used in all other diagrams, needs one two-input AND gate and one NOR gate.

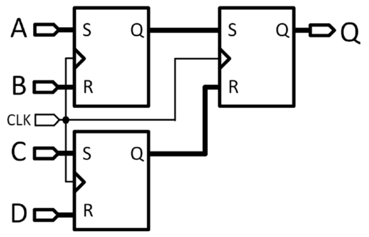


Fig. 23 Testing diagram implemented in Spartan-6 FPGA

6 Lukasiewicz Fuzzy Flip-Flops in FPGA

The main components of Lukasiewicz flip-flop diagrams, presented in Sect. 5.2, are adders and binary flip-flops. Primary resources in FPGAs dedicated for implementation of high-speed adders are based on a carry logic [13]. Figure 21 represents the carry chain of L/M slice of the Spartan-6 FPGA with additional elements. Similar structures have slices in Xilinx 7 Series FPGAs.The individual bits of the sum s_i are calculated by XORCY gates according to the equation

$$s_i = c_{i-1}\,\mathrm{XOR}\,p_i \qquad (36)$$

where $p_i = a_i\,\mathrm{XOR}\,b_i$ is produced in a LUT (lookup table). The a_i and b_i are, respectively, the ith bits of input fuzzy numbers A and B. The carry bits c_i

$$c_i = a_i b_i + c_{i-1} p_i \qquad (37)$$

are generated by multiplexers MUXCY. Multiplexers MUXCY can calculate conjunction $a_i b_i$ as

$$a_i b_i = a_i\overline{p_i} = b_i\overline{p_i} = a_i b_i\overline{p_i} \qquad (38)$$

and it is required to supply a signal a_i or b_i or their logical product.

During implementation, attention has to be paid to several important properties of the Spartan-6 slice structure.

Property 1 *D flip-flops are directly after XOR gates. Therefore, if in Lukasiewicz flip-flop diagrams D flip-flops occur immediately after the adder (Figs. 15, 16, 19, 20), it*

Table 6 Summary of three fuzzy flip-flop implementation in Spartan-6 FPGA (Fig. 23)

F3 type/arch.	Area [slice]			
	4-bit	8-bit	12-bit	16-bit
SR Set Zad./beh.	13	58	34	44
SR Reset Zad./beh.	18	41	35	41
Hs Zad./beh.	21	72	44	59
Hr Zad./beh.	34	81	54	81
JK Zad./beh.	57	87	70	98
SR Set Luk./beh.	20	23	34	42
SR Reset Luk./beh.	21	27	36	44
Hs Luk./beh.	22	30	30	40
Hr Luk./beh.	23	25	32	41
SR Set Luk./str.	6	12	18	24
SR Reset Luk./str.	9	15	21	27
Hs Luk./str.	9	15	21	27
Hr Luk./str.	9	15	21	27
F3 type/arch.	Frequency$_{max}$ [MHz]			
	4-bit	8-bit	12-bit	16-bit
SR Set Zad./beh.	263.644	211.820	166.417	164.609
SR Reset Zad./beh.	254.518	209.987	164.718	162.075
Hs Zad./beh.	238.436	114.051	109.087	111.508
Hr Zad./beh.	211.640	129.450	117.317	109.914
JK Zad./beh.	186.498	107.945	106.101	101.153
SR Set Luk./beh.	225.023	211.730	201.086	199.362
SR Reset Luk./beh.	215.285	208.638	201.248	176.616
Hs Luk./beh.	211.640	211.999	197.044	181.422
Hr Luk./beh.	215.146	211.999	190.042	185.667
SR Set Luk./str.	254.065	243.191	212.404	212.134
SR Reset Luk./str.	225.023	213.721	199.601	189.934
Hs Luk./str.	229.516	217.014	211.416	208.464
Hr Luk./str.	214.316	212.630	201.167	194.894

is possible to implement them in the same slice. However, if there are other logic elements between the adder and flip-flops, such as multiplexers in Figs. 17 and 18, it is required to engage additional logic resources. This increases the occupied area, as well as decreases timing performance.

Property 2 *D flip-flops in the 6 and 7 Series FPGAs have one shared SR input which is active high. For this reason, there is no possibility to connect signals Set and Reset to D flip-flop at the same time . Therefore, one of these functions must be done before applying the signal to the flip-flop. Figure 22 presents a diagram of the H flip-flop from Fig. 20 with the Reset function performed by OR and AND gates.* The OR gate can be replaced by the XOR gate. As in Property 1, this implementation requires additional logic resources.

Property 3 *Additional resources are also required by the logic gates, which bound the output value based on carry signal from adders. These gates often build a critical time path; thus, their performance determines the maximum operating frequency of the whole fuzzy flip-flop. The best way to realize the inverter (Fig. 16), the XOR gate (Fig. 22 instead of OR gate) and the AND gate (Figs. 17, 19, 20) is to use the components of carry logic which are associated with the second adder carry chain. Timing simulations have indicated that the inverter implemented in a XORCY gate in the form of* $\overline{CI_2} = (CI_2 \text{ XOR } 1)$ *significantly affects the maximum operating frequency (in the case of the 4-bit Lukasiewicz flip-flop, the maximum frequency has increased from 164.962 MHz to 225.023 MHz). The AND gate can be the product of a multiplexer MUXCY configured according to equation* $CI_1CI_2 = CI_1CI_2 + \overline{CI_1}0$. *On the other hand, both OR and NOR gates can not be created in such an effective way.*

Property 4 *LUTs, calculating propagation group* p_i, *allow to implement one arbitrary six-input logic function or two five-input logic functions which share the same inputs. The O5 output can also supply an input of MUXCY multiplexer. As a result, logic elements on inputs of Lukasiewicz flip-flop adders can be done in LUT tables calculating propagation group. Thence, logic elements on adder inputs will not have a significant impact on the occupied area and timing performance. In the case of the H flip-flop from Fig. 22, two-output LUTs also permit to place AND gates (of the Reset circuit) and D flip-flops in the area of the second adder. This can be done by looping the signal from the second adder to its subsequent input of LUTs and producing on their O5 outputs AND gates. Such action significantly reduces the occupied area by H flip-flop, but additional delays, caused by the loop, have to be reckoned with.*

Taking into account the discussed properties, the best implementation results, in terms of both occupied resources and the maximum clock frequency, are expected for the Set-type SR flip-flop (Fig. 15). In this fuzzy flip-flop, according to Property 1, D flip-flop comes right after an adder, only the Set input is used (Property 2) to which is applied a signal CO_2 without additional gates (Property 3), and all logic gates can be integrated with adders (Property 4).

7 The Results of Implementation in FPGA

Comparison of fuzzy flip-flop FPGA implementation results was performed as follows. Zadeh and Lukasiewicz fuzzy flip-flops were coded behaviorally in VHDL. In order to fulfill the requirements stated in Sect. 6, Lukasiewicz flip-flops were additionally described structurally. In structural descriptions, primitives of Xilinx FPGA device components (LUTs, XORCY gate, MUXCY multiplexer and D flip-flop) were used. Components were properly distributed with the use of RLOC attributes. Details of fuzzy operations structural description can be found in [14]. The Xilinx Spartan-6 FPGA XC6SLX16 was selected as the target device, mounted on an inexpensive evaluation kit SP601. In this device, three flip-flops connected as shown in Fig. 23 were implemented.This diagram allows the area occupied by three fuzzy flip-flops and their maximum clock frequency (associated with the delay produced by combinatorial logic placed between synchronous elements) to be determined. A timing constraint of 200 MHz was imposed on the clock frequency. During synthesis and implementation, the design goal was timing performance. The occupied area was taken from the Place and Route Report, and the maximum operating frequency came from the Post-PAR Static Timing Report. Table 6 summarizes the implementation results.

Considering the occupied area, the best results for all tested bit resolutions were obtained for structural descriptions of a Lukasiewicz Set-type SR flip-flop. Their area was equal to the resources of two adders. Three 4-bit flip-flops took a total area of 6 slices (2 slices per flip-flop). A slightly worse result was achieved for other bounded flip-flops described structurally. It is connected with the implementation of flip-flop boundaries logic. Behavioral description outcomes were far worse than structural ones. Among them, most of the flip-flops with the same resolution took a similar area. Significantly lower resources than the average were required for the behavioral description of a 4-bit Zadeh Set-type SR flip-flop. This flip-flop also reached the highest clock frequency of all tested flip-flops. Unfortunately, for other resolutions the outcome was much worse. The worst results gave a Zadeh JK flip-flop. It occupied the largest area and had the worst timing characteristics. Zadeh H flip-flops had also poor performance. It is worth to point out that the area of Zadeh flip-flops shrank from 8-bit to 12-bit implementation. The reason was that the synthesizer in 12-bit and 16-bit implementations used MUXCY multiplexers.

Comparing the maximum clock frequency, it is noticeable that the frequency decreased very fast in the case of Zadeh flip-flops. In Lukasiewicz flip-flops, both described behaviorally and structurally, and this decline was much slower. Although the 4-bit Zadeh Set-type SR flip-flop reached the highest clock frequency, bounded flip-flops performances outweighed Zadeh flip-flops at 8-bit resolution and higher. Structurally described flip-flops, with the properties listed in Sect. 6, got a high clock frequency compared to other flip-flops. In this case, like in the case of the occupied area, the best results were obtained by the Set-type SR flip-flop. Good results were also achieved by H flip-flops, despite the fact that the structure of FPGAs does not exactly fit to Figs. 19 and 20. The lowest clock frequency had JK flip-flops.

8 Conclusion

The paper presents a new type H (Hold) of fuzzy flip-flop. Its definition, characteristic equations and selected properties were covered. The characteristics of this flip-flop with a variety of fuzzy operations are similar but more homogeneous then corresponding JK and SR flip-flops. Moreover, fuzzy H flip-flops can easily be used in many applications starting from fuzzy register or shift register to more sophisticated like neural nets, fuzzy Petri nets or fuzzy-binary system. The article also presents schematics of Zadeh and Lukasiewicz fuzzy flip-flops. Despite more complex characteristic equations, the Lukasiewicz H flip-flop demands the smallest number of digital blocks. On the other hand, the implementation of fuzzy flip-flops in the FPGA showed that the best fitted to the structure of FPGAs is the Lukasiewicz Set-type SR flip-flop. Nevertheless, with a structural description Lukasiewicz flip-flops with a small area and high timing performance can be obtained.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Atmaca, H., Yavuz, S.H.: The implementation of fuzzy flip-flops as mempry modules. In: Proceedings of ELECO'99 International Conference Conference on Electrical and Electronics Engineering, pp. 423–427 (1999)
2. Choi, B., Shukla, K.: Multi-valued logic circuit design and implementation. Int. J. Electron. Electr. Eng. **3**(4), 256–262 (2015)
3. Diamond, J., Pedrycz, W., McLeod, D.: Fuzzy JK flip-flops as computational structures: design and implementation. IEEE Trans. Circuits Syst. II Analog Digit. Signal Process. **41**(3), 215–226 (1994)

4. Gniewek, L.: Sequential control algorithm in the form of fuzzy interpreted petri net. IEEE Trans. Syst. Man Cybern. Syst. **43**(2), 451–459 (2013)
5. Gniewek, L., Kluska, J.: Family of fuzzy JK flip-flops based on bounded product, bounded sum and complementation. IEEE Trans. Syst. Man Cybern. Part B Cybern. **28**(6), 861–868 (1998)
6. Hirota, K.: Mathematical properties of various fuzzy flip-flops as a basis of fuzzy memory modules. Mach. Intell. Quo Vadis **21**, 161–178 (2004)
7. Hirota, K., Ozawa, K.: The concept of fuzzy flip-flop. IEEE Trans. Syst. Man Cybern. **19**(5), 980–997 (1989)
8. Kluska, J., Hajduk, Z.: Digital implementation of fuzzy petri net based on asynchronous fuzzy RS flip-flop. In: International Conference on Artificial Intelligence and Soft Computing, pp. 314–319. Springer (2004)
9. Lovassy, R., Gál, L., Tóth, Á., Kóczy, L.T., Rudas, I.J.: Fuzzy flip-flop based neural networks as a novel implementation possibility of multilayer perceptrons. In: 2012 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), pp. 280–285. IEEE (2012)
10. Lovassy, R., Kóczy, L., Gál, L.: Analyzing fuzzy flip-flops based on various fuzzy operations. Acta Technica Jaurinensis **1**(3), 447–465 (2008)
11. Ozawa, K., Hirota, K., Kczy, L.T., Omori, K.: Algebraic fuzzy flip-flop circuits. Fuzzy Sets Syst. **39**(2), 215–226 (1991). doi:10.1016/0165-0114(91)90214-B
12. Rudas, I.J., Batyrshin, I.Z., Zavala, A.H., Nieto, O.C., Horváth, L., Vargas, L.V.: Generators of fuzzy operations for hardware implementation of fuzzy systems. In: Mexican International Conference on Artificial Intelligence, pp. 710–719. Springer (2008)
13. Spartan-6 FPGA Configurable Logic Block User Guide UG384
14. Surdej, Ł., Gniewek, L.: Synchronous and asynchronous structural implementation of Lukasiewicz norms in spartan-6 FPGAs. Meas. Autom. Monit. **62**(11), 361–366 (2016)
15. Yoshida, S.I., Takama, Y., Hirota, K.: Fuzzy flip-flops and their applications to fuzzy memory element and circuit design using FPGA. J. Adv. Comput. Intell. Intell. Inf. **4**(5), 380–386 (2000)
16. Zavala, A.H., Nieto, O.C., Batyrshin, I., Vargas, L.V.:VLSI implementation of a module for realization of basic t-norms on fuzzy hardware. In: IEEE International Conference on Fuzzy Systems, 2009. FUZZ-IEEE 2009. pp. 655–659. IEEE (2009)

Łukasz Surdej is a Ph.D. 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.

Lesław Gniewek received the Ph.D. and D.Sc. degrees in computer science from Wroclaw University of Technology. 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.