1.6. A Real-World Model of Partially Defined Logic

Gürkan Uygur Sebastian M. Sattler

1.6.1. Real-World Asynchronous Feedback

Digital circuit systems can be abstracted as a system which defines a map of input values to output values [390]. In CMOS technology, the values can be formalized, e.g., by specifying a voltage upper limit to digitize (respectively abstract) each upper voltage level as a logical high, abbreviated with the propositional value 1 and by specifying a voltage lower limit to digitize each lower voltage level as a logical low, abbreviated with the propositional value 0. Within this abstraction, Boolean Algebra is the state of the art modeling approach for formalizing combinational circuit systems. Thus in general, the input values $x \in \mathbf{x}$ can be encoded by the binary (2-valued) input vector $\mathbf{x} = (x_{n-1}, \ldots, x_i, \ldots, x_1, x_0)$ of length n, $x_i \in \{1, 0\}$, and the output values $y \in \mathbf{y}$ can be encoded by the binary output vector $\mathbf{y} = (y_{m-1}, \ldots, y_j, \ldots, y_1, y_0)$ of length m, $y_j \in \{1, 0\}$. Then a combinational function F can be termed as

$$F : \mathbb{B}^n \to \mathbb{B}^m \,, \qquad F : \mathbf{x} \mapsto F(\mathbf{x}) \,, \qquad \mathbf{y} = F(\mathbf{x}) \,, \tag{1.28}$$

saying that F maps source $\mathbf{x}$ to target $\mathbf{y}$ by uniquely assigning $\mathbf{x} \in \mathbb{B}^n$ to $\mathbf{y} \in \mathbb{B}^m$. Further, function $F(\mathbf{x})$ is said to be totally defined (in short total) if and only if $F(\mathbf{x})$ is defined for all $\mathbf{x} \in \mathbb{B}^n$ otherwise $F(\mathbf{x})$ is said to be partially defined (in short partial).

In the state of the art literature, a most common digital circuit system can be modeled as a finite state machine having Mealy architecture [390]. A Mealy machine is composed of two combinational structures, one for implementing the state transition function $\delta : (\mathbf{x}, \mathbf{z}) \mapsto \delta(\mathbf{x}, \mathbf{z})$ with state value $\mathbf{z} \in \mathbb{B}^{|\mathbf{z}|}$, another for implementing the output function $\lambda : (\mathbf{x}, \mathbf{z}) \mapsto \mathbf{y}$. In case the output function μ only depends on $\mathbf{z}$, $\mu : \mathbf{z} \mapsto \mathbf{y}$, the model describes a Moore machine. And, if trivially feeding $\mathbf{z}$, $\mathbf{1}(\mathbf{z}) = \mathbf{y}$ with $\mu : \mathbf{z} \mapsto \mathbf{1}(\mathbf{z})$, then the model describes a Medwedew machine. The sequential logic system becomes

realized by asynchronously feeding the state value $\mathbf{z}$ back to the input $\mathbf{z}$; this successfully works if δ is functionally stabilized, i.e., the following predicate logical expression holds: $\mathbf{z} = \delta(\mathbf{x}, \mathbf{z})$. At this point, it should be noted, that using the different symbols ${}^{o}\mathbf{z}$ and ${}^{n}\mathbf{z}$ for old and new state values can sometimes be helpful for mathematically modeling sequential logic as a combinational function, but in reality, the fully asynchronously feedback sequential structure determines the state value $\mathbf{z}$ by superimposing signals on the feedback.

In a case where δ is not functionally stabilized, superimposed signals can lead to hazardous glitches, metastability and races, respectively. Within the digital abstraction, such analog feedback effects are not determined, staying undefined associated with the symbol $*$. In terms of digital abstraction, for state transition function δ, $*$ means that the next state value $\delta(\mathbf{x}, \mathbf{z})$ is not defined, $\delta(\mathbf{x}, \mathbf{z}) = *$. Hence in general, a real-world asynchronous feedback structure realizing a discrete event based sequential logic module clones a partially defined state transition function δ; so in general, a digital circuit system realizes a partially defined function.

1.6.2. Related Topics

Safety critical systems increasingly profit from integrated circuit technology, which bundle analog-to-digital functionality with high efficiency and quality at low cost. From the view of safety and application, determined and also efficient components are needed. Analog is efficient but ambiguous, contrastingly digital is determined by (clocked) synchronization but is thus inefficient. Hence, there is a requirement to couple the advantages of becoming deterministic as well as efficient [121]. At this point, asynchronous circuit systems arise as a dedicated candidate for safety-critical applications. There is a renewed and increasing interest in asynchronous design; emphasis is shifting from asynchronous-in-the-small to asynchronous Very-Large-Scale Integration (VLSI). Asynchronous VLSI is progressing from a fashionable academic research topic to a viable solution for a number of VLSI design challenges [371]. The inherent reason is that the asynchronous feedback is provided with lots of potential benefits such as improved system performance, modular design, low power consump-

tion and reduced electromagnetic emissions [135, 299]. Further essential characteristics are that the asynchronous circuit systems show no problems with clock skew and related subtle issues, and are fundamentally more tolerant of voltage, temperature and manufacturing process variations [174, 226, 278]. It is remarkable that the international technology roadmap for semiconductors (report on design [155]) predicts that up to 40% of the designs will be driven by *handshake clocking* (i.e., asynchronous) by 2020 [174].

Due to the increasing complexity of structure and functionality the test of the real-world structure also becomes more and more complex, hence the formal assignment between the (virtual) functionality and the (real-world) structure becomes more and more critical. This constellation leads to an ever-increasing challenge particularly regarding safety critical circuits and systems. Analog destructive signal superimposing effects such as metastability and race, respectively, most often result in a delay reaction at the output, which in a synchronous circuit potentially violates the timing assumption [239]. This also means that due to the increasing impact of asynchronous feedback effects, the modeling inconsistencies—in state of the art so-called "don't cares"—become more and more relevant with regard to the modeled (virtual) function and the real-world structure. These inconsistencies play an important role particularly on (elementary) structures storing information, e.g., latches up to complex automata. The focus of this section is to establish a formal methodology to warrant the match between the (partially) specified function and the real world asynchronous feedback structure.

1.6.3. Use Case: Low-Active RS-Latch

Let us look at an elementary non-trivial asynchronous fed-backed circuit structure, the low-active RS-latch shown in Figure 1.20 (a). It can be interpreted on different abstraction levels. In the *logical* view, each combination $(\mathbf{x}, \mathbf{z})$ determines the next state $\delta(\mathbf{x}, \mathbf{z})$. The associated $\mathbf{z}$-equations are given as totally defined combinational functions. In the *digital* view, metastable state values, which can trigger races, are not accepted as the *digital state*. For a more detailed characterization, let us consider the circuit structure of the low-active RS-latch as a

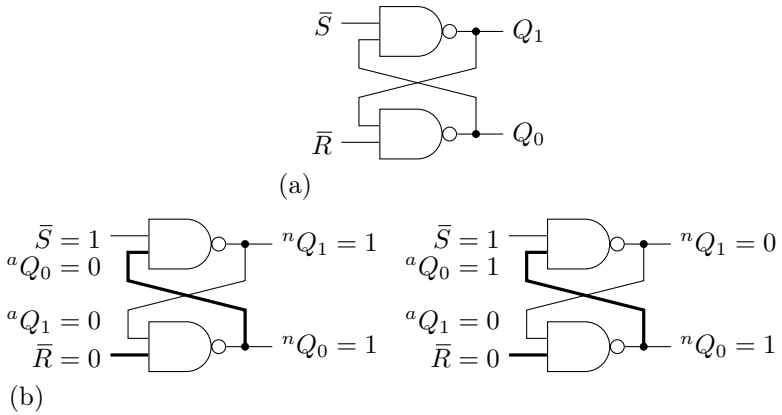


Figure 1.20. Low-active RS-latch and metastability: (a) circuit structure, (b) different scenarios resulting in metastability.

Device Under Test (DUT): the following test pattern of simultaneous signal transitions (sequence) $[0 \to 1 \to 1 \to \cdots, 0 \to 1 \to 1 \to \cdots]$ on *logical* input variables $(\overline{S}, \overline{R})$ triggers a logical oscillation (sequence) $[1 \to 0 \to 1 \to 0 \to \cdots, 1 \to 0 \to 1 \to 0 \to \cdots]$ on *logical* state variables (Q_1, Q_0); contrastingly, the same test pattern on *digital* real-world input pins $(\overline{S}, \overline{R})$ triggers a race condition resulting in non-determined digital signals on real-world state pins (Q_1, Q_0), and on *analog* real-world state pins (Q_1, Q_0), it results in destructive signal superimposing, which then can be observed as concurring discharging and precharging voltage signal curves (see Figure 1.21).

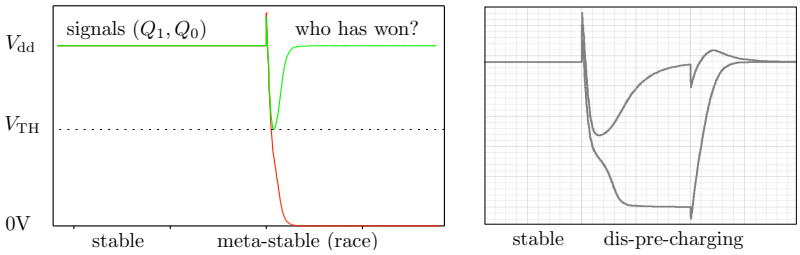


Figure 1.21. Race condition resulting in non-determined (digital) signals respectively (analog) dis-pre-charging voltage signal on real-world state pins (Q_1, Q_0).

Table 1.16. Logical vs. digital switching table of the low-active RS-latch

combinational				logical		digital		
${}^aQ_1$	${}^aQ_0$	S	R	${}^nQ_1$	${}^nQ_0$	${}^nQ_1$	${}^nQ_0$	comment for digital
0	0	0	0	1	1	$*$	$*$	metastable
0	0	0	1	1	1	$*$	1	metastable
0	0	1	0	1	1	1	$*$	metastable
0	0	1	1	1	1	1	1	stable state
0	1	0	0	0	1	0	1	stable state (hold)
0	1	0	1	0	1	0	1	stable state(hold)
0	1	1	0	1	1	1	1	transient state
0	1	1	1	1	1	1	1	stable state
1	0	0	0	1	0	1	0	stable state (hold)
1	0	0	1	1	1	1	1	transient state
1	0	1	0	1	0	1	0	stable state (hold)
1	0	1	1	1	1	1	1	stable state
1	1	0	0	0	0	$*$	$*$	metastable
1	1	0	1	0	1	0	1	stable state (reset)
1	1	1	0	1	0	1	0	stable state (set)
1	1	1	1	1	1	1	1	stable state

Similarly, the initial condition $(Q_1, Q_0, \bar{S}, \bar{R}) = [0011]$ also triggers a race condition. $(Q_1, Q_0, \bar{S}, \bar{R}) = [0010]$ triggers a metastability and makes Q_1 undefined. Figure 1.20 (b) shows different scenarios resulting in metastability. Similarly, $(Q_1, Q_0, \bar{S}, \bar{R}) = [0001]$ triggers a metastability and makes Q_0 undefined. Hence, it results in the non-reduced structure preserving digital switching table, see Table 1.16.

All combinational inputs $({}^aQ_1, {}^aQ_0, S, R)$ resulting in digitally undefined (symbol $*$) can be encoded and termed as a Ternary Vector List (TVL) [46, 342, 390] and be further compressed (resolved), see (1.29); thus $\bar{*}$ means all combinations result in a defined state transition, see (1.30). Now, all reachable states [00], [01], [10] and [11] formally assigned to $({}^nQ_1, {}^nQ_0)$ can be calculated by differentiating[3] all $*$ from the logical model, which then result in digital state

[3] One should notice that in TVL calculus differentiating $*$ is equivalent to intersecting with $\bar{*}$.

equations, see Equations (1.31)−(1.34). Hence, after appending all state equations to one TVL with header $({}^{a}Q_1, {}^{a}Q_0, S, R, {}^{n}Q_1, {}^{n}Q_0)$, it results in a TVL encoded automaton graph AG respectively and all defined state transitions of associated digital automaton, see Equation (1.35).

$$* = \begin{matrix} {}^{a}Q_1 & {}^{a}Q_0 & S & R \end{matrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} = \begin{matrix} {}^{a}Q_1 & {}^{a}Q_0 & S & R \end{matrix} \begin{bmatrix} 0 & 0 & 0 & - \\ 0 & 0 & - & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \tag{1.29}$$

$$\bar{*} = \begin{matrix} {}^{a}Q_1 & {}^{a}Q_0 & S & R \end{matrix} \begin{bmatrix} 0 & 1 & - & - \\ 1 & 0 & - & - \\ 1 & - & 1 & - \\ - & - & 1 & 1 \\ 1 & - & - & 1 \end{bmatrix} \tag{1.30}$$

$$\begin{matrix} {}^{n}Q_1 & {}^{n}Q_0 \end{matrix} \begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{matrix} {}^{a}Q_1 & {}^{a}Q_0 & S & R \end{matrix} \begin{bmatrix} 1 & 0 & - & 1 \\ 0 & 1 & 1 & - \\ - & - & 1 & 1 \end{bmatrix} \tag{1.31}$$

$$\begin{bmatrix} 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & - & 0 \\ 1 & - & 1 & 0 \end{bmatrix} \tag{1.32}$$

$$\begin{bmatrix} 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & - \\ - & 1 & 0 & 1 \end{bmatrix} \tag{1.33}$$

$$\begin{bmatrix} 0 & 0 \end{bmatrix} = 0 \tag{1.34}$$

$$AG = \begin{bmatrix} 010- \\ -101 \end{bmatrix} [01] \vee \begin{bmatrix} 10-0 \\ 1-10 \end{bmatrix} [10] \vee \begin{bmatrix} 10-1 \\ 011- \\ --11 \end{bmatrix} [11]$$

$$AG = \begin{array}{c} \begin{matrix} {}^aQ_1 & {}^aQ_0 & S & R & {}^nQ_1 & {}^nQ_0 \end{matrix} \\ \left[\begin{array}{cccccc} 0 & 1 & 0 & - & 0 & 1 \\ - & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & - & 0 & 1 & 0 \\ 1 & - & 1 & 0 & 1 & 0 \\ 1 & 0 & - & 1 & 1 & 1 \\ 0 & 1 & 1 & - & 1 & 1 \\ - & - & 1 & 1 & 1 & 1 \end{array}\right] \end{array} \tag{1.35}$$

1.6.4. Functionally Stabilized Dual-Rail Implementation

The case above shows that the digital state transition function

$$\delta : (\mathbf{x}, \mathbf{z}) \mapsto \delta(\mathbf{x}, \mathbf{z})$$

with $\mathbf{x} = (S, R)$ and $\mathbf{z} = (Q_1, Q_0)$ is partially defined. For creating the associated Signal Flow Plan (SFP) in gate logic view, we use the dual-rail approach, which means that each logical positive assignment gets termed by the associated positive literal (the positive rail) and each logical negative assignment gets termed by the associated negative literal (the positively encoded negative rail). Hence, we are speaking unary (1-valued modeling) for each rail $Q_1 \in \{1\}$, $\bar{Q}_1 \in \{1\}$, $Q_0 \in \{1\}$, $\bar{Q}_0 \in \{1\}$, see Equations (1.36) and (1.37):

$$\left(Q_1, \bar{Q}_1\right) = \left(\begin{bmatrix} 1\,0 - - \\ - - 1\,1 \\ -\,1\,1 - \end{bmatrix}, \begin{bmatrix} 0\,1\,0 - \\ -1\,0\,1 \end{bmatrix} \right) , \tag{1.36}$$

$$\left(Q_0, \bar{Q}_0\right) = \left(\begin{bmatrix} 0\,1 - - \\ - - 1\,1 \\ 1 - - 1 \end{bmatrix}, \begin{bmatrix} 1\,0 - 0 \\ 1 - 1\,0 \end{bmatrix} \right) . \tag{1.37}$$

Now, the associated SFP in gate logic can be deduced from the TVL of each rail, which encodes the gate logic in Disjunctive Normal Form (DNF): one line (ternary vector) of a TVL encodes the conjunction of literals, whereas the TVL encodes the disjunction of the conjunctions.

Functionally Stabilized Parallel Composition. Each dual-rail needs to be parallelly composed (+) to its corresponding single-rail. For this we will use our programmable JK-/RS-buffer. The behavior of this buffer is shown in Figure 1.22:

(a)

CS	J,S	K,R	B	$*$	comment
0	0	0	B	1	store
0	0	1	0	0	reset
0	1	0	1	0	set
0	1	1	B	1	store
1	0	0	B	1	store
1	0	1	0	0	reset
1	1	0	1	0	set
1	1	1	$\overline{B}$	1	toggle

(b)

CS	${}^aB$	${}^nB$			
0	0	0	0	0	1
0	1	1	0	1	1
1	1	1	0	0	1
1	0	0	0	1	1
		0	1	1	0 K,R
		0	0	1	1 J,S

Figure 1.22. Behavior of the programmable JK-/RS-Buffer: (a) function table, (b) Karnaugh map.

Figure 1.23 depicts the associated structure of the circuit together with functions on selected points:

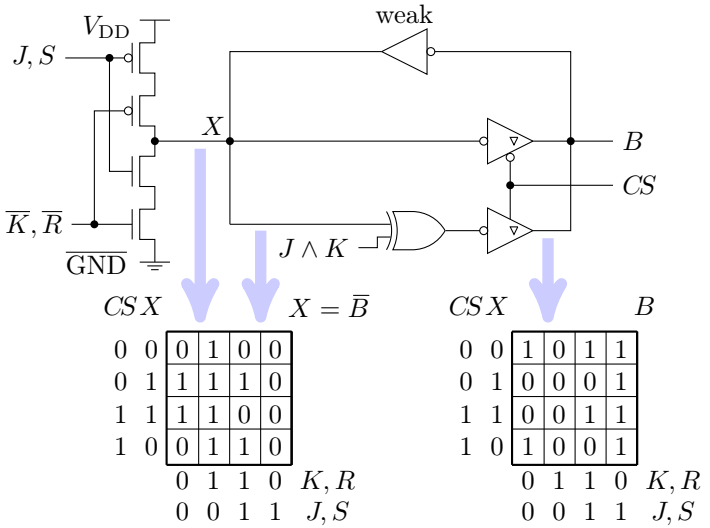


Figure 1.23. Structure of the programmable JK-/RS-buffer.

For implementation of the fully asynchronous parallel dual-rail composition, we program the RS-buffer by assigning the chip select pin CS with digital 0, $CS = [0]$, which deactivates the JK-rail and activates the RS-rail, see Figure 1.23.

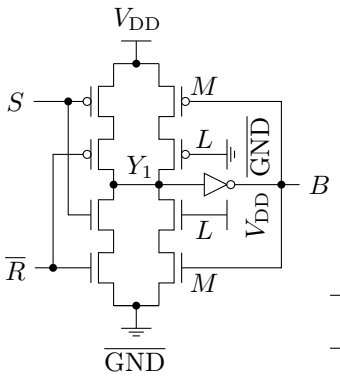
To implement the SFP to be functionally stabilized, we deploy the RS-buffer composed of a tri-state front circuit to catch dual-rail signals and a fully asynchronously feedback babysitter to safely set respectively re-set the fed state signal; in the case of $*$ input, the front circuit gets deactivated (push and pull transistors are non-conductive resulting in high impedance) such that the babysitter safely holds the last defined state signal. Figure 1.24 (a) shows the function table, and the derived Karnaugh map, as well as the algebraic expression for hazard-free implementation of this RS-buffer in gate logic. Figure 1.24 (b) depicts the transistor implementation of the asynchronous feedback RS-Buffer [368]; the minimum transistors are labeled with M (minimum), and the weak transistors are labeled with L (long). Figure 1.24 (c) shows the switching symbol used to refer to the functionally stabilized parallel RS-buffer.

S	R	B	$*$	comment
0	0	B	1	store
0	1	0	0	reset
1	0	1	0	set
1	1	B	1	store

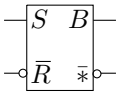
	${}^aB$			${}^nB$
0	0	0	0	1
1	1	0	1	1
	0	1	1	0 R
	0	0	1	1 S

$${}^nB = S\,\overline{R} \vee S\,{}^aB \vee \overline{R}\,{}^aB$$

(a)



(b)



(c)

Figure 1.24. RS-Buffer: (a) function, (b) circuit structure, (c) switching symbol.

The RS-buffer in Figure 1.25 is additionally provided with the $\bar{*}$ output, which realizes the test output for monitoring any violation of the specified operation.

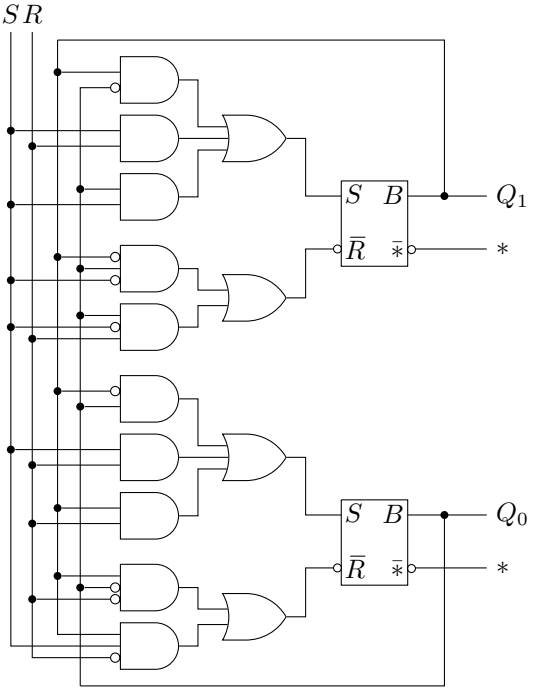


Figure 1.25. Block diagram of the parallel composed partial low-active RS-latch.

1.6.5. Results

The problem definition is that asynchronously fed-backed circuitry instantaneously underlies analogue principles. Therefore interfering signals can lead to hazardous glitches, metastability as well as races. In digital abstraction, such analog effects are undefined signals. Hence, generally speaking, a digital circuit system represents a partially defined function. This is contrary in the state of the art modeling where people mostly deploy a formalism and optimization approach that is based on a total and determined function, hard to derive from a given structure. The inconsistency between a formally derived (virtual) function and its underlying real-world structure is obvious. At this point the inherent challenging problem definition arises, that—

especially for the Quality Assurance (QA) of safety critical circuits and systems—not only is any solution itself sufficient but also that a reliably reproducible traceability of each step that results in the correct solution is needed in order to warrant a structure that preserves the axiomatic solution; plainly speaking, the correct solution depends on the correct construction of the solution.

As an alternative solution to the problem definition above, we offer our approach for real-world modeling of partially defined logic. Instead of using don't cares for modeling undefined inputs, we *implement partiality* by deploying a standardized sequential circuit structure (Moore). It doesn't transmit the formally undefined inputs by *generating* a tristate behavior for incoming undefined signals. The analog effects triggered at this structural weakness can now—within digital abstraction—be interpreted as metastability and race, respectively. They are undefined values in digital view. For the unique encoding of partial logic we deployed the positive logic on positive as well as negative signals. These dual rails in addition with the RS-buffer—provided with additional output for monitoring any violation of a specified operation—warrant a function stable implementation of a partially defined logic.