Tool for topological reliability analysis of reversible logic circuits

Peter Sedlacek

***Abstract*—**It is assumed that progress with classical logic gates development can have its boundaries in the near future. Therefore, it is needed to make research with other approaches. One of them is application of reversible logic. In this paper, we present a method for reliability analysis of logic circuits composed of reversible logic gates. The method is based on structure function, and we implemented it in a software tool that also allows us to design reversible logic gates and circuits.

Keywords **–** reversible logic, logic circuit, reliability analysis

I. INTRODUCTION

Reversible logic circuits are getting to the foreground in presence, as it is assumed that progress with present irreversible logic gates can have its limits in the near future [1].

The computing devices loss part of information in the process of computing. Irreversible logic functions are always connected with physical non-reimbursement, and so, there is a minimal amount of heat generated by the device for every irreversible function (Fig. 1). This disappearing serves for signals to become independent of their history. But it is possible for computing to run without this loss of information. This can be ensured that way, calculations will be reversible, what means, it is possible to return from any state to any previous. Landauer pointed out, that majority of physical laws are reversible, so if you have the full information about state of closed system in some time, it is possible, at least in principle, to perform these laws in reverse order and get exact state of system in any previous time [2].

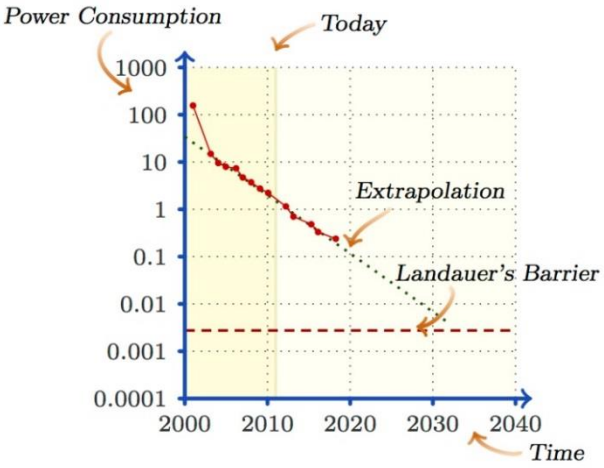


Fig. 1 Landauer's barrier and trend in energy consumption [3]

In 1973 Charles Bennett in [4] showed that it is possible to build full reversible computer capable to perform any calculations without needing of overwhelm memory with temporary data. He reverted operations, that produce temporary result. Reversible computer designed this way would be in principle capable to cross Landauer's barrier, but design of a more complex reversible computer is demanding.

An important aspect when designing a reversible logic circuits is its reliability, which is the main subject of this paper.

P. Sedlacek, Faculty of Management science and Informatics, University of Zilina, Slovakia (e-mail: Peter.Sedlacek@st.fri.uniza.sk)

II. Reliability Analysis

A. Structure Function

The structure function is used for mathematic description of system and reflects system operation depending on its components and theirs states [5]. This function is defined as a map of the following form:

$$\phi(x_1, x_2, \dots, x_n) = \phi(\boldsymbol{x}): \{0,1\}^n \to \{0,1\}, \tag{1}$$

where n is a number of components in the system, x_i is a state of component i, for $i = 1, 2, \dots, n$, and $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ is a vector of components states (state vector).

The structure function can be used to calculate system availability or perform importance analysis, which deals with evaluation of influence of system components on the system [5]. However, the key issue is how to find or construct the structure function of a system.

B. Structure Function of Logic Circuit

Reliability analysis of logic circuits based on structure function has been investigated in [6]. The authors of the paper proposed a method for finding the structure function of a logic circuit. They assumed that the relevant components of a circuit are logic gates that the circuit is composed of. For a specific combination of input signals (e.g., (1,1) in case of a gate with two inputs), a gate (e.g., AND with two inputs) is working if its output agrees with the expected output (1 in this specific case). If the output does not agree with the expected one, then the gate is considered to be non-working for the specific combination of the input signals. Similarly, the logic circuit is functioning for a specific combination of input signals if its output agrees with the expected one. Otherwise, it is failed. This indicates that the functionality (and so the structure function) of a logic circuit depends not only on states of its components (logic gates) but also on values of the input signals of the circuit.

To show how the structure function of a logic circuit looks like, let us consider a logic circuit composed of n logic gates, and let us assume that the circuit has k inputs and m outputs. The expected output of the circuit is defined by a vector-valued function of the following form:

$$\boldsymbol{F}(y_1, y_2, \dots, y_k) = \boldsymbol{F}(\boldsymbol{y}) = \big(F_1(\boldsymbol{y}), F_2(\boldsymbol{y}), \dots, F_m(\boldsymbol{y})\big): \{0,1\}^k \to \{0,1\}^m, \tag{2}$$

where y_l defines value of the l-th input signal, for $l = 1, 2, \dots, k$, $\boldsymbol{y} = (y_1, y_2, \dots, y_k)$ is a vector of input signals (input vector), $F_t(\boldsymbol{y})$ agrees with output t, for $t = 1, 2, \dots, m$, and $\boldsymbol{F}(\boldsymbol{y}) = \big(F_1(\boldsymbol{y}), F_2(\boldsymbol{y}), \dots, F_m(\boldsymbol{y})\big)$ represents a vector of functions defining values of the output signals. This vector-valued function is created based on the logic gates and their expected behavior. However, if gates are unreliable, i.e., they can fail and generate an output that does not agree with the expected one, the output of the circuit depends also on states of the gates. In this case, the output of the circuit agrees with the following map:

$$\boldsymbol{F}_o(\boldsymbol{y}; \boldsymbol{x}) = \boldsymbol{F}(\boldsymbol{y}) = \Big(F_{1,o}(\boldsymbol{y}; \boldsymbol{x}), F_{2,o}(\boldsymbol{y}; \boldsymbol{x}), \dots, F_{m,o}(\boldsymbol{y}; \boldsymbol{x})\Big): \{0,1\}^{k+n} \to \{0,1\}^m, \tag{3}$$

where $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$ is a state vector defining states of the logic gates, $F_{t,o}(\boldsymbol{y}; \boldsymbol{x})$ agrees with real output t, for $t = 1, 2, \dots, m$, and $\boldsymbol{F}_o(\boldsymbol{y}; \boldsymbol{x}) = \Big(F_{1,o}(\boldsymbol{y}; \boldsymbol{x}), F_{2,o}(\boldsymbol{y}; \boldsymbol{x}), \dots, F_{m,o}(\boldsymbol{y}; \boldsymbol{x})\Big)$ represents a vector of functions defining real values of the output signals.

Based on (2) and (3), the structure function of a logic circuit has been defined in [6] as the equivalence of the real outputs of the circuit (which assumes that the gates are unreliable) and the expected ones (which assumes that the gates are perfectly reliable), i.e.:

$$\begin{aligned} \phi(\boldsymbol{x}; \boldsymbol{y}) &= \boldsymbol{F}_o(\boldsymbol{y}; \boldsymbol{x}) \leftrightarrow \boldsymbol{F}(\boldsymbol{y}) \\ &= \big\{F_{1,o}(\boldsymbol{y}; \boldsymbol{x}) \leftrightarrow F_1(\boldsymbol{y})\big\} \wedge \big\{F_{2,o}(\boldsymbol{y}; \boldsymbol{x}) \leftrightarrow F_2(\boldsymbol{y})\big\} \wedge \dots \wedge \big\{F_{m,o}(\boldsymbol{y}; \boldsymbol{x}) \leftrightarrow F_m(\boldsymbol{y})\big\}, \end{aligned} \tag{4}$$

where ↔ is logical biconditional and ∧ logical conjunction. So, the structure function of a classic logic circuit can be obtained as the conjunction of the Boolean functions defining the situations in which the output signals of the circuit agrees with the expected ones. Please note that this function is quite different from (1) since its output depends not only on states of the components (state vector $\boldsymbol{x}$) but also on the environment, which is included in input vector $\boldsymbol{y}$.

C. Structure Function of Reversible Logic Circuit

Reliability analysis of reversible logic circuit can be done in similar way as in the case of irreversible logic circuit presented above. In general, an irreversible logic gate performs one logic operation. Output of logic gate can be interpreted as value 0 or 1. On the other hand, reversible logic gate performs multiple logic functions. Output of such gate is now vector of values. As failure state we consider when output vector is different from the expected output vector at least in one element.

The other difference is that reversible logic circuits contains garbage bits and constant inputs. Garbage bit is an output of logic circuit that is not connected, i.e., we are not interested whether this output agrees with the expected one or not. On the other hand, a constant input is an input set to a specific value, so we do not have to verify cases when this input is set to other value. So, the set of output vectors we have to explore is smaller. This implies that the structure function of a reversible logic circuit can be defined as follows:

$$\phi(\boldsymbol{x};\boldsymbol{y}_r) = \boldsymbol{F}_o(\boldsymbol{y};\boldsymbol{x}) \leftrightarrow \boldsymbol{F}(\boldsymbol{y}) = \wedge_{t\in\mathbf{R}}\{F_{t,o}(\boldsymbol{y};\boldsymbol{x}) \leftrightarrow F_t(\boldsymbol{y})\}, \tag{5}$$

where $\boldsymbol{y}_r$ is a vector of input signals from which the constant signals are excluded and **R** is a set of the output signals from which the garbage bits are excluded. This formula implies if a failure of a logic gate of the circuit causes that just garbage bits are different from the expected values, then this situation will not be considered as a failure of the reversible circuit.

D. Example

For illustration of obtaining the structure function of a reversible logic circuit, let us consider an example of a full adder from [7], which is composed of two Modified Islam Gates (MIGs) and one Controlled Operation Gate (COG) introduced in [8]. This circuit is depicted in Fig. 2. The block diagrams defining operation of MIG and COG are shown in Fig. 3, where symbol ⊕ denotes logic operation XOR and symbol ′ agrees with logic complement, i.e., NOT. As we mentioned before, an unreliable gate performs different logic function as working one. For our example, let us assume, the unreliable gate performs function defined by following formula: $\boldsymbol{F}(\boldsymbol{y}) = \mathbf{0}$. That means output of a broken gate is always $\mathbf{0}$. We can notice that the circuit in Fig. 2 has 2 constant inputs (denoted by symbol 0) and 4 garbage outputs (denoted as g_1, g_2, g_3, and g_4). They are not interesting for reliability analysis.

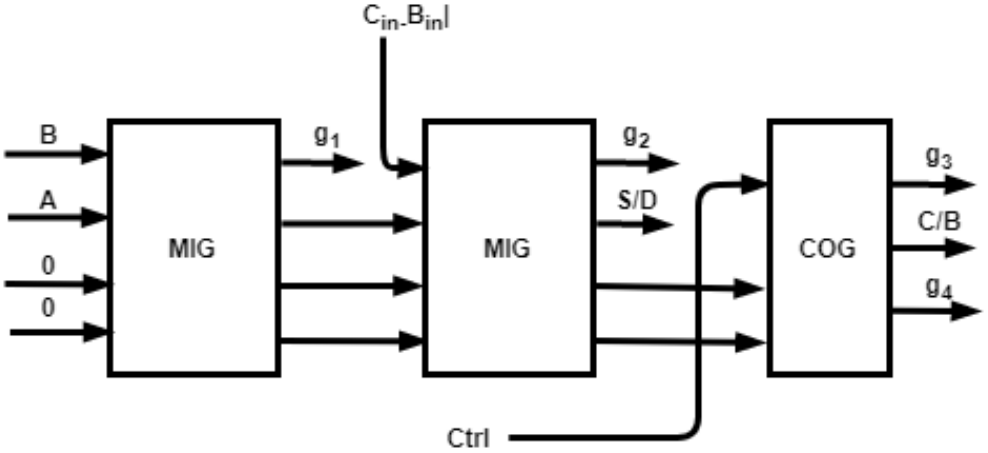
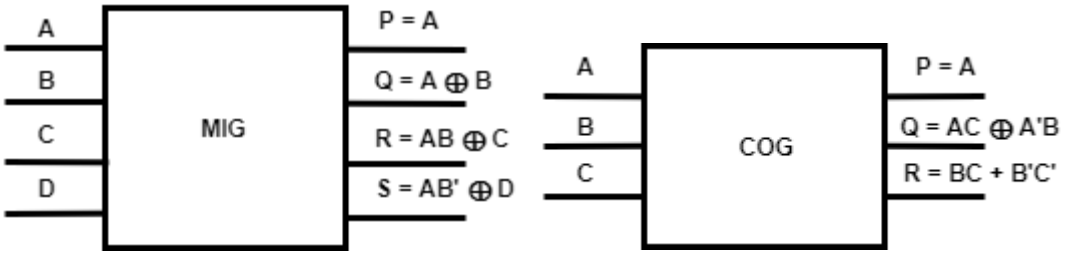


Fig. 2 Full adder using MIG and COG gates (according to [7])



(a) Modified Islam gate (b) Controlled operation gate

Fig. 3 Block diagrams of reversible logic gates (according to [7])

To find structure function of the circuit, we build two truth tables. The first one contains all combinations of input values and their respective output values (Table 1). This table describes situation when all components are functioning, i.e., it defines the expected outputs of the circuit for individual combinations of the input signals. The second table is extended by all combinations of logic gate states and respective output values. This one describes real output values of logic circuit when a specific gate is in function or failure state (Table 2). We can notice that none of these tables contain columns for constant inputs and garbage outputs, as we mentioned before.

Table 1 Selected part of truth table for full adder in Fig. 2

A	B	$C_{in}\_B_{in}\|$	$Ctrl$	S/D	C/B
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	1	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	0

Table 2 Selected part of truth table including failure states for full adder in Fig. 2

A	B	$C_{in}\_B_{in}\|$	$Ctrl$	x_1	x_2	x_3	S/D	C/B	ϕ
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	1	0	1	0
0	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	1	0	1	0
0	0	0	0	1	1	0	0	1	0
0	0	0	0	1	1	1	0	0	0
0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	1	0	0	1
0	0	0	1	1	1	1	0	0	0
0	0	1	0	0	0	1	1	1	0

Based on the tables introduced above, we can find the structure function of the circuit. This can be done by comparing the expected values of outputs (from Table 1) with their real values (from Table 2). If these values match, the structure function will acquire value 1, otherwise, it will take value 0. For illustration, we placed values of the structure function of the circuit into the last column of Table 2.

Based on the structure function, we can compute several reliability measures. One of them is topological availability, which was introduced in [9] as the relative frequency of system state 1. It can be computed simply as a proportion of states at which the structure function takes value 1. In our case, it is computed as value $56/128$ because the structure function of the circuit is defined with respect to 7 variables (A, B, $C_{in} - B_{in}|$, $Ctrl$, x_1, x_2, and x_3), i.e., it is defined at $2^7 = 128$ different points, and it takes value 1 in 56 out of these points.

III. Implementation of Tool for Reliability Analysis of Reversible Logic Circuits

We implemented the method for obtaining the structure function of a reversible logic circuit that was described above in a form of a software tool. We set several goals that our tool should do. Firstly, we should be able to create and modify reversible logic gates. Then we should be able to create reversible logic circuits from these gates and, finally, we should be able to find the structure function of the circuit and, based on it, compute its availability. We decided to implement it in C# language.

A. Model-View-View-Model (MVVM) Architecture

We decided to implement our tool using MVVM architecture [10] depicted in Fig. 4 as it allows relatively high independence of individual layers, e.g., a complete redesign of user interface will not require change in model and vice versa. As we can see in Fig. 4, this architecture is composed of the following three layers:

1) *Model*, which is a representation of business objects. It is optimized for logic relationships and operations between individual entities without its representation in user interface. The Model communicates with View-Model layer via events.
2) *View*, which is a representation of user interface. It displays information for users and responses to user inputs sends to View-Model layer.
3) *View-Model*, which is a bridge between Model layer and View layer. Every class of the View has its corresponding class in View-Model. This layer obtains information from the Model and processes it into a form suitable for the View. Simultaneously, it responds on requests from the View by updating the Model.

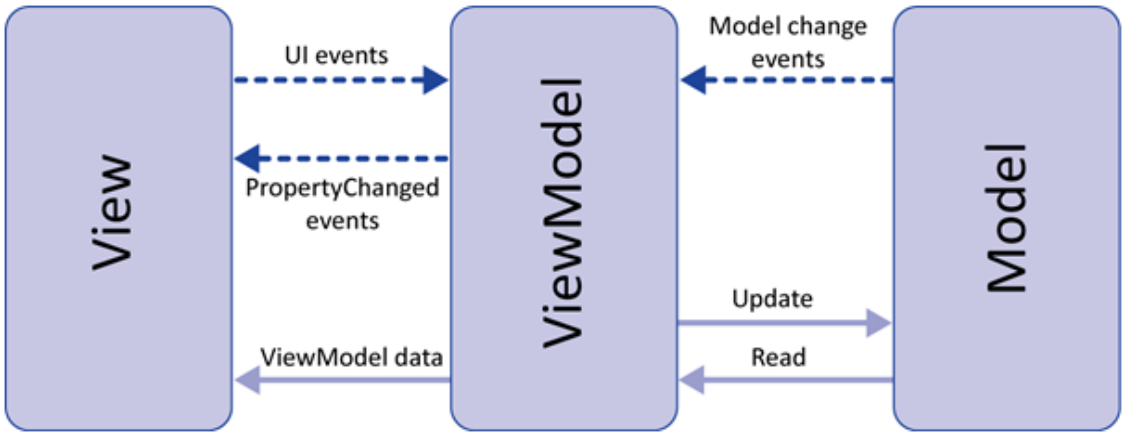


Fig. 4 Diagram of MVVM architecture [10]

B. User Interface

Based on the requirements specified above, our tool has three main components that are:

1) *editor for creating logic circuits* (Fig. 5) – this component allows us to build a logic circuit from the gates, which are located in the left part of the editor. We are able to add inputs in logic circuits and connect individual components to each other. We are also able to turn on/off inputs and see values of the output signals.

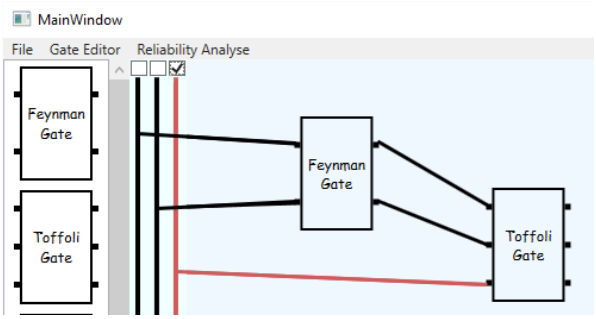
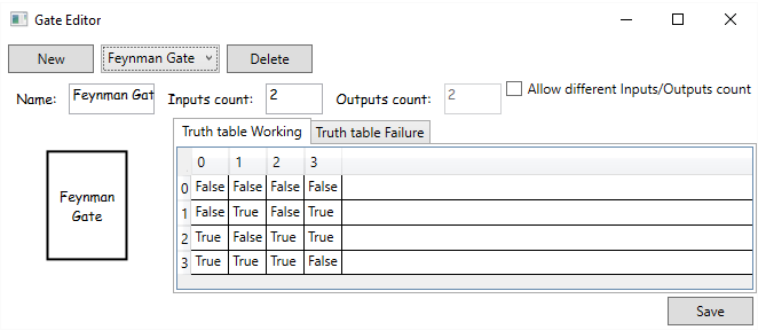


Fig. 5 Editor for creating logic circuits

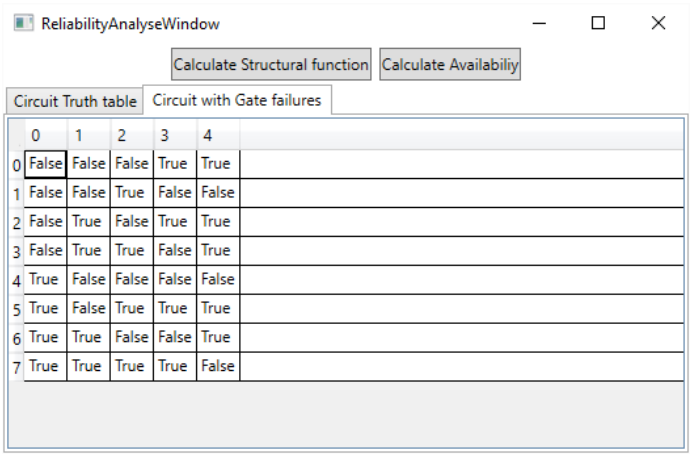
2) *gate editor* (Fig. 6) – this component is used for creating and editing logic gates. Gates are identified by their names. A function that the gate performed is specified by truth table. We can also specify function for failure state of the gate. A logic gate created by this editor is stored into logic gate library, which is responsible for reading/saving gates into file and accessing gates upon request.



	0	1	2	3
0	False	False	False	False
1	False	True	False	True
2	True	False	True	True
3	True	True	True	False

Fig. 6 Gate editor

3) *reliability analysis calculator* (Fig. 7) – it works with two truth tables as we described above. Based on these tables, it is able to find the structure function of the circuit that is created using the editor mentioned above. Based on the structure function, it allows us to calculate topological availability of the circuit.



	0	1	2	3	4
0	False	False	False	True	True
1	False	False	True	False	False
2	False	True	False	True	True
3	False	True	True	False	True
4	True	False	False	False	False
5	True	False	True	True	True
6	True	True	False	False	True
7	True	True	True	True	False

Fig. 7 Reliability analysis calculator – table including failure states

IV. Conclusion

In this paper, we proposed a method for topological reliability analysis of reversible logic circuits based on the method described in [6]. It uses the same approach, i.e., a logic circuit is considered to be working correctly if and only if all real outputs match with the expected ones for a given combination of inputs. The main difference in reliability analysis of reversible circuits is by using garbage bits and constant inputs that are not used in reliability analysis as we do not consider as failure state when real value of a garbage bit is different from the expected one and do not take into account a possibility of a change of a value of constant inputs.

We implemented the method for analysis of reversible logic circuits in our own tool that was also presented in this paper. The tool allows us to create and modify reversible logic gates, design logic circuits from created gates, find the structure function of the circuit and investigate its topological availability based on the structure function.

Acknowledgment

This research was implemented based on result developed in framework of the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 610425(RASimAs)

References

[1] M. P. Frank: *Throwing Computing into Reverse,* IEEE Spectrum (Volume: 54, Issue: 9, September 2017)
[2] R. Landauer: *Irreversibility and Heat Generation in the Computing Process,* IBM Journal of Research and Development (Volume: 5, Issue: 3, July 1961), pp 183-191.
[3] R. Hanson: *Slowing Computer Gains,* http://www.overcomingbias.com/2013/03/slowing-computer-gains.html [accessed 2018-04-28].
[4] C. Bennett: *Logical Reversibility of Computation,* IBM, Journal of Research and Development (Volume: 17, Issue: 6, 1973), pp. 525-532.
[5] W. Kuo and X. Zhu: *Importance Measures in Reliability, Risk, and Optimization: Principles and Applications,* Wiley, Chichester, UK, 2012.
[6] M. Kvassay, E. Zaitseva, V. Levashenko, and J. Kostolny: *Reliability analysis of multiple-outputs logic circuits based on structure function approach,* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 36, Issue: 3, 2017) pp. 398-411.
[7] J. B. Chacko and P. Whing: *Low Delay based Full Adder / Substractor by MIG and COG Reversible Logic Gate,* Computation Intelligence and Communication Networks (October 2017).
[8] S. Mamataj, B. Das, A. Rahaman: *An Ease Implementation of 4-Bit Arithmetic Circuit for 8 Operation by using a New Reversible Cog Gate,* International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering (Volume: 3, Issue: 1, January 2014).
[9] M. Kvassay and E. Zaitseva: *Topological Analysis of Multi-state Systems Based on Direct Partial Logic Derivatives,* in Recent Advances in Multi-state Systems Reliability, A. Lisnianski, I. Frenkel, and A. Karagrigoriou, Eds. Cham, CH: Springer International Publishing, 2018, pp. 265–281.
[10] *Implementing the Model-View-ViewModel Pattern,* https://msdn.microsoft.com/en-us/library/ff798384.aspx [accessed 2018-04-28].