Mealy-to-Moore transformation in safety-critical systems

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

1 Introduction

AUTOMOTIVE is a safety critical application with high need of functional safety, under extreme operating respectively environmental conditions and parameter variations [1,2]. This growing need and also information and communication technologies for automotive lead to new applications, which can be integrated into the system by innovative ideas and solutions under the use of microelectronics. The used micro electronic (electronic control unit, ECU) can take on complex tasks, such as steering and signal processing. An ECU can be formally abstracted to a Mealy machine [3] with the state transfer function δ and the output function λ, see figure 1. 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': $\mu : (z') \mapsto \mu(z')$.
By comparing the two machines 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.
In the paper the transofmation of a Mealy machine into a Moore machine is presented. In this transformation the feedback of the state of the transformed Moore machine is at the output. In order to set the correct state at the input of the state transfer function of the Mealy, a function will be integrated, which will generate the initial state from the output signal of the Mealy machine. For function stable asynchronous machines this can be done with dual-rail logic and the RS-Buffer [4]. With the method presented in the 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 and complex circuits for safety relevant applications can be realized.

2 Theory

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

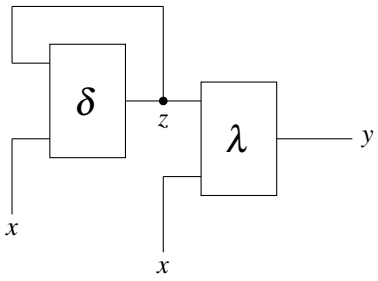

Figure 1: Fully asynchronous Mealy machine

Figure 1 shows a fully asynchronous Mealy machine. The branches entering a node in the graph should end reflexive, so that only transient states are allowed which are conscious and triggered from outside. To view 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, which generates the reduced state $\underline{z}$ from the feedback y. This equivalent transformation is outlined in figure 2.

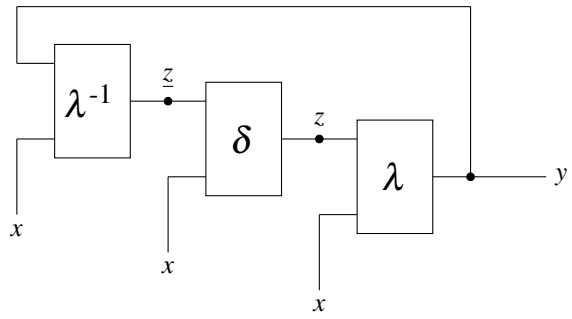

Figure 2: Equivalent transformation

The following applies:

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

2.1 Dual-rail logic

For the implementation in dual-rail logic, 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.

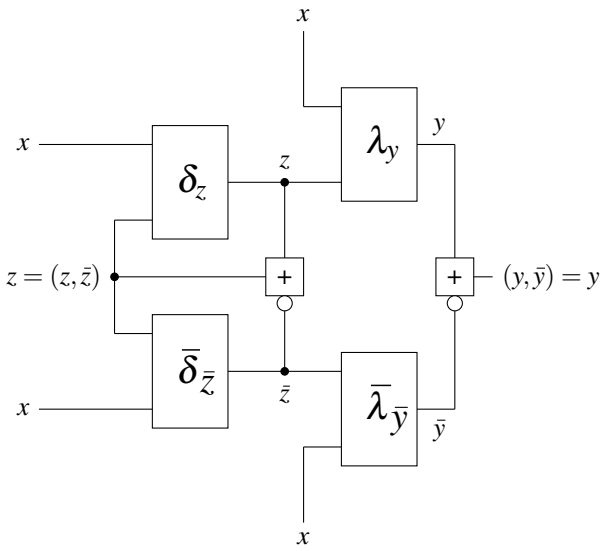

Figure 3: Mealy realized in dual-rail logic

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

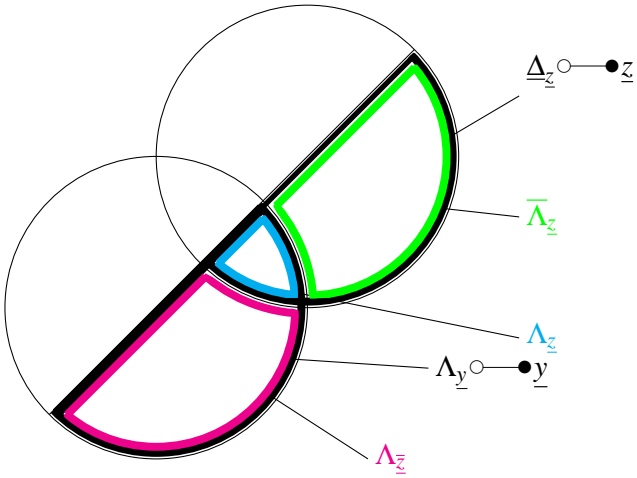

Figure 4: Venn diagram of 1-states in 1-outputs

The stabilized 1-states, $\underline{\Delta}_z \circ\!\!-\!\!\bullet \underline{z}$, 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 $\Lambda_{\underline{\bar{z}}}$. The 1-partition is composed of:

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

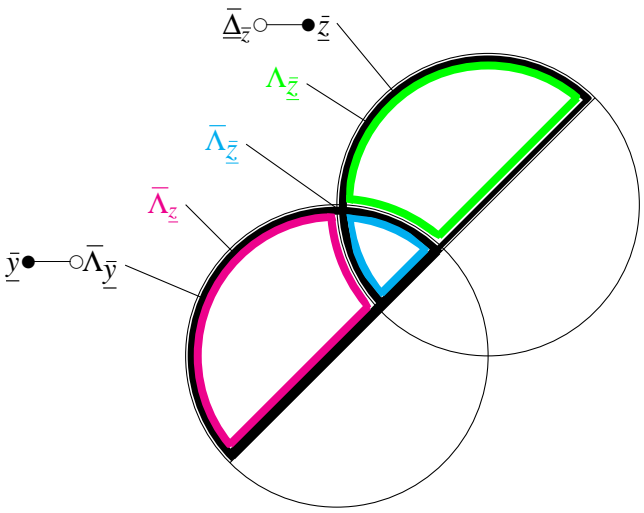

Figure 5: Venn diagram of 0-states in 0-outputs

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

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

3 Use-Case

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

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

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.

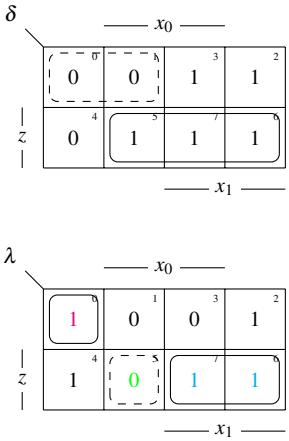

Figure 6: State transfer function and output

And the stabilized 0-states of z, $\overline{\underline{\Delta}}_{\underline{z}} \circ\!\!-\!\!\bullet\, \bar{z}\bar{x}_1$, the stabilized 0-states $\overline{\Lambda}_{\underline{z}} \circ\!\!-\!\!\bullet\, \bar{z}x_0\bar{x}_1$, 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

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

3.1 RS-Buffer

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

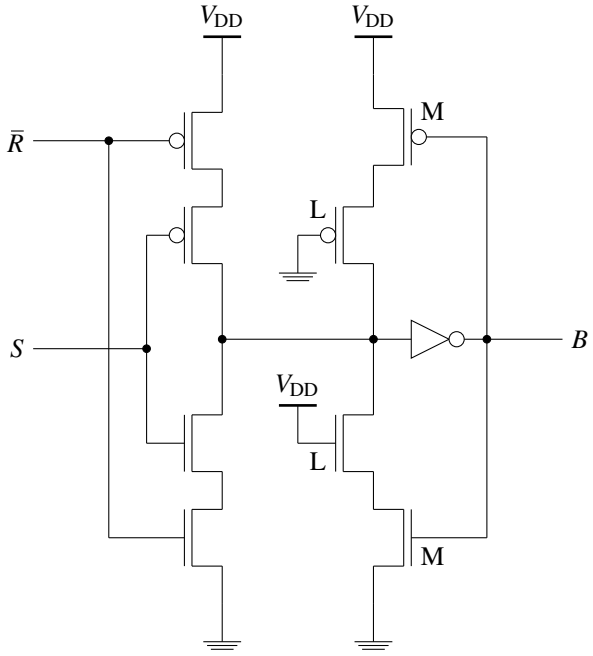

Figure 7: RS-Buffer (schematic)

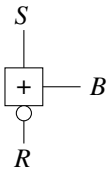

Figure 8: Circuit symbol of the used RS-Buffer

its logic table is specified in table 1. This RS-Buffer is used to guarantee handshaking and synchronization, especially with asynchronous feedback 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 steering structure provides, depending on the input, either a set or 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, the positive superposition trigger the setting signal and the negative superposition triggers the resetting signal, or the last defined signal is on hold [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 $B := \overline{R}(B \vee S) \vee S(B \vee \overline{R}) = \overline{R}S \vee \overline{R}B \vee SB$.

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

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

in figure 9, figure 10 respectively will be executed.

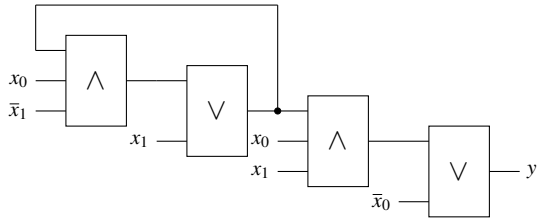

Figure 9: Mealy machine before the transformation

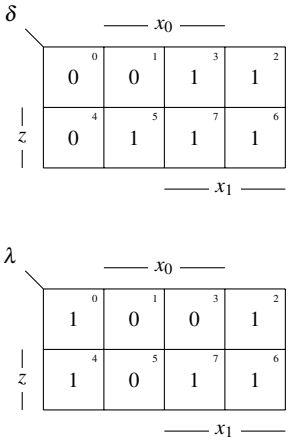

Figure 10: ... and the KV-diagrams

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

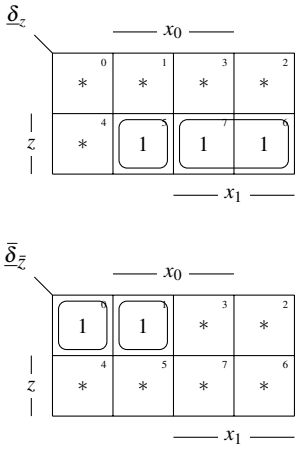

Figure 11: State transfer function $(\underline{\delta}_z, \underline{\overline{\delta}}_{\overline{z}})$

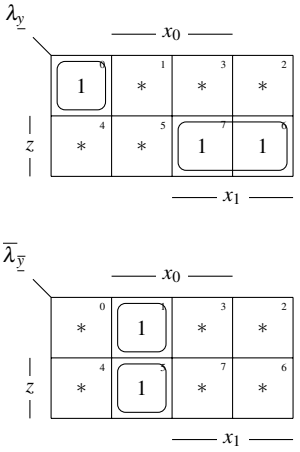

Figure 12: Output $(\lambda_{\underline{y}}, \overline{\lambda}_{\underline{\overline{y}}})$

For the function λ^{-1} the following expression applies

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

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

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

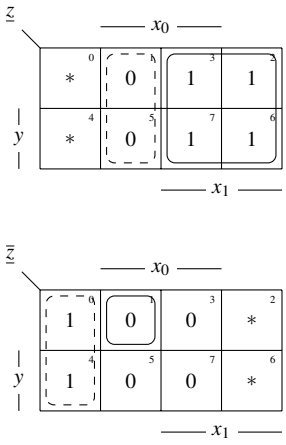

Figure 13: KV-diagrams of $\underline{z}$ and $\underline{\overline{z}}$

The partial truth table of the automation is represented in table 2. The resulting automaton can be seen in figure 14.

z	x_1	x_0	δ	$\underline{\delta}_z$	$\overline{\underline{\delta}}_{\overline{z}}$	λ	$\lambda_{\underline{y}}$	$\overline{\lambda}_{\overline{\underline{y}}}$	$\underline{z}$	$\overline{\underline{z}}$	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 automaton

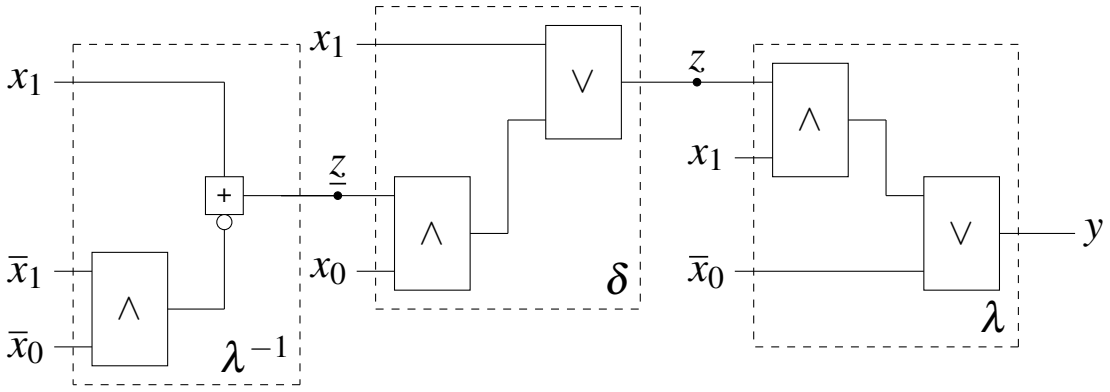

Figure 14: Automaton after the transformation, $z' = (z, x)$

4 Conclusion

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 created correctly at the input δ via the function λ^{-1}. The part of delta 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.

5 Literature

[1] Gürkan Uygur, Sebastian 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] Udo Hebisch: Formale Sprache und Automatentheorie WS 13/14 page 45-50. `http://www.mathe.tu-freiberg.de/ hebisch/skripte/auttheorie/formsprach.pdf`

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

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