Mealy-to-Moore Transformation

Mustafa Özgül, Florian Deeg, Sebastian M. Sattler
Lehrstuhl für Zuverlässige Schaltungen und Systeme
Friedrich-Alexander-Universität Erlangenen-Nürnberg
Paul-Gordan-Str. 5, 91052 Erlangen, Deutschland
Email: {mustafa.oezguel,florian.deeg,sebastian.sattler}@fau.de

Abstract

In this paper we will show a method for transforming an asynchronously feed-backed Mealy machine into an equivalent Moore machine under use of dual-rail logic and the RS-Buffer. The resulting machine will be safe, stable and reproducable. We will further present a use-case to demonstrate the before mentioned transformation.

Keywords - asynchronous feedback, functional safety, stabilization, hazard-free, parallel de-composition, dual-rail

I. Introduction

ASYNCHRONOUS CIRCUITS are gaining significant importance in circuit design. This growing need for asynchronous circuits results from a number of benefits. These are that asynchronous circuits have a better system performance, lower power consumption and reduced electromagnetic emissions. Also asynchronous circuits can be modularly designed and have no problems with clock skew and related issues [4]. This paper presents the transformation of a Mealy machine into an equivalent Moore machine for function stabilized modeling of asynchronous automata. The Mealy machine with the state transfer function δ and the output function λ, see figure 1 be given. Each Mealy machine can be transformed into an equivalent Moore machine [3] for example by increasing the arity of δ by $|x|$ and coding it with x, $z' = (z,x)$ with $\delta'(z',x) \mapsto (\delta(z,x),x)$, and the output function is only dependent on the new state variable z' with $\mu : (z') \mapsto \mu(z')$.

By comparing the two machines the pros and cons can be shown. Therefore the idea of transforming the machines into each other to profit from the benefits of both is obvious. Mealy machines have the advantage of requiring less states since one state can produce a number of different outputs in combination with the input. A Moore machine's state on the other hand only produces one output. A Mealy machine is also faster by reacting directly to the input. This feature however is not always wanted, since it can lead to undesired outputs (e.g. hazards, glitches) when the input is variable. A Moore machine is more stable in this regard, since it only indirectly reacts to input changes. The output only changes when transferring into the next state. Transforming a Mealy machine into a Moore machine is therefore useful in case a direct dependence on the input is to be avoided [2].

In the paper the transformation of a Mealy machine into a Moore machine is presented. In this transformation the state of the transformed Moore machine is feedbacked at the output. In order to set the correct state at the input of the state transfer function of the Moore, a function will be integrated. This function will generate the initial state from the output signal of the Mealy machine. For function stable asynchronous machines this can be done via dual-rail logic and the RS-Buffer [4]. With the method presented in this article, function stable circuit parts can be abstracted as blocks and moved over other blocks at will. With this method, individual machines can be strung together to realize complex circuits for safety relevant applications.

II. Theory

This paper describes the underlying theory of the transformation and provides an illustrative example.

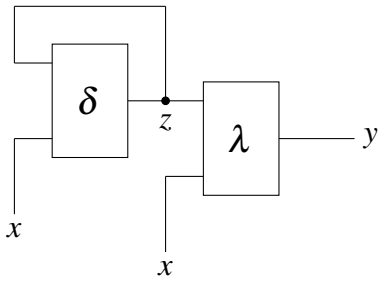

Figure 1: Fully asynchronous Mealy machine

Figure 1 shows a fully asynchronous Mealy machine. The branches entering a node in the graph should end reflexively, so that only transient states are allowed which are conscious and triggered from the outside [2]. To consider all branches of the Mealy machine as locally reflexively concluded, the feedback should be moved over λ, making the output y the feedback. In order to set the correct state z for the transfer function δ, a function λ^{-1} is realized. This function generates the reduced state $\underline{z}$ from the feedback y. This equivalent transformation is outlined in figure 2.

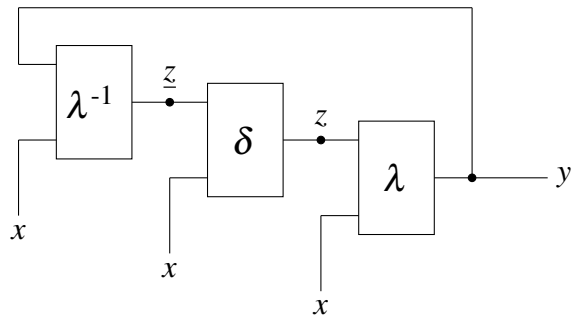

Figure 2: Equivalent transformation

The following applies:

$$\begin{aligned} \delta(\underline{z},x) &= \underline{z} \\ \delta\left(\lambda^{-1}(y,x),x\right) &= \lambda^{-1}(y,x) \\ \lambda\left(\delta\left(\lambda^{-1}(y,x),x\right),x\right) &= y \\ \text{mit } \lambda\left(\lambda^{-1}(y,x),x\right) &= y \end{aligned}$$

A. Dual-rail logic

For the implementation in dual-rail logic [1, 5], the fully asynchronous circuit from figure 1 will be divided in the 1- and 0-share. This is done by partitioning the state transfer function and the output function, see figure 3.

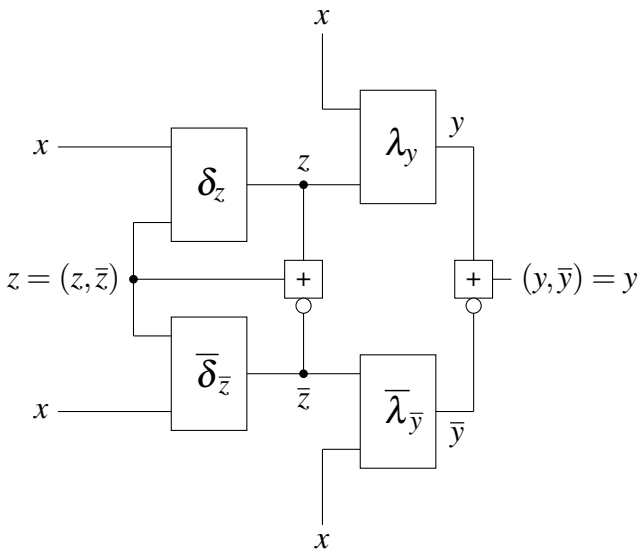

Figure 3: Mealy realized in dual-rail logic

The functions δ and λ are each w.l.o.g. realized in two blocks $\delta = (\delta_z, \overline{\delta}_{\overline{z}})$ and $\lambda = (\lambda_z, \overline{\lambda}_{\overline{z}})$. In order to guarantee this secure dual rail structure, RS-Buffers [4] are used.

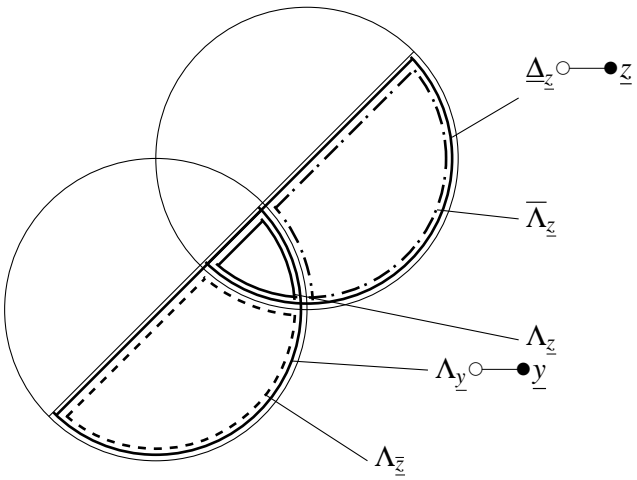

Figure 4: Venn diagram of 1-states in 1-outputs

The stabilized 1-states, $\underline{\Delta}_{\underline{z}} \circ\!\!-\!\!\bullet \underline{z}$, are transformed into 1-outputs, $\Lambda_{\underline{y}} \circ\!\!-\!\!\bullet \underline{y}$. The corresponding Venn diagram of the stabilized 1-states in 1-outputs can be seen in figure 4. 1-states that appear as 0-outputs are declared as $\overline{\Lambda}_{\underline{z}}$, 0-states which appear as 1-outputs are declared as $\Lambda_{\underline{\overline{z}}}$. The 1-partition is composed of:

$$\Lambda_{\underline{y}} \circ\!\!-\!\!\bullet \lambda_y(\underline{\delta}_z) + \lambda_y(\underline{\overline{\delta}}_{\overline{z}}) + \overline{\lambda}_{\overline{y}}(\underline{\delta}_z)$$

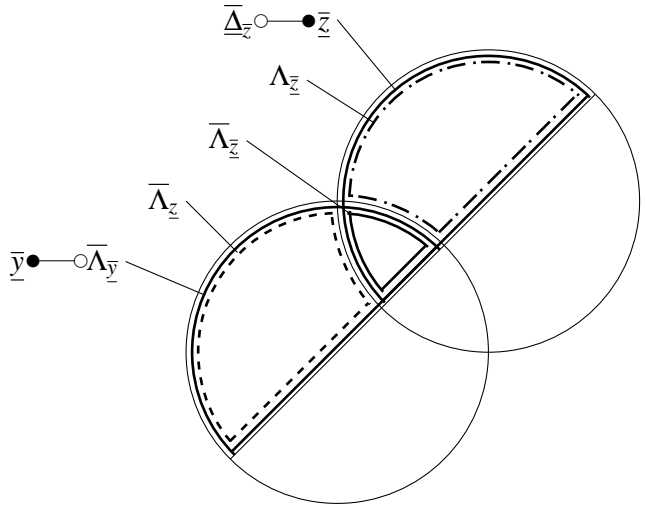

Figure 5: Venn diagram of 0-states in 0-outputs

The stabilized 0-states, $\overline{\underline{\Delta}}_{\overline{z}} \circ\!\!-\!\!\bullet \underline{\overline{z}}$, are transformed into 0-outputs, $\overline{\Lambda}_{\underline{\overline{y}}} \circ\!\!-\!\!\bullet \underline{\overline{y}}$. The corresponding Venn diagram of the 0-states in 0-outputs is shown in figure 5. 0-states which appear as 1-outputs are declared as $\Lambda_{\underline{\overline{z}}}$, 1-states which appear as 0-outputs are declared as $\overline{\Lambda}_{\underline{z}}$. The 0-partition is composed of:

$$\overline{\Lambda}_{\underline{\overline{y}}} \circ\!\!-\!\!\bullet \overline{\lambda}_{\overline{y}}(\underline{\overline{\delta}}_{\overline{z}}) + \overline{\lambda}_{\overline{y}}(\underline{\delta}_z) + \lambda_y(\underline{\overline{\delta}}_{\overline{z}})$$

III. Use-Case

For a better understanding, an example is shown below. The Mealy machine from figure 6 be given

$$\begin{aligned}\delta(z,x) &= zx_0\bar{x}_1 \vee x_1 \\ &= \underbrace{z(x_0 \vee x_1)}_{\underline{z}} \vee \bar{z}x_1 \\ \lambda(z,x) &= zx_0x_1 \vee \bar{x}_0 \\ &= \underbrace{z(x_1 \vee \bar{x}_0)}_{\underline{y}} \vee \overline{z}\overline{x}_0\end{aligned}$$

with the stabilized 1-states of z, $\underline{\Delta}_z \circ\!\!-\!\!\bullet z(x_0 \vee x_1)$, the stabilized 1-states $\Lambda_{\underline{z}} \circ\!\!-\!\!\bullet zx_1$, which appear as 1-outputs, the stabilized 1-states $\overline{\Lambda}_{\underline{z}} \circ\!\!-\!\!\bullet zx_0\bar{x}_1$, which appear as 0-outputs and the stabilized 0-states $\Lambda_{\overline{\underline{z}}} \circ\!\!-\!\!\bullet \bar{z}\bar{x}_0\bar{x}_1$, which appear as 1-outputs

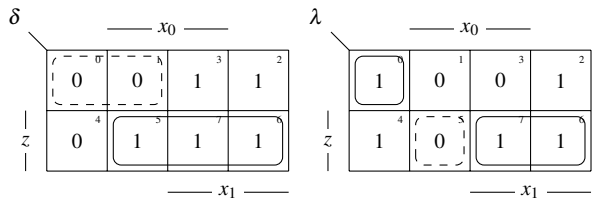

Figure 6: State transfer function and output

and the stabilized 0-states of z, $\overline{\underline{\Delta}}_{\overline{\underline{z}}} \circ\!\!-\!\!\bullet \bar{z}\bar{x}_1$, the stabilized 0-states $\overline{\Lambda}_{\overline{\underline{z}}} \circ\!\!-\!\!\bullet \bar{z}x_0\bar{x}_1$, which appear as 0-outputs, the stabilized 0-states $\Lambda_{\overline{\underline{z}}} \circ\!\!-\!\!\bullet \bar{z}\bar{x}_0\bar{x}_1$, which appear as 1-outputs and the stabilized 1-states $\overline{\Lambda}_{\underline{z}} \circ\!\!-\!\!\bullet zx_0\bar{x}_1$, which appear as 0-outputs.

$$\underline{z} := \lambda_{\underline{z}}^{-1}(y,x) + \neg\overline{\lambda}_{\overline{\underline{z}}}^{-1}(y,x) = \underline{z} + \neg\overline{\underline{z}}$$

respectively

$$(\underline{z},\overline{\underline{z}}) := (\delta \wedge (\bar{z} \vee \lambda), \overline{\delta} \wedge (z \vee \lambda))$$

A. RS-Buffer

The RS-Buffer used and its circuit symbol are represented in figure 7 and 8 and

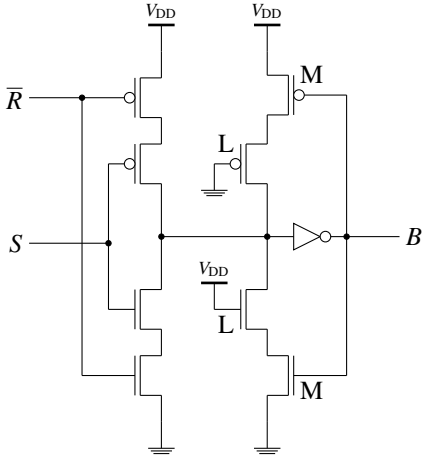

Figure 7: RS-Buffer (schematic)

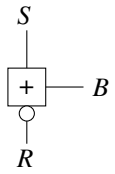

Figure 8: Circuit symbol of the used RS-Buffer

its logic table is specified in table 1. This RS-Buffer is used to guarantee handshaking and synchronization, especially with asynchronous feedbacked structures. Overlaid signals can lead to dangerous errors, meta stabilities and races. Therefore these dangers must be intercepted reliably. The recommended RS-Buffer is suitable for this task. On the transistor level it consists of a first stage (tri-state-combinationals) and a second stage (so called babysitter). The tri-state control structure provides, depending on the input, either a set signal. a reset signal or a high resistive closure to the babysitter. The babysitter itself consists of two complementary loops. If no signal is pending (high resistive closure) the babysitter saves the last state. The RS-Buffer is used to let every constructive superposition of signals pass, while the positive superposition triggers the setting signal and the negative superposition triggers the resetting signal or the last defined signal is held [5].

R	0	0	1	1
S	0	1	0	1
B	B	1	0	B

Table 1: Truth table of the RS-Buffer

The formula for the output B is therefore $B := \overline{R}(B \vee S) \vee S(B \vee \overline{R}) = \overline{R}S \vee \overline{R}B \vee SB$.

B. 1-dimensional Example

For $x = (x_1, x_0)$, $y = (y)$ and $z = (z)$ the transformation of the Mealy machine $(X, Y, Z, \delta, \lambda)$ with the state transformation function and output function

$$\delta(z,x) = zx_0\bar{x}_1 \vee x_1 \qquad \lambda(z,x) = zx_0x_1 \vee \bar{x}_0$$

in figure 9, figure 10 respectively will be executed.

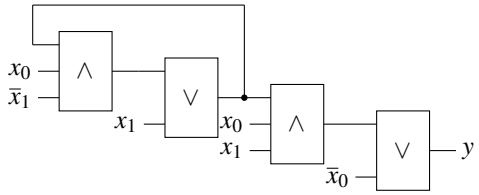

Figure 9: Mealy machine before the transformation

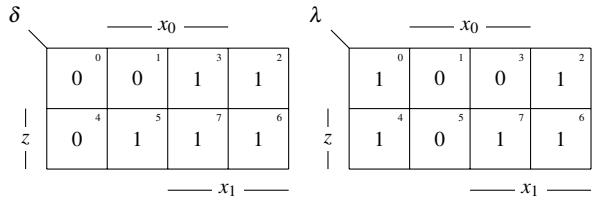

Figure 10: ... and the KV-diagrams

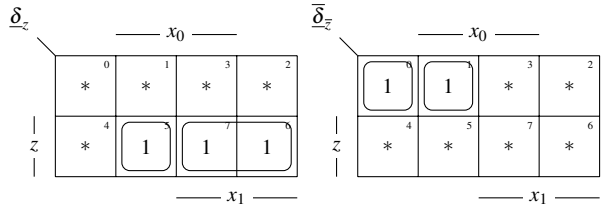

Figure 11: State transfer function $(\underline{\delta}_z, \overline{\underline{\delta}}_{\overline{z}})$

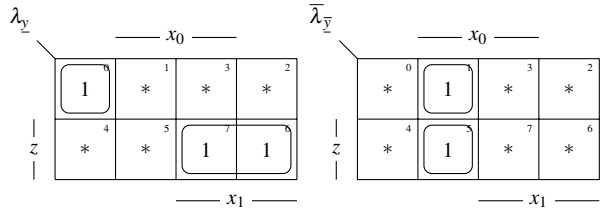

Figure 12: Output $(\underline{\lambda}_y, \overline{\underline{\lambda}}_{\overline{y}})$

The function δ will be partitioned in the stabilized 1-states $\underline{\delta}_z$ and the stabilized 0-states $\overline{\underline{\delta}}_{\overline{z}}$ in propositional logic in the following KV-diagrams, see figure 11. Same applies for the output function, represented in figure 12.
For the function λ^{-1} the following expression applies

$$\underline{z} := \overline{z}(z(x_0 \vee x_1) \vee \overline{z}x_1) + z(x_1) = x_1$$

$$\overline{\underline{z}} := z\overline{x}_0\overline{x}_1 + \overline{z}\overline{x}_0\overline{x}_1 = \overline{x}_0\overline{x}_1$$

with the illustration in the KV-diagram in figure 13.

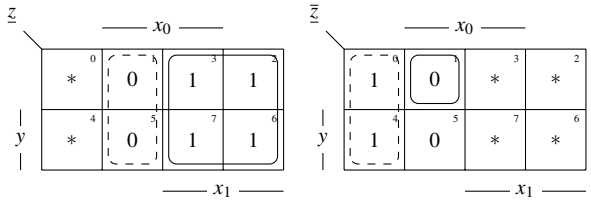

Figure 13: KV-diagrams of $\underline{z}$ and $\overline{\underline{z}}$

The partial truth table of the automaton is represented in table 2. The resulting automaton can be seen in figure 14. The example showed, that there is no need of feeding y back, because the interim feedback of the RS-Buffer either keeps the current state or changes of the state occur via the input signals. Therefore there is no actual need for a feedbacked y for generating λ^{-1}.
For a better understanding we will further introduce an example of a mealy to moore transformation of a mealy machine with a 2-dimensional state transfer function.

z	x_1	x_0	δ	$\underline{\delta}_z$	$\overline{\underline{\delta}}_{\overline{z}}$	λ	$\underline{\lambda}_y$	$\overline{\underline{\lambda}}_{\overline{y}}$	$\underline{z}$	$\overline{\underline{z}}$	z
0	0	0	0	*	1	1	1	*	0	1	0
0	0	1	0	*	1	0	*	1	*	0	z
0	1	0	1	*	*	1	*	*	1	*	1
0	1	1	1	*	*	0	*	*	1	*	1
1	0	0	0	*	*	1	*	*	*	1	0
1	0	1	1	1	*	0	*	1	0	0	z
1	1	0	1	1	*	1	1	*	1	*	1
1	1	1	1	1	*	1	1	*	1	*	1

Table 2: Partial truth table of the transformed 1-dimensional automaton

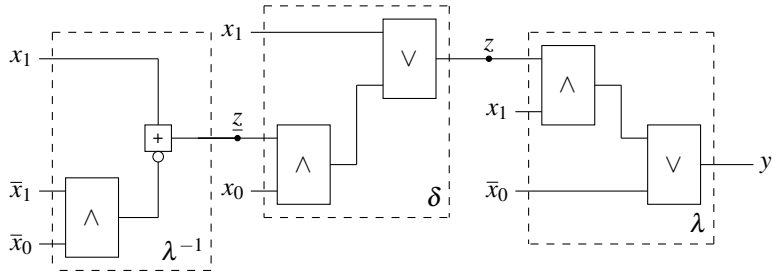

Figure 14: 1-dimensional automaton after the transformation, $z' = (z, x)$

C. 2-dimensional example

For $x=(x_1,x_0)$, $y=(y)$ and $z=(z_1,z_0)$ the transformation of the Mealy machine (X,Y,Z,δ,λ) with the state transformation functions and output function

$$\delta_0(z,x)=z_1\overline{x}_1\vee z_0x_1x_0 \qquad \delta_1(z,x)=z_1x_0\vee x_1x_0$$

$$\lambda(z,x)=z_1\overline{x}_1\overline{x}_0\vee z_1\overline{z}_0\overline{x}_1x_0$$

in figure 15, figure 16 respectively will be executed.

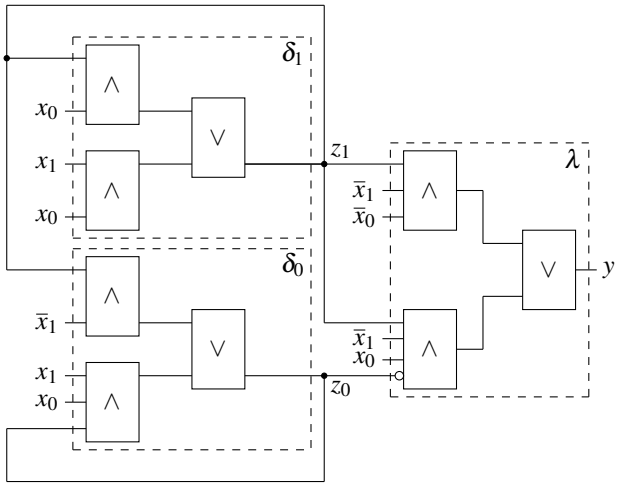

Figure 15: Mealy machine before the transformation

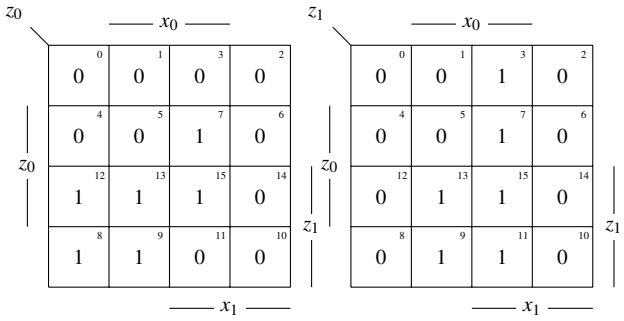

Figure 16: ... and the KV-diagrams

The compacted states $\underline{z_0}$ and $\underline{z_1}$ can now be calculated with z_0=$\delta_{z_0}(z_1,x)$ and z_1=$\delta_{z_1}(z_0,x)$ respectively. The compacted KV-diagram of z_0 is illustrated in figure 17.

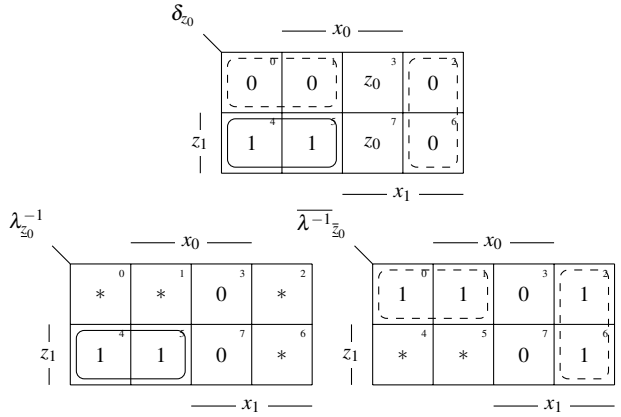

Figure 17: Compacted functions δ_{z_0} and λ^{-1} of δ_{z_0}

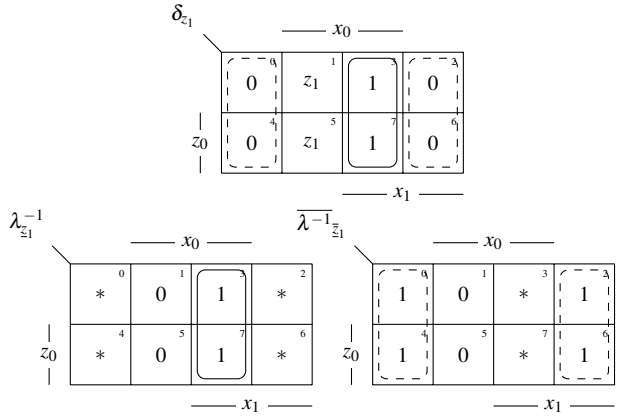

Figure 18: Compacted functions δ_{z_1} and λ^{-1} of δ_{z_1}

The function δ_{z_0} will now be partitioned in three parts. There is one part, where the 1-state of the state transfer function is independent from z_0 (marked by continuous lines in the KV-diagram), the second part is the 0-state, which is independent from z_0 (marked by dashed lines) and the configuration of x which is dependent on the last state z_0 where the interim feedback of the RS-Buffer will keep the old state (0-values in the KV-diagram), see figure 17. Same applies for δ_{z_1}, see 18. The reduced states $\underline{z_0}$, $\overline{\underline{z}}_0$, $\underline{z_1}$, $\overline{\underline{z}}_1$ can now be calculated:

$$\lambda_{\underline{z}_0}^{-1}(z_1,x)=z_1\overline{x}_1\vee\neg(x_1x_0) \qquad \overline{\lambda^{-1}}_{\overline{\underline{z}}_0}(z_1,x)=\overline{z}_1\overline{x}_1\vee x_1\overline{x}_0\vee\neg(x_1x_0)$$

$$\underline{z_0}=\lambda_{\underline{z}_0}^{-1}(z_1,x)+\neg(\overline{\lambda^{-1}}_{\overline{\underline{z}}_0}(z_1,x))$$

$$\lambda_{\underline{z}_1}^{-1}(z_0,x)=x_1x_0\vee\neg(\overline{x}_1x_0) \qquad \overline{\lambda^{-1}}_{\overline{\underline{z}}_1}(z_0,x)=\overline{x}_0\vee\neg(\overline{x_1}x_0)$$

$$\underline{z_1}=\lambda_{\underline{z}_1}^{-1}(z_0,x)+\neg(\overline{\lambda^{-1}}_{\overline{\underline{z}}_1}(z_0,x))$$

The partial truth table of the automation is represented in table 3. The resulting automaton can be seen in figure 19.

z_1	z_0	x_1	x_0	δ_1	δ_0	λ	$\underline{z}_1$	$\overline{\underline{z}}_1$	$\underline{z}_0$	$\overline{\underline{z}}_0$	z_1	z_0
0	0	0	0	0	0	0	0	1	0	1	0	0
0	0	0	1	0	0	0	*	*	0	1	z_1	0
0	0	1	0	0	0	0	0	1	0	1	0	0
0	0	1	1	1	0	0	1	0	*	*	1	z_0
0	1	0	0	0	0	0	0	1	0	1	0	0
0	1	0	1	0	0	0	*	*	0	1	z_1	0
0	1	1	0	0	0	0	0	1	0	1	0	0
0	1	1	1	1	1	0	1	0	*	*	1	z_0
1	0	0	0	0	1	1	0	1	1	0	0	1
1	0	0	1	1	1	1	*	*	1	0	z_1	1
1	0	1	0	0	0	0	0	1	0	1	0	0
1	0	1	1	1	0	0	1	0	*	*	1	z_0
1	1	0	0	0	1	1	0	1	1	0	0	1
1	1	0	1	1	1	0	*	*	1	0	z_1	1
1	1	1	0	0	0	0	0	1	0	1	0	0
1	1	1	1	1	1	0	1	0	*	*	1	z_0

Table 3: Partial truth table of the transformed automaton

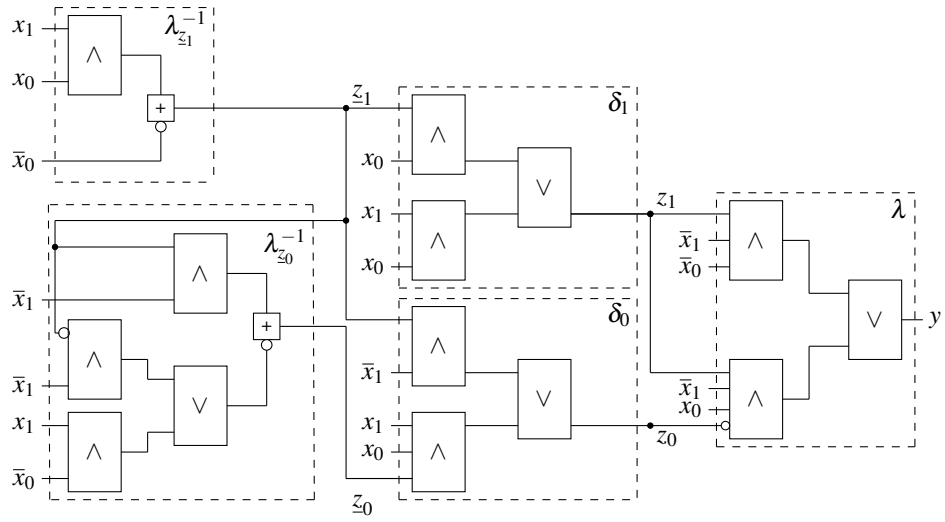

Figure 19: Automaton after the transformation, z'=(z,x)

IV. Conclusion and future work

The Moore transformation of a function stable asynchronous Mealy machine in dual-rail logic under the use of the RS-buffer was exemplarily shown in this paper. The feedback of the state was moved over the output function λ of the Mealy machine and correctly created at the input of δ via the function λ^{-1}. The part of δ that is stabilized in z can be held in the interim feedback of λ^{-1} when needed and the transitive part, which is only dependent on the input variables x, will be generated by the combinationals. The simple 2-dimensional example was consciously chosen to show that this method is also applicable for more dimensions. It shows a structure that is asynchronously feedbacked and seems to end within a finite amount of steps in a stable state. There is need for further investigation of more complex and more-dimensional Mealy machines to define a general n-dimensional procedure.

References

[1] G. Uygur, S. M. Sattler: Pin-Type Based VLSI Partitioning. AmE 2014, Dortmund.

[2] M. Ipek: Eine Testfallspezifikation für das funktionsorientierte Testen von reaktiven eingebetteten Systemen im Automobilen Bereich. [Ph.D. Dissertation]. Universität Kaiserslautern, Mai 2011.

[3] U. Hebisch: Formale Sprache und Automatentheorie WS 13/14 page 45-50. `http://www.mathe.tu-freiberg.de/ hebisch/skripte/auttheorie/formsprach.pdf`

[4] G. Uygur, S. M. Sattler: A Real-World Model of Partially Defined Logic. 12th International Workshop on Boolean Problems 2016, Freiberg.

[5] G. Uygur, S. M. Sattler: A New Approach for Modelling Inconsistencies in Digital-Assisted Analog Design. Journal of Electronic Testing 2016, Page 498-503.