Towards Hazard-Free Multiplexer Based Implementation of Self-Timed Circuits

Alexander Kushnerov[1*], Moti Medina[2], Alexandre Yakovlev[3]
[1]Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel
[2]Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel
[3]School of Engineering, Newcastle University, UK
[*]E-mail: kushnero@ee.bgu.ac.il

***Abstract*—The cost of design, test and fabrication of self-timed circuits remains prohibitive for their wider adoption in practice. Addressing this issue, researchers are trying to find ways for rapid prototyping of self-timed circuits in FPGAs. Combinational logic is realized in FPGAs by look-up tables (LUTs), which are typically built as a binary tree of 2-way multiplexers (MUX 2:1). This brings us to the idea of using MUX 2:1 in self-timed designs particularly, in quasi-delay-insensitive (QDI) circuits. Multiplexers however, realize a binate (non-monotone) Boolean function and therefore may cause logic hazards. A standard way for preventing these hazards requires designing of special circuit for MUX 2:1. On the other hand, there are indirect evidences that the multiplexers in some commercial FPGAs are hazard-free. Based on this assumption, we propose an original approach for realizing a multi-input C-element, which is widely used in QDI circuits. This paves the way for using hazard-free MUX 2:1 in more complex self-timed elements. All the proposed circuits are designed and verified in a CAD tool Workcraft.**

Keywords—binate function; C-element; consensus cube; hazard; lookup table; multiplexer; QDI circuit

I. Introduction

Self-timed or asynchronous circuits do not use clock to ensure the validity of signals and operate in the mode of request-acknowledge. As per definitions in [1] this is called a compliant operation mode between a circuit and its environment. The compliant operation is often considered at the level of individual logic gates. In [1], compliance is captured in the property of semi-modularity. Let us briefly recall it here. A self-timed circuit is an interconnection of logic gates. Each logic gate is defined by its Boolean function. An output of a logic gate can be in a stable or in an excited state. In a stable state its output is in logical 0 or logical 1, and this value corresponds to the value of the Boolean function of the gate. In an excited state the gate's output value is opposite to that of its Boolean function. The gate can thus either switch to the new stable state or return to the previous stable state. The effect of returning to the previous stable state is often called a hazard[1], but in the theory of asynchronous circuits [1] it has been defined more rigorously by the concept of "violation of semimodularity". There are two classes of self-timed circuits that are considered to be hazard-free. One is speed-independent (SI)[2] circuits [1], which assumes that gates have finite, but unbounded delays, and wires have zero delays. This implies in particular, that the difference of delays in any branching of wires is also zero. The other class is quasi-delay-insensitive (QDI) circuits [2] that also assumes gates to have finite yet unbounded delays. With respect to wires QDI assumes that the difference between the delays of the branches is less than the minimum gate delay. This assumption is called the assumption of an *isochronic* fork.

While in theory there is a subtle distinction between these two classes, SI (or more precisely semimodular SI) and QDI, mostly due to their different theoretical origins, in practice one can always modify the description of the SI circuit and consider it as a QDI circuit [3]. Therefore, avoiding some tedious explanations, we will in the following use the more widely used term QDI, which can be applied to semimodular SI circuits. Notably, in order to meet the requirements of QDI circuits, one requires to assume that each gate is atomic in the sense that its internal structure only has a delay element associated with the gate's output, which we will call here the atomicity assumption. Within the class of QDI, in this work, we also use the term output-persistence [3], which allows us to extend the class of semi-modular circuits with the circuits that have inputs from the environment. So, the semimodularity condition can be applied only to the outputs of the circuit's gates.

A C-element (strictly speaking, Muller C-element [1]) is a logic circuit realizing a latch function (see Section II), and widely used in asynchronous systems for:

1) the implementation of information processing (data path) with indication of inputs and outputs [4], [5],
2) the indication of the completion of transient processes (completion detection) in the data path [6], [7], and
3) the coordination of concurrent processes (control path) [3].

Most of the existing methods for designing SI and QDI circuits rely on the use of two-input or multi-input C-elements in either original or generalized form [3], [4]. It is often the case, however, that the library of logic gates may be restricted by the so-called simple gates, such as NAND2 and NOR2. Hence a problem of realizing a QDI circuit in a restricted basis arises. It is associated with the problem of QDI decomposition [3], which is pertinent for ASIC design, and even more so for FPGA-based design.

Nowadays, not only FPGAs, but entire FPGA development boards are very low cost. This makes them extremely attractive for prototyping self-timed circuits. The main obstacle for that is the fact that FPGAs are intended for synchronous, clocked design.

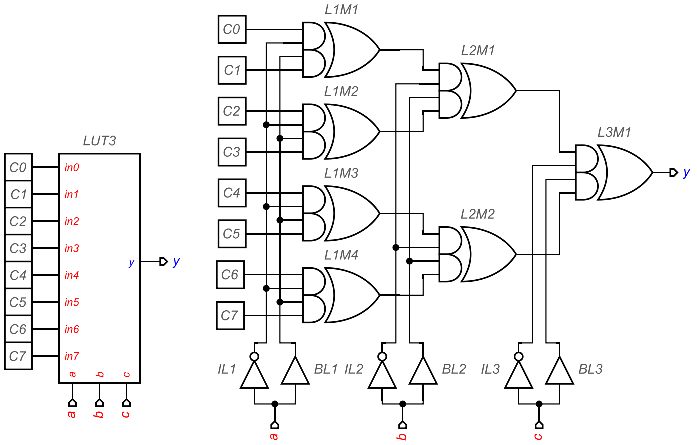
[1] As per original definitions of D.A. Huffman [12] hazards are linked with the property of Boolean functions (functional hazards) or logic gates (logic hazards) to produce spurious transitions in circuits operating in fundamental mode.

[2] In the original work of D.E. Muller [1] a circuit is called SI if it has only one final class of behaviors reachable from the initial state. This is not sufficient to prevent hazards and therefore the concept of semimodularity is introduced.

The problems with self-timed design on FPGAs can be divided into two groups. The first involves problems with hazard-free realization of logic gates, and the second is related to delays in programmable interconnects and wires. In this paper we consider the problems only from the first group.

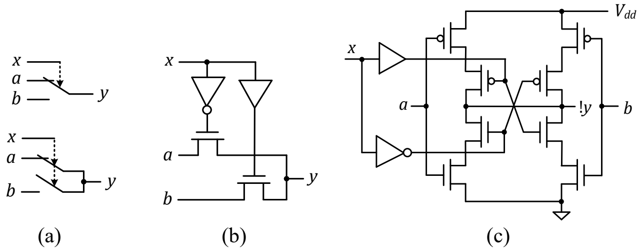
Traditionally QDI circuits are realized in monotone logic basis (see Section II). This is not only because this basis is well-suited for decomposition [3], but also because monotone gates in CMOS technology occupy very small area on a chip. In some sense QDI circuits behave like a latch-based oscillator and therefore minimal basis for their realization differs from that for combinational logic. A proof that for building semimodular autonomous (having no inputs) circuits it is sufficient to have NAND2 and NOR2 gates with a fanout of two is given in [6]. Thus, the inclusion of binate (non-monotone) gates such as XOR and MUX into the realization basis makes the hazard-free QDI design a non-trivial problem, as has been shown, for example, in [8].

Let us recall how FPGAs make use of multiplexers and what the internal circuit of 2-way multiplexers is. A LUT with n inputs is built in an FPGA as a binary tree of $2^n - 1$ elementary 2-way multiplexers as shown in Fig. 1. It should be stressed that the LUT structure presumes that all multiplexers are fanout-free. This will be an important constraint for our implementations. Each gate AO22 in Fig. 1 along with the inverter and buffer is a model of MUX 2:1. Traditional realizations of such a MUX, starting from a relay, are given in Fig. 2. Although the realization on relay may seem too old, there are now LUTs built of nanoelectromechanical (NEM) relays [9]. The on-resistance of a switch should be low in the commutation of 0 and 1, therefore for $V_{dd} \leq 1V$ it is realized on a transmission gate [10].



(a) (b)

Fig. 1: LUT with 3 inputs (a) and its model at the level of logic gates (b).



(a) (b) (c)

Fig. 2: MUX 2:1 based on relay (a), pass-transistor (b) and tristate inverter (c).

This, however, requires twice as many transistors as the pass-transistor circuit in Fig. 2(b). Yet another variant of MUX 2:1 is shown in Fig. 2(c). It is based on tristate inverter and used only to switch the outputs of two adjacent LUTs [11].

Each pair of inverter and buffer in Fig. 1(b) control a single layer of gates. In practice, the delays of the inverter and of the buffer are made approximately equal [10]. The column C0…C7 in Fig. 1(b) corresponds to constants $i_0 \ldots i_7$, which specify the LUT function as follows:

$$y = ((i_0\bar{a} \vee a i_1)\bar{b} \vee b(i_2\bar{a} \vee a i_3))\bar{c} \vee c((i_4\bar{a} \vee a i_5)\bar{b} \vee b(i_6\bar{a} \vee a i_7)) \quad (1)$$

Sometimes (1) is written in the SOP form with minterms $\tilde{a}\tilde{b}\tilde{c}i_j$, where $j = 0 \ldots 7$, and "$\sim$" is a polarity. Let $i_0 \ldots i_7$ in (1) are set for example to [00111111]. This after factoring out, will give us $y = (\bar{a} \vee a)(b\bar{c} \vee c(\bar{b} \vee b)$. In case if the delays of {IL1, BL1} and {IL2, BL2} in Fig. 1(b) are zero, $\bar{a} \vee a = 1$ and $\bar{b} \vee b = 1$. This gives $y = b\bar{c} \vee c$, which turns into $y = b \vee c$ if the delays of {IL3, BL3} are zero. The problem of non-zero delays is linked to logic hazards [12] and discussed in Section IV.

The objective of this paper is to study the feasibility of the MUX basis for realization of hazard-free asynchronous elements, in particular on LUTs with feedbacks. The contribution of this paper is in demonstrating the examples of the MUX 2:1 based circuits, which are hazard-free under different assumptions about delays within MUX 2:1. The paper proposes two types of the circuits. One is *quasi-binate circuits* obtained by mapping unate gates into MUX 2:1. The other type is *binate circuits* built only from XOR and transparent latches. An important contribution of the paper is a set of new realizations of multi-input C-elements. Obtaining these realizations can be described as a combination of structural and behavioral refinements in the Workcraft CAD tool [13].

II. Theoretic Background

In this paper we need the following basic concepts [3]. A QDI circuit is a netlist of logic gates. A logic gate is called *atomic* if it instantaneously evaluates its Boolean function and has finite, but unbounded delay. A Boolean function $f(x_1, \ldots, x_n)$ is called positive [negative] *unate in variable* x_i if $f(x_i = 1) \geq f(x_i = 0)$ $[f(x_i = 1) \leq f(x_i = 0)]$. A function that is not unate in x_i is *binate in* x_i. A function is called positive (negative) *unate* if it is positive (negative) unate in all variables, otherwise it is binate. Positive unate functions are also called monotone. Thus, the MUX 2:1 function $y = a\bar{x} \vee xb$ is binate. A positive unate in x_i function can be represented as $f(\boldsymbol{x}) = f(x_i = 0) \vee x_i f(x_i = 1)$ and therefore is related to the concept of generalized C-element[3]. It is given as:

$$y(\boldsymbol{x}, y) = S(\boldsymbol{x}) \vee y\overline{R}(\boldsymbol{x}) \quad (2)$$

where $S(\boldsymbol{x})$ and $R(\boldsymbol{x})$ are set and reset functions, which satisfy the condition $S(\boldsymbol{x})R(\boldsymbol{x}) = 0$, i.e., $S(\boldsymbol{x})$ and $R(\boldsymbol{x})$ are orthogonal.

The behavior of a QDI circuit can be described in a concise and convenient way by *signal transition graphs* (STGs) [14], [15]. An STG is a particular type of a labeled Petri net, where transitions are associated with the changes in the values of binary signals.

[3] The term "generalized" covers both "symmetric" and "asymmetric" C-elements, depending on whether the set of literals in the set and reset are equal or not [4].

For example, a label “$x+$” is used to denote the transition of a signal x from 0 to 1 (a rising edge), while “$x-$” is used for a 1 to 0 transition (a falling edge). This labeling may differentiate between input, internal, and output signals. The arcs in an STG capture the causal relations between the signal transitions. An STG can include (explicit) Petri net places with multiple input and output transitions. Such an STG describes behavior with choice, which is associated with the non-deterministic selection of input transitions made by the environment. In this paper we consider only the so-called distributive circuits[4] [6], which are described by STGs without places.

There are special algorithms for synthesizing QDI circuits by an STG specification as well as algorithms using this specification to verify the obtained circuits for hazard-freedom. The circuit can be either autonomous or have inputs driven by the environment (cf. compliant operation [1], [6]). For the case of verification, the circuit is converted to the so-called *circuit Petri net* that itself is a type of STG. If the circuit is autonomous, its STG is checked against various properties, for example deadlocks. To verify the circuit having inputs and outputs, its circuit Petri net is composed with the STG model of the environment by means of parallel composition. This forms an STG of the closed system.

In both synthesis and verification the most important property of the STG is *output-persistence* [3]. An STG is called *output-persistent* if every signal transition being enabled eventually fires, that is once enabled transitions cannot be disabled. The internal and output signals of a circuit must be output-persistent, and the environment must provide persistency of the input signals. Stricter definitions can be found in [3]. In this paper we use Workcraft [13] for STG construction, simulation, verification and synthesis of QDI circuits.

It is a common belief that the property of output-persistence guarantees absence of hazards. That is true only in case if a QDI circuit is built of monotone logic gates. However, gates with inverted inputs can exhibit “hidden” hazardous conditions akin to the problems of static logic hazards. This are gates that are binate in some variables or purely binate. Let us consider the latter case by example of the multiplexer function $y = a\bar{x} \vee xb$. If there is a some delay in the inverter for x in the cube $a\bar{x}$, compared to the non-inverting input in cube xb, a takeover between these two cubes may cause a hazard. Let for example $a = b = x = y = 1$ then cube $xb = 1$, while cube $a\bar{x} = 0$. Let now x switches from 1 to 0. Logically, there is a takeover of holding the output y at 1, between xb and $a\bar{x}$. So the state $y = 1$ should be stable. If the delay of $\bar{x}$ is smaller than the delay of the AO22 gate realizing the MUX and the delay model is inertial [3], the effect of takeover will not cause any hazard. Otherwise, there is a potential hazard.

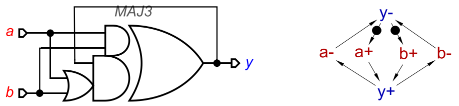
A standard way to prevent this hazard is to introduce a third, consensus cube ab, which would extend the minimal SOP form of the MUX function to the so-called complete sum of prime implicants. Unfortunately, the traditional MUX realizations, like AO22 gate with input inverter, are fixed in their minimal form. Therefore, in our design we should take special precautions. With the aid of Workcraft we can detect cube takeovers for critical transitions that cause violation of the consensus conditions.

Summary on the circuit classification used in this paper:

1) QDI circuits are circuits insensitive to gate delays and wire delays up to the isochronic fork assumption.
2) Semimodular circuits are (closed, i.e. having no inputs) QDI circuits that are free from hazards. They assume all gates being atomic, i.e. delays are attached to gate outputs.
3) Distributive circuits are semimodular circuits which exhibit only AND causality (modelled by STGs without places).
4) Output-persistent circuits are QDI circuits that are free from hazards, but unlike semimodular circuits, they can have inputs. Output-persistence can be checked using Workcraft.
5) Circuits with binate consensus are QDI circuits in which for every 2 terms with a binate variable, there is a consensus cube, which prevents the circuit from a potential hazard.
6) Circuits that have binary consensus violation (such as those using standard MUX) may experience static hazards unless their binate elements are designed appropriately.

III. Problems with Initial Decomposition

Consider a 2-input C-element, defined by $y = ab \vee y(a \vee b)$. Fig. 3(a) shows its realization on an atomic majority (MAJ3) gate with zero-delay feedback. The STG determining the environment of this C-element as two inverters is shown in Fig. 3(b).

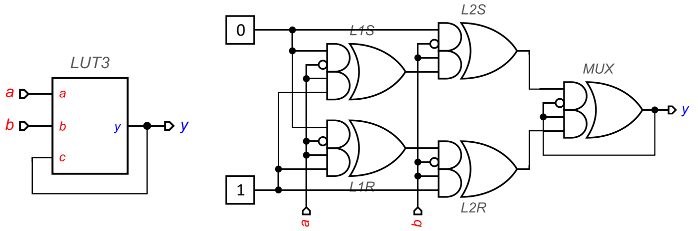


(a) (b)

Fig. 3: C-element realized on atomic MAJ3 gate (a) and its STG (b).

A realization of the MAJ3 gate on a LUT does not guarantee that the obtained C-element would be output-persistent. Moreover, as shown in [16], such a C-element is hazard-free only in case of a single input change (SIC), which is a special case of the protocol shown in Fig. 3(b). However, SIC protocol is apparently used in asynchronous systems built on FPGAs [17], [18], [19].

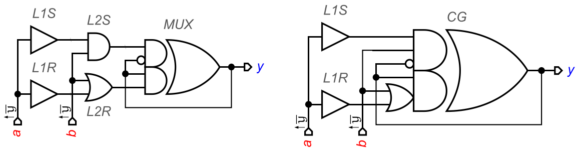
Fig. 4(a) shows a typical implementation of the MAJ3 gate on a 3-input LUT. To analyze it, we use the circuit in Fig. 1(b), where C0…C7 are set to [00010111]. Substituting them into (1), we get $y = ab\bar{c} \vee c(a\bar{b} \vee b(\bar{a} \vee a))$, where $\bar{a} \vee a$ is a problem. Indeed, if IL1 and BL1 in Fig. 1(b) have different delays a hazard 1-0-1 will appear at the output of L1M4. We consider the hazards in more detail in Section IV that contains an important assumption. Based on this assumption, we conclude that L1M4 will be always in 1. Thus, the C-element can be represented by the circuit shown in Fig. 4(b), which is reduced to the circuit shown in Fig. 5(a), if each MUX is atomic and hazard-free.



(a) (b)

Fig. 4: C-element realized on a single LUT3 (a) and its simplified circuit (b).

[4] The term “distributive circuits” stems from [1] and is linked to the distributive lattice that is formed by the so-called cumulative states of the circuit. Later the behavior of distributive circuits was characterized by the class of STGs that has no places with choice and merge [3], [6].



(a) (b)

Fig. 5: The circuit in Fig. 4(b) in another view (a) and merging of its gates (b).

Verifying the circuit in Fig. 5(a) in Workcraft, we get a warning "output persistency is violated" and the following report:

> Event 'a-' disables signal 'L1R'. Event 'b-' disables signal 'L2R'. Event 'L2R+' disables signal 'y'. Violation trace: a+, b+, L1S+, L2S+, y+.

This implies that L1R, L2R and what is most important, the output y were excited, but did not fire. Let us now merge the gates AND and OR with the MUX as shown in Fig. 5(b) and run the verification again. Unfortunately, in this case we get the same warning, but now:

> Event 'a-' disables signal 'L1R'. Violation trace: a+, b+, L1S+, y+

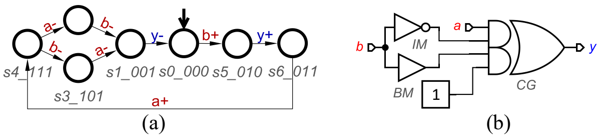
In other words, there is no output hazard now, but the transition of L1R from 0 to 1 is not acknowledged yet. This effect is known in the context of sensitivity to the delays of wires [6]. If an input wire has a fork, the delay of each branch must be zero, otherwise the circuit may lose the property of output-persistence. In the next section we show under what conditions the circuits realized on MUX 2:1 can be hazard-free and output-persistent.

IV. Proposed Approach

The above problems do not allow us to consider the entire LUT as an atomic gate and therefore we need to lower the level of abstraction. If we have already obtained a QDI decomposition of a circuit and want to map it into the MUX basis, we can assume that MUX 2:1 is atomic. This provides output persistence, but hazards may still appear due to internal delays.

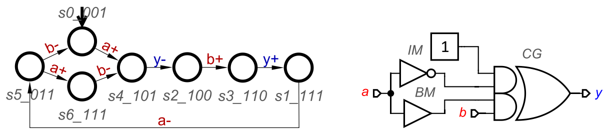
Let for example the variables $[a, b, y]$ change according to the transition diagram shown in Fig. 6(a). If this order of firings is realized on the monotone gate $y = a \vee b$, there are no hazards. However, if we realize the same on the MUX 2:1 model shown in Fig. 6(b), the single input change from 111 to 101 may lead to a hazard 1-0-1 (logic static-1 hazard). Indeed, if IM is slower than BM and b is changing from 1 to 0, then the bottom AND forces the OR to switch from 1 to 0. It can be shown that other input changes (single and multiple[5]) do not cause hazards.

With the transition diagram shown in Fig. 7(a) the situation is a bit different. Here, we realize $y = \bar{a} \vee b$, and the diagram contains two transitions ($a-$ and $a+$) that may lead to hazards in the MUX model shown in Fig. 7(b). For example, if BM is slower than IM and b is in 1, the transition $a+$ leads to a hazard 1-0-1.



(a) (b)

Fig. 6: Transition diagram for $y = a \vee b$ gate (a) and its SOP model with external inverter and buffer (b).



(a) (b)

Fig. 7: Transition diagram for $y = \bar{a} \vee b$ gate (a) and its SOP model with external inverter and buffer (b).

The fact that $a-$ is followed by $a+$ does not present a serious cause for concern since we assume that the delay of propagating the $a-$ through IM and BM is considerably smaller than a concurrent path that exists in the circuit. As will be shown in all examples of our circuits for C-elements, this constraint of relative timing (for definition see [20]) easily holds. Thus, the only concerning case would be the cube takeover described above in Fig. 6 and Fig. 7. Of course, these hazards can be prevented in the standard way, but there is no such an option in FPGAs.

On the other hand, the very fact that the circuit in Fig. 2(c) uses the tristate inverter implies that the MUX can hold the previous state[6] on the output capacitance. Moreover, it is assumed in [8] that the pass-transistor based MUX in Fig. 2(b) holds the previous state longer than the difference between the delays of BM and IM. Based on this assumption we can claim that the mapping of any monotone gate (with no more than 3 inputs) into MUX 2:1 will be hazard-free. The remaining question is how "genuine" binate gates will map into MUX 2:1, which is used for example, to realize $y = a\bar{b} \vee by$. In this case we have to assume that the following conditions are satisfied: transition $b+$ must happen well before $a+$, and similarly, well before $a-$. Then, if $b-$ happens in parallel with $a+$ or $a-$, we can again rely on the temporary retention of the MUX state on the output capacitance.

A general procedure for designing QDI circuits in the MUX 2:1 basis from the STG specification can be formulated as follows:

1) synthesize a complex gate circuit from a given initial STG.
2) if the obtained circuit fits the fanout-free structure of a LUT, map it into MUX 2:1, else refine the STG and go to 1).
3) verify the MUX circuit for the absence of deadlocks and for output persistence. If both of these verifications succeeded, end, else correct the STG and go to 1).

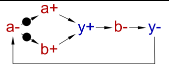
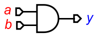
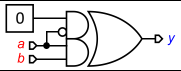
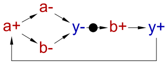
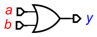
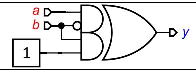
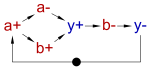
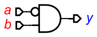
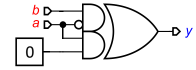
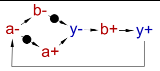
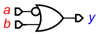
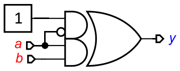
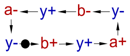
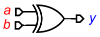
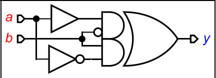
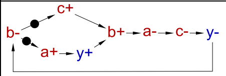
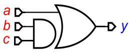
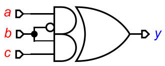
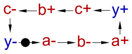
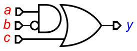
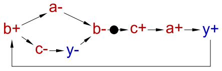
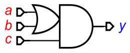
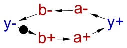
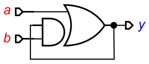
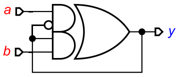
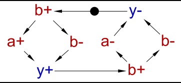
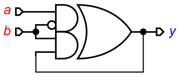
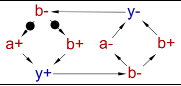
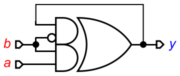
The synthesis here means mapping into gates that are specified in the library. Since MUX 2:1 has 3 inputs, we restrict the Workcraft library by gates having no more than 3 inputs. The second step of the procedure is non-formal and may require a lot of iterations. To get around this obstacle, we start from a circuit that is already realized on simple gates and potentially suited for realization on a LUT. Such a circuits will be called a "prototype". To fit it to the structure of a LUT, we convert "problematic" forks (nodes) to wires. To this end, we first convert the circuit along with the given environment to the circuit Petri net. Then we contract some signal transitions and go to 1) to see how this affects the circuit structure.

Once a suitable circuit has been obtained, we map its gates to the MUX basis using the stamps shown in Table I. Each stamp is specified by an admissible behavior obtained by a projection of an STG describing the entire circuit on the corresponding gate. All the listed behaviors require MUX 2:1 to be hazard-free.

[5] Note that a hazard-free MUX 2:1 can be used for building multilevel circuits that are hazard-free under single and multiple input changes [29], [30].

[6] This state holding is reminiscent of what happens in a dynamic C-element, whose generalization can be used in acyclic charge-storage circuits introduced in [31].

Table I: Some behaviors realized by 2- and 3-input combinational gates and latches, which can be replaced by the corresponding MUX 2:1 stamps.

№	Behavior	Gate/Latch	MUX stamp
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			

It is important to stress that there are such behaviors for AO21 and OA21 gates and based on them latches that cannot be realized by a single MUX 2:1. For example, the STG shown in Fig. 8 describes two unate circuits of a C-element [20], [21] which have no equivalent MUX 2:1 realization. This counterexample allows us to conjecture that if the used MUX 2:1 are hazard-free, then quasi-binate circuits (considered in the next section) is a subset of the circuits realized in the basis of 3-input AOI and OAI gates.

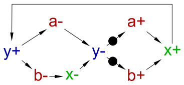


Fig. 8: Behavior that cannot be realized on two MUX 2:1.

V. Quasi-Binate Circuits

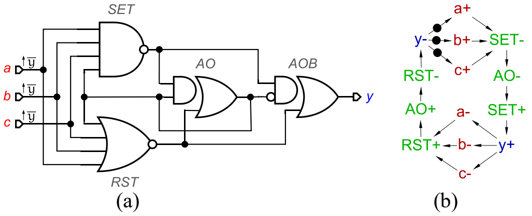
If we consider MUX 2:1 as an atomic gate and assume that it is hazard-free, the problems with realization of a C-element on a LUT are reduced to finding an initial QDI decomposition. Let us recall the basic facts about the operation of a C-element. AND (OR) logic performs AND causal [3] synchronization in the phase 0-1 (1-0). Thus, we can select as prototypes some circuits that exploit this property. The C-element is a latch that uses set (S) and reset (R) signals. We can provide them in two different ways. One is a full-cycle protocol with mutually exclusive S and R. The other way is with overlapping S and R. In both cases we can use the latch from row 9 of Table I, since S and R are unlocked in terms of complete state coding (CSC) [3]. For the first case the order of firings (projection) can be written as:

$$S+\cdots L+\cdots S-\cdots y+\cdots R+\cdots L-\cdots R-\cdots \ y-\cdots \qquad (3)$$

where L is an auxiliary latch, y is an output of C-element, and "…" means transitions of some internal signals. So, in the first case we have to use a combination of a latch (to solve the CSC) and indicator producing the output y. As seen from the following projection, in the second case it is sufficient to have only the output latch y, since the phases of S and R are matched with y.

$$\overline{R}+\cdots S+\cdots y+\cdots S-\cdots \overline{R}-\cdots y-\cdots \qquad (4)$$

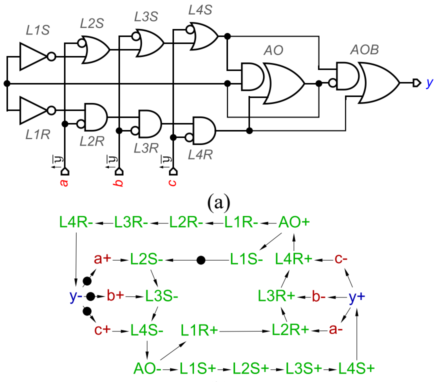
As a prototype for the first case we take the circuit from [22] and use the procedure described in Section IV. The resulting circuit is shown in Fig. 9(a). It is fully symmetric not only by the structure, but also by behavior, as seen from the STG in Fig. 9(b). Since this STG does not contain places, the circuit is distributive. Note that we can detach the bubble from the AOB gate and turn it to an external inverter, without violation of output-persistency. Moreover, the wire connecting the latch AO with the SET and RST gates can have arbitrary delay.



(a) (b)

Fig. 9: Unate circuit of a 3-input C-element (a) and its STG (b).

Let us introduce a *response delay* as the number of transitions of internal signals needed to acknowledge the transition of a certain input at the output. The response delays may differ for transitions "+" and "−" and therefore we separate these delays by "/". For example, the response delay for $a+$ in Fig. 9(b) is determined by trace SET-, AO-, SET+, while for $a-$ it is RST+, AO+, RST-. We write it as "response delay: a±3/3". Since NAND4 and NOR4 in Fig. 9(a) cannot be realized on MUX 2:1, we decompose them as shown in Fig. 10(a). Fortunately, this decomposition preserves output persistence. Now, using the MUX stamps from Table I, we can isomorphically[7] map the unate circuit of Fig. 10(a) into a quasi-binate circuit shown in Fig. 11.



(a)

(b)

Fig. 10: Circuit in Fig. 9(a) after decomposition (a) and its STG (b).

[7] Two circuits that have the same signals and are described by the same STG will be called *isomorphic*.

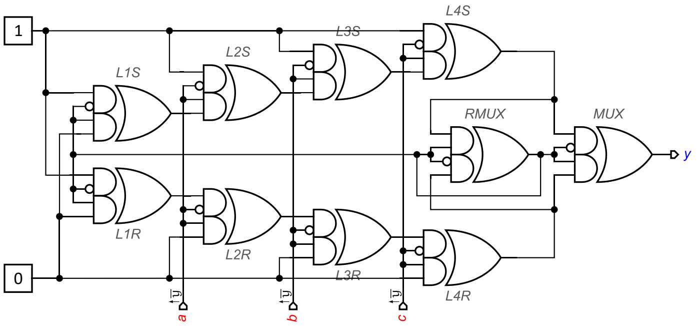


Fig. 11: LUT suited quasi-binate circuit of a 3-input C-element isomorphic to the circuit in Fig. 10(a). Response delays: a±8/8, b±7/7, c±6/6.

To map the circuit in Fig. 11 into an FPGA, we need two 4-input LUTs, since L4S and L4R have a fanout greater than one. Note that the chains L1S…L3S and L1R…L3R are actually diagonal borders in the structure of LUT3 shown in Fig. 1(b). Thus, for L1S…L3S the constants C0….C6 should be set to 1, and C7 to 0. This means that for a 4-input LUT the constants C0…C14 should be set to 1, and C15 to 0. The chain L1R…L4R is realized by the second 4-input LUT, where C0 should be set to 1, and C1…C15 to 0. Note that the constants in Fig. 11 are formed inside the LUTs when C0…C15 pass through MUX 2:1. Since we assume that the multiplexers are hazard-free, the internal constants are stable.

Typically, the outputs of LUTs are connected pairwise to an external multiplexer that realized on tristate inverter as shown in Fig. 2(c). This can be either RMUX or MUX. However, we still need an additional multiplexer and, what is often not available, the corresponding interconnects. Thus, theoretically the circuit in Fig. 11 can be mapped into an FPGA as shown in Fig. 12.

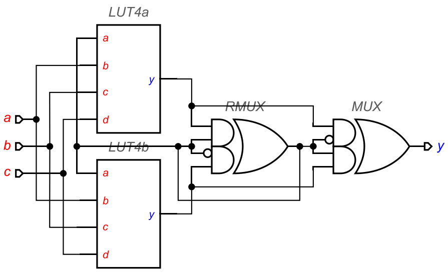


Fig. 12: FPGA implementation of the circuit in Fig. 11.

As in the circuit of Fig. 9(a), the wire connecting RMUX with L1S and L1R in Fig. 11 can have arbitrary delay, therefore we can weaken the requirement for its fork to the inputs of L1S and L1R. If we want to have a larger number of inputs in the quasi-binate circuit in Fig. 11, we can insert the corresponding MUX stamps into the set and reset chains. Moreover, this circuit can be used for realization of asymmetric C-elements like those used in NCL logic [5]. It is evident from the STG in Fig. 10(b) that the shortest response delay is 6, that is the circuit in Fig. 11 is relatively slow.

There is yet another problem that can appear in practice. As evident from Fig. 12, the inputs a, b, c are common for (physically adjacent) LUT4a and LUT4b. This means that the forks in the corresponding wires must be isochronic. Whether this can be accomplished in commercial FPGAs is yet unknown. Let us suppose that we can increase the speed by introducing additional feedbacks. In this case the problem of isochronic forks turns into a problem of minimal delay in each of the feedbacks [6], which for some feedbacks can be solved by relative timing assumptions.

As a prototype for the second case (with overlapping S and R) we take the circuit from [23] and refine it using the procedure from Section IV. This gives the circuit shown in Fig. 13(a).

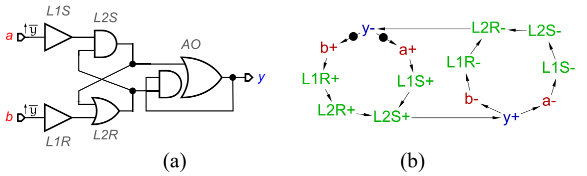


Fig. 13: Unate cross-circuit of a 2-input C-element (a) and its STG (b).

From the STG in Fig. 13(b) we can see that the response delays of a and b are asymmetric. Fig. 14 shows an isomorphic mapping of the unate circuit in Fig. 13(a) into a quasi-binate one.

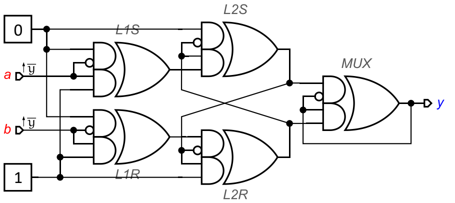


Fig. 14: LUT suited quasi-binate circuit of a 2-input C-element isomorphic to the circuit in Fig. 13(a). Response delays: a±2/3, b±3/2.

Unfortunately, we did not find a method allowing the unate circuit in Fig. 13(a) to have a larger number of inputs with minimum number of cross-feedbacks. The latter is necessary because of the fanout-free structure of a LUT. However, we can apply heuristics and try to combine the ideas behind the circuits in Fig. 9(a) and in Fig. 13(a). For this we use the procedure from Section IV and obtain the circuit shown in Fig. 15(a).

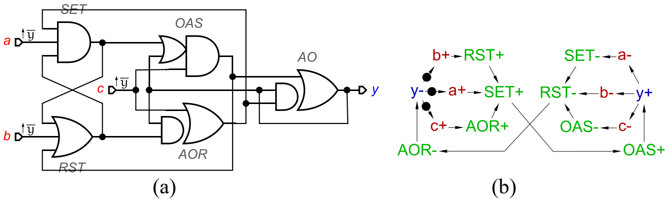


Fig. 15: Unate combined circuit of a 3-input C-element (a) and its STG (b).

Since the STG in Fig. 15(b) does not contain places, the circuit in Fig. 15(a) is distributive. As before, the response delays of a and b are asymmetric, but that of c is symmetric. AND3 and OR3 in Fig. 15(a) should be decomposed into 2-input gates. We can either keep the structural symmetry of feedbacks or make them asymmetric. Fig. 16(a) shows one of the variants of symmetric decomposition obtained by the procedure from Section IV.

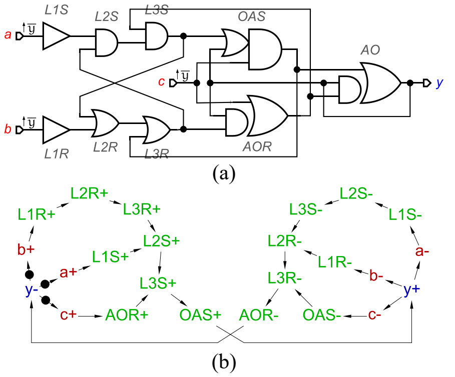


Fig. 16: Circuit in Fig. 15(a) after decomposition (a) and its STG (b).

It is evident from the STG in Fig. 16(b) that the response delays of a, b and c are ±4/6, ±6/4 and ±3/3 respectively. An isomorphic mapping of the unate circuit in Fig. 16(a) into the MUX basis is shown in Fig. 17.

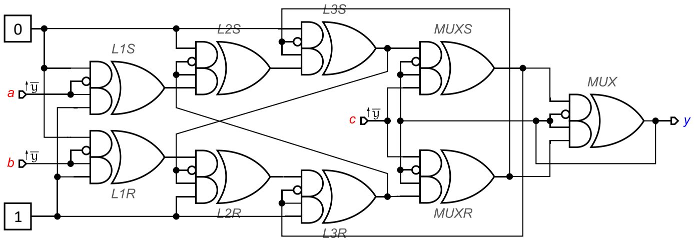


Fig. 17: LUT suited quasi-binate circuit of a 3-input C-element isomorphic to the circuit in Fig. 16(a). Response delays: a±4/6, b±6/4, c±3/3.

VI. Binate Circuit

For realizing a C-element we can also use “genuine” binate gates such as XOR and transparent latches. As a prototype for this case, we take the autonomous circuit of a binary counter [24]. In this circuit we can isolate the functional part of the C-element, while the rest of the parts can be considered as the environment. Thus, we obtain the circuit shown in Fig. 18(a), where the wires represented by DM and DX can have arbitrary delays.

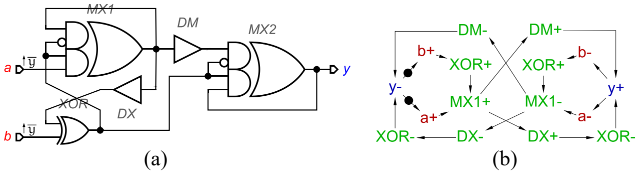


Fig. 18: Binate circuit of a 2-input C-element (a) and its STG (b). Response delays without DM and DX: a±2/2, b±3/3.

As evident from the STG in Fig. 18(b), the signal XOR has 4 transitions that indicates the binate nature of the circuit. It is also evident from this STG that the response delays of a and b are symmetric. Verifying the circuit in Fig. 18(a) in Workcraft, we get violation of binate consensus for both MX1 and MX2. Thus, these multiplexers should be hazard-free. Note that in this binate circuit we need to have access to data inputs of both MUX 2:1. The same access is required in the full adder built of multiplexers and XOR gate. Such an adder is typically used in commercial FPGAs and we hope that in some of them it can be reconfigured to realize the circuit in Fig. 18(a).

VII. Conclusion and Discussion

It has been shown in the paper that hazard-free MUX 2:1 can be used for building the circuits of multi-input C-element. Two new types of the MUX based circuits have been introduced. One is quasi-binate circuits, which are built originally on monotone gates and then mapped into MUX 2:1. We conjecture that these circuits are a subset of the circuits based on 3-input AOI and OAI gates. The circuits of the second type are inherently binate, since they are built of transparent latches and XOR gates. All the presented circuits have been obtained and verified in Workcraft.

The transistor realizations of the multiplexers used in FPGAs have been shown. They differ from the traditional ones realized in the minimal SOP (POS) form by behavior in transients. We assume that the FPGA multiplexers during a race of the control signals are in a high impedance state and hold their previous state on the output capacitance. For the minimal SOP (POS) form this behavior emulates the consensus cube. It is important to notice two specific things related to MUX based circuits. One is that any latch built on pass-transistor based MUX requires a buffer at the output. The other thing is that any MUX with a feedback may start to oscillate under certain conditions, such as stuck-at faults.

Although the proposed circuits can in principle, be realized on FPGAs, there are at least two problems that require future study. One is that certain wires need to have delay smaller than a delay of some path in the circuit. Such a path consists of other wires, interconnects and logic gates. The other problem is that the fork of every input wire inside a LUT should be isochronic. There are several published results that can help in coping with the above problems. The internal delays of a LUT and routing delays within a slice of a commercial FPGA have been characterized in [25]. The latter are also given in [26] along with the routing delays between two slices. As it turns out, a long routing wire carrying a logical 1, reduces the delay of an adjacent long wire. This effect is studied in [27]. Having the exact information about wires and interconnects and wires, we could use assumptions on relative timing, which are available in Workcraft.

Using the MUX basis can be promising from the point of view of new technologies. We have already mentioned the NEM relay based MUX [9], which is closest to atomic gate and similarly to the pass-transistor circuit, is bidirectional. Thus, it can be used for realization of free choice in the circuits and generally speaking, allows revisiting the old idea of bidirectional SI nets [28].

VIII. Future Work

The proposed circuits have been obtained heuristically and require stricter analysis and classification both in terms of hazards and design. More specifically the directions for further studies can be formulated as follows:

1) To analyze programmable interconnects, wires in feedbacks and internal LUT circuit from the point of view of delays, relative timing and isochronic forks.
2) To conduct experiments on real FPGA boards, especially with the “scalable” circuit in Fig. 11. Unfortunately, those FPGAs that contain two detached multiplexers often do not allow forks at the terminals of these multiplexers.
3) To generate all possible projections for MUX 2:1 and then select only those that do not lead to hazards at all or lead only to static hazards.
4) To devise a systematic approach that filters out the circuits, which only seem (by logic transforms) to be in the MUX basis, but do not fulfil the behavioral criteria (such as the output persistence and binate consensus)
5) To compare the quasi-binate basis with 3-input {AOI, OAI} basis and demonstrate how restriction of fanout influences this comparison.

Acknowledgements

We would like to thank Boris S. Tsirlin for discussing the very idea of using multiplexers in semimodular circuits and for his patents. We are also grateful to Danil Sokolov, who realized the binate consensus verification in Workcraft. This research was partially supported by ISF, grant 867/19 and by EPSRC, grant EP/N023641/1 "STRATA".

REFERENCES

[1] D. E. Muller, "Theory of asynchronous circuits," Report no. 66, Digital Computer Laboratory, University of Illinois at Urbana-Champaign, 1955.

[2] A. Martin, "The limitations to delay-insensitivity in asynchronous circuits," in *MIT Conference on Advanced Research in VLSI*, 1990.

[3] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev, Logic synthesis for asynchronous controllers and interfaces, Springer, 2002.

[4] J. Sparsø, Introduction to Asynchronous Circuit Design., TU of Denmark, 2020.

[5] K. M. Fant, Logically Determined Design, Wiley, 2005.

[6] V. I. Varshavsky, Ed., Self-Timed Control of Concurrent Processes, Kluwer, 1990.

[7] W. B. Toms, Synthesis of Quasi-Delay-Insensitive Datapath Circuits. PhD thesis, University of Manchester, 2006.

[8] P. S. Siegel, Automatic Technology Mapping for Asynchronous Designs. PhD thesis, Stanford University, 1995.

[9] T. Qin, S. J. Bleiker, S. Rana, F. Niklaus and D. Pamunuwa, "Performance analysis of nanoelectromechanical relay-based field-programmable gate arrays," *IEEE Access,* vol. 6, pp. 15997-16009, 2018.

[10] C. Chiasson and V. Betz, "Should FPGAs abandon the pass-gate?," in *IEEE Int. Conf. on Field Programmable Logic and Applications*, 2013.

[11] C. Manoj, "Lookup table with relatively balanced delays". Patent US7471104, 30 Dec. 2008.

[12] D. A. Huffman, "The design and use of hazard-free switching network," *Journal of the ACM,* vol. 4, no. 1, pp. 47-62, 1957.

[13] "Workcraft homepage," [Online]. Available: http://workcraft.org.

[14] L. Y. Rosenblum and A. V. Yakovlev, "Signal graphs: from self-timed to timed ones," in *IEEE Int. Workshop on Timed Petri Nets*, 1985.

[15] T. A. Chu, Synthesis of Self-timed VLSI Circuits from Graph-Theoretic Specifications. PhD thesis, Massachusetts Institute of Technology, 1987.

[16] K. Maheswaran and V. Akella, Hazard-free implementation of the self-timed cell set in a Xilinx FPGA. Tech. report, U.C. Davis, 1994.

[17] Q. T. Ho, J. B. Rigaud, L. Fesquet, M. Renaudin and R. Rolland, "Implementing asynchronous circuits on LUT based FPGAs," in *Int. Conf. on Field Programmable Logic and Applications*, 2002.

[18] M. M. Kim and P. Beckett, "Design techniques for NCL-based asynchronous circuits on commercial FPGA," in *IEEE Euromicro Conf. on Digital System Design*, 2014.

[19] Z. Shin, M. H. Oh, H. Kwon, H. Kim and D. Kang, "Implementation of an asynchronous micro-controller on the commercial FPGA," *Int. Journal of Computer Theory and Engineering,* vol. 9, no. 6, pp. 466-472, 2017.

[20] K. S. Stevens, R. Ginosar and S. Rotem, "Relative timing," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems,* vol. 11, no. 1, pp. 129-140, 2003.

[21] N. Starodoubtsev, A. Bystrov and A. Yakovlev, "Semi-modular latch chains for asynchronous circuit design," in *PATMOS Int. Workshop*, 2000.

[22] B. S. Tsirlin, "Multi-input H flip-flop," USSR author's certificate SU1162019, 15 Jun. 1985.

[23] B. S. Tsirlin, "H flip-flop," USSR author's certificate SU1324106, 15 Jul. 1987.

[24] J. C. Nelson, "Speed-independent counting circuits," Report no. 71, Digital Computer Laboratory, University of Illinois at Urbana-Champaign, 1956.

[25] B. Gojman, S. Nalmela, N. Mehta, N. Howarth and A. DeHon, "GROK-LAB: Generating real on-chip knowledge for intra-cluster delays using timing extraction," *ACM Trans. on Reconfigurable Technology and Systems,* vol. 7, no. 4, pp. 1-23, 2014.

[26] J. V. Manoranjan and K. S. Stevens, "Burst-mode asynchronous controller implementation on FPGA using relative timing," in *Southern Conf. on Programmable Logic (SPL)*, 2014.

[27] I. Giechaskiel and J. Szefer, "Information leakage from FPGA routing and logic elements," in *ACM/IEEE Int. Conf. on Computer-Aided Design*, 2020.

[28] W. D. Frazer, A Switching Theory for Bilateral Nets of Threshold Elements, PhD thesis, University of Illinois at Urbana-Champaign, 1963.

[29] B. Lin and S. Devadas, "Synthesis of hazard-free multilevel logic under multiple-input changes from binary decision diagrams," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol. 14, no. 8, pp. 974-985, 1995.

[30] S. Jukna, "Notes on hazard-free circuits," *SIAM Journal on Discrete Mathematics,* vol. 35, no. 2, pp. 770-787, 2021.

[31] D. E. Muller and F. P. Preparata, "Toward a switching theory of CMOS circuits," in *Fall Joint Computer Conf. on Exploring Technology*, 1987.