A Method to Design Compact Dual-rail Asynchronous Primitives

Alin Razafindraibe, Michel Robert, Marc Renaudin, Philippe Maurine

A Method to Design Compact DUAL-RAIL Asynchronous Primitives

Alin Razafindraibe[1], Michel Robert[1], Marc Renaudin[2], and Philippe Maurine[1]

[1] LIRMM, UMR CNRS/Université de Montpellier II,
(C5506), 161 rue Ada, 34392 Montpellier, France
{razafind,robert,pmaurine}@lirmm.fr
[2] TIMA Laboratory 46, avenue Félix Viallet, 38031, France
marc.renaudin@imag.fr

Abstract. This paper aims at introducing a method to quickly design compact dual-rail asynchronous primitives. If the proposed cells are dedicated to the design of dual-rail asynchronous circuits, it is also possible to use such primitives to design dual-rail synchronous circuits. The method detailed herein has been applied to develop the schematics of various basic primitives. The performances of the 130nm obtained cells have been simulated and compared with more traditional implementations.

I Introduction

If asynchronous circuits can outperform synchronous ICs in many application domains such as security, the design of integrated circuits still remains essentially limited to the realization of synchronous chips. One reason can explain this fact: no CAD suite has been proposed by the EDA industry to provide a useful and tractable design framework. However, some academic tools have been or are under development [1], [2], [3], [6].

Among them TAST [3] is dedicated to the design of micropipeline (μP) and Quasi Delay Insensitive (**QDI**) circuits. Its main characteristic is to target a standard cell approach. Unfortunately, it is uncommon to find in typical libraries (dedicated to synchronous circuit design) basic asynchronous primitives such as Rendezvous cells also called C-elements. Consequently, the designer of dual-rail asynchronous IC, adopting a standard cell approach, must implement the required boolean functions with AO222 or Majority gates [1], [4], [5]. It results in sub optimal physical implementations. Within this context, we developed a method allowing the quick design of the main combinatorial functionalities in dual-rail CMOS style.

Our goal is here to introduce the proposed design method and to compare the performances of the resulting cells with those of more traditional implementation styles. The remainder of the paper is organized as follows. Section II is dedicated to the introduction of the method itself. Before concluding, section III is devoted to the comparisons of performances (propagation delays, power consumption and realization cost) for the various boolean functions implemented with our design technique and with more usual implementation styles.

V. Paliouras, J. Vounckx, and D. Verkest (Eds.): PATMOS 2005, LNCS 3728, pp. 571–580, 2005.

II Dual-Rail Cell Design

In this paragraph, we introduce the dual-rail cell design technique which can be either used to design new functionalities or to quickly translate in dual-rail the content of any single rail library.

II.a Targeted Cell Topology

Before to detail the design technique, let us remind the main specificities of the primitives required to design dual-rail logic blocks exchanging data one with another according to a four phase handshake protocol, and to the dual-rail encoding given in Fig.1.

For such circuits, the data transfer through a channel starts by the emission of a request signal encoded into the data, and finishes by the emission of an acknowledge signal. During this time interval, which is a priori unknown, the incoming data must be hold in order to guarantee the quasi-delay-insensitivity property. This implies the intensive use of logical gate including a state holding element usually latch or feedback loops.

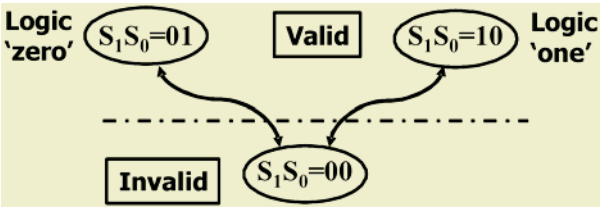

Fig. 1. Dual-rail encoding

As we target a CMOS implementation, it results from the preceding consideration that any dual-rail primitive may be considered as the juxtaposition of two complex gates, each one driving a latch as illustrated in Fig.2.

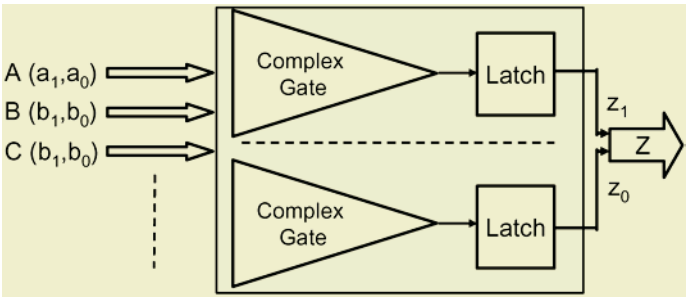

Fig. 2. Targeted cell topology

II.b Dual-Rail Cell Design Method

Adopting the dual-rail cell topology of Fig.2, the design of dual-rail asynchronous primitives amount to the identification of the two complex cells controlling the

latches. This can be realized in six successive steps that constitute the proposed design method.

Step n°1

The first step consists in identifying the three boolean expressions characterizing the functionality to be realized.

Two of those three expressions are related to the settling conditions of the output rails z_1 and z_0 to V_{DD}, indicating respectively that the output has a boolean value of '1' or '0'.

The third expression is related to the settling conditions z_1 and z_0 to Gnd which correspond to the invalid state of the dual-rail encoding. So, in order to illustrate this identification step, let us realize a 3-input OR gate: $A(a_1,a_0)+B(b_1, b_0)+C(c_1,c_0)$. The three expressions characterizing this functionality in dual-rail logic are:

$$Z = A + B + C \tag{1}$$

$$\overline{Z} = \overline{A} \cdot \overline{B} \cdot \overline{C} \tag{2}$$

$$Z^I = A^I \cdot B^I \cdot C^I \tag{3}$$

If expressions (1) and (2), that are related to the settling conditions of z_1 and z_0 to V_{DD} are well known, expression (3) is more specific to the dual-rail encoding of Fig.1 since it defines the settling conditions to Gnd of both rails. In this expression, the character 'I' is used to stipulate that the logical bits Z, A, B and C are all in the invalid state. In the same way, the character 'V' will be used to stipulate that a logical bit is in a valid state (see fig. 1).

Table 1. Truth table of a 3-input OR gate

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Step n°2

After the identification stage of the three characteristic expressions, the second step of the method consists in rewriting these three expressions under a canonical form. This can easily be done starting from the truth table of the functionality to be realized (see Table 1). For the considered 3-input OR gate, this step leads to the following expressions:

$$Z = A \cdot B \cdot C + A \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot C + A \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C \tag{4}$$

$$\overline{Z} = \overline{A} \cdot \overline{B} \cdot \overline{C} \tag{5}$$

$$Z^I = A^I \cdot B^I \cdot C^I \tag{6}$$

Step n°3

In the third step, the canonical expressions obtained in the second step are reformulated so that the dual-rail encoding appears explicitly. This translation, from a natural encoding to the dual-rail encoding of Fig.1, is done using the conversion table (see Table 2) that defines the equivalences between the traditional simple rail logic and the dual-rail logic. Note that in table 2, 'bit' and 'bit!' mean that the 'bit' logical variable correspond respectively to '1' logical value and '0' one. So, for example, if we consider the following expression: $\overline{A} \cdot \overline{B} \cdot \overline{C}$, the dual-rail equivalent will be: $A_0B_0C_0$. As a result, for the considered 3-input OR gate, this step leads to the following dual-rail expressions:

$$z_1 = a_1 \cdot b_1 \cdot c_1 + a_1 \cdot b_1 \cdot c_0 + a_1 \cdot b_0 \cdot c_1 + a_1 \cdot b_0 \cdot c_0 + a_0 \cdot b_1 \cdot c_1 + a_0 \cdot b_1 \cdot c_0 + a_0 \cdot b_0 \cdot c_1 \tag{7}$$

$$z_0 = a_0 \cdot b_0 \cdot c_0 \tag{8}$$

$$\overline{z}_1 \cdot \overline{z}_0 = \overline{a}_1 \cdot \overline{a}_0 \cdot \overline{b}_1 \cdot \overline{b}_0 \cdot \overline{c}_1 \cdot \overline{c}_0 \tag{9}$$

Expressions (7-9) are sufficient to derive in a traditional way the topologies of the complex cells. However, for the considered 3-input OR gate, expression (9) leads to serially stack an excessive number of P transistors. In order to overcome this problem, it is necessary to introduce some additional variables. This is done in a fourth step.

Table 2. Single rail to dual-rail conversion table

	Type	**bit**	**bit!**	**Bit^V**	**Bit^I**
A(a1,a0)	Input	a_1	a_0	$a_1 \oplus a_0 \equiv a_1 + a_0$	$a_1 ! \cdot a_0 !$
B(b1,b0)	Input	b_1	b_0	$b_1 \oplus b_0 \equiv b_1 + b_0$	$b_1 ! \cdot b_0 !$
C(c1,c0)	Input	c_1	c_0	$c_1 \oplus c_0 \equiv c_1 + c_0$	$c_1 ! \cdot c_0 !$
Z(z1,z0)	Output	z_1	z_0	$z_1 \oplus z_0 \equiv z_1 + z_0$	$z_1 ! \cdot z_0 !$

Step n°4

If it is possible to insert any additional variables to implement the P transistor arrays, it is also possible to take advantage of the dual-rail encoding specificities. Among those main specificities of the dual-rail encoding, two are particularly interesting. The existence of forbidden state (1,1) constitutes the first one, while the second is related to the mutually exclusive behaviour of the rails conveying the boolean value of a logical bit (see Table 2). These two specificities confer some interesting properties to the 'exclusive-or' operation.

Table 3. Truth table of $a_1 \oplus a_0$

a_1	a_1	A	$a_1 \oplus a_0$	$a_1 + a_0$
0	0	Invalid	0	0
0	1	Valid, '0'	1	1
1	0	Valid,'1'	1	1
1	1	forbidden	Impossible	Impossible

As shown in Table 3, the existence of the prohibited state allows defining a bijection between the state of validity of the logical bit A, and the value of $a_1 \oplus a_0$ which is strictly equal to $a_1 + a_0$.

This bijection, that justifies the single-rail to dual-rail conversion table (see Table 2), can be used to redefine the settling conditions to the invalid state of the output

bit. For the considered 3-input OR gate, this implies the definition of three additional variables:

$$\begin{aligned} U &= a_1 \oplus a_0 = a_1 + a_0 \\ V &= b_1 \oplus b_0 = b_1 + b_0 \\ W &= c_1 \oplus c_0 = c_1 + c_0 \end{aligned} \tag{10}$$

These three intermediate values being defined, it is afterward possible to simplify the expressions (7), (8), (9) to obtain:

$$z_1 = a_1 \cdot V \cdot W + a_0 \cdot b_1 \cdot W + a_0 \cdot b_0 \cdot c_1 \tag{11}$$

$$z_0 = a_0 \cdot b_0 \cdot c_0 \tag{12}$$

$$\bar{z}_1 = \bar{z}_0 = U \cdot V \cdot W \tag{13}$$

Expressions (11), (12), (13) enable implementing the N and P transistor arrays without transgressing the rules defining the maximum number of transistor that can be serially stacked.

Step n°5

The fifth step consist in designing the schematic of the two complex cells starting from expressions (11) (12) (13) and considering that the additional variables are delivered by the neighbourhood of the cell. This can be performed in traditional way. In the case of the 3-input OR gate, this leads to the schematic of the Fig.3.

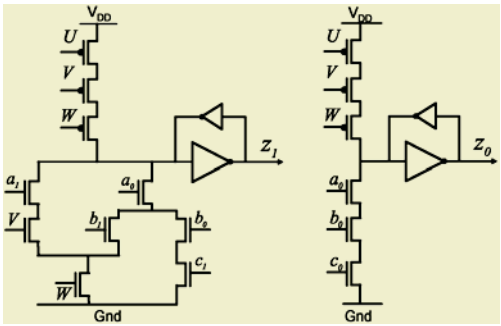

Fig. 3. Proposed dual-rail 3-input OR gate schematic

Step n°6

The final step aims at obtaining the complete schematic of the dual-rail primitive. This is achieved by implementing the calculus of the additional variables, i.e. by integrating the exclusive-or of all the pairs of rails carrying a single binary value.

These 2-input XOR gates, being equivalent to classical 2-input OR gates, as demonstrated earlier, they can be integrated at a low cost using usual single-rail 2-input OR gate. Consequently, it is very easy to obtain the complete schematic (see Fig. 4) of the dual-rail 3-input OR gate which is constituted of 42 transistors. This schematic clearly highlights that the incoming signals will have to cross in the worst case: two layers of transistors (elementary layer) allowing the evaluation of the intermediate values U, V and W, a layer related to the complex gates and finally a layer relating to the output latches.

Consequently, if the transistors are properly sized, the forward latency, i.e. the propagation delay of the 3-input OR gate under consideration will be in the worst case of four transistor layers what is very little with respect to more usual implementations, as we will see in section III.

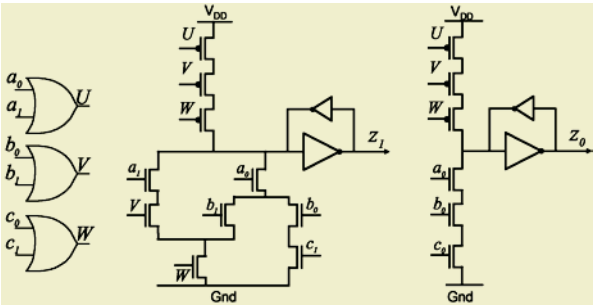

Fig. 4. Pseudo-static implementation of the 3-input OR gate.

It should be noted that this pseudo static implementation of the 3-input OR gate can easily be transformed into a fully static one (see Fig.5). However, if this can reduce significantly the propagation delays of the considered dual-rail 3-input OR gate, this increases its realization cost.

III Performance Analysis

In order to evaluate the performance and the limits of the method suggested herein, we derived the dual-rail schematics of various combinatorial primitives adopting the proposed method and also three other implementation styles.

Among these three implementation styles, two of them have the characteristic of using only cells that can be traditionally found in industrial libraries such as INV, NAND, NOR, Majority and AO222 gates.

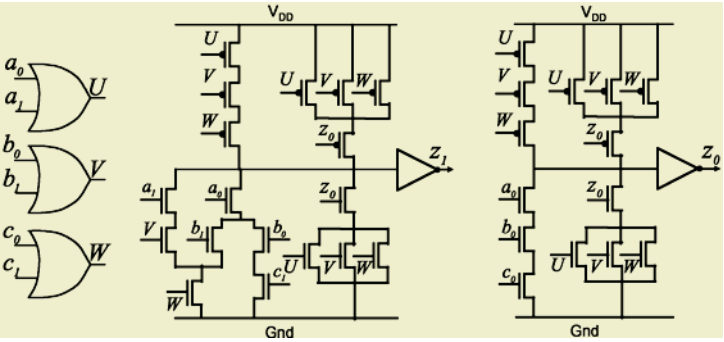

Fig. 5. Proposed static schematic of the dual-rail 3-input OR gate

Those two styles are afterwards denoted by AO222 and Majority gate based styles since they do use (Fig.6) respectively AO222 gates and Majority gates to implement the electrical rendezvous (C-element) of two signals. The third implementation style considered in the remainder of this paper is the one proposed in [5].

III.a Implementation Cost

The implementation of those various boolean functions has allowed evaluating the integration cost of the four implementation styles considered. Table 4 that gives the obtained results highlights the benefits offered by the suggested design technique. As shown, following this technique we are able to design the most widely used functionalities with up to 65% less transistors.

This reduction of the realization cost, ranging from 43% to 73%, is due to the fact that several electrical paths share the same state holding element. This pooling of the state holding element appears clearly while comparing the structure of Fig.4 to that of Fig.6. Indeed, one can note that only one state holding element by rail is required for the 3-input OR gate we proposed, whereas eight state holding elements (8 of the C-element) are necessary to guarantee the correct behaviour of the structures represented in fig. 6.

If the pooling of the state holding elements explains partially this reduction of the integration cost, this one is mainly due to the fact that the suggested method enables the implementation of 3 and 4-input gates without necessarily cascading several 2 input gates. This constitutes a decisive advantage in term of cost, but also in term of speed, since the number of elementary layers crossed by the signals is also reduced. This point is illustrated by Table 5.

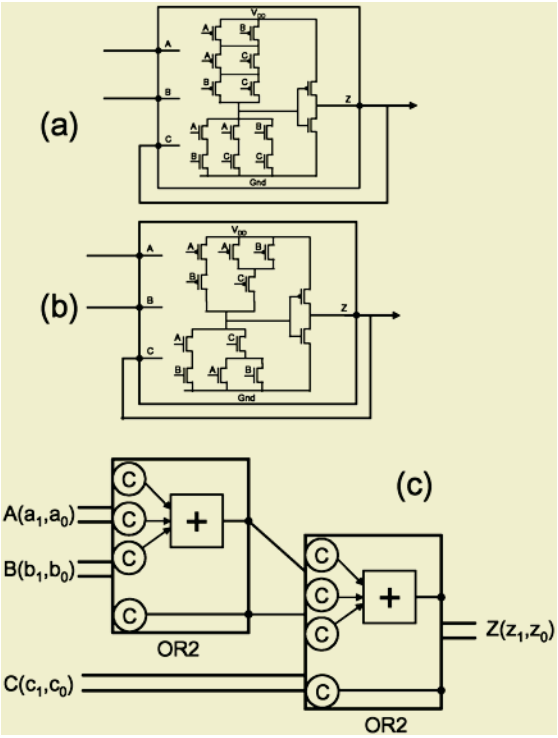

Fig. 6. Wiring (a) an AO222 and (b) a majority cell to obtained a C-element, (c) synoptic scheme of a dual-rail OR3 realized with AO222 or Majority cells; © is the symbol of the C-element

Table 4. Realization cost

Integration cost (transistor count)					
Dual-rail Cell	**Proposed Pseudo-static style**	**Proposed Static Style**	**AO222 based style**	**Majority based style**	**[5]**
Or2 / And2 / Nor2 /Nand2	30	38	64	56	42
Or3 / And3 / Nor3 /Nand3	42	54	128	112	84
Or4 / And4 / Nor4 /Nand4	56	72	192	168	126
Xor2 / Xnor2	32	40	68	60	44
Xor3 / Xnor3	52	64	136	120	88
Xor4 / Xnor4	72	88	204	180	132
AO21/AOI21	43	55	128	112	84
AO22/AOI22	58	74	192	168	126

Table 5. Min and Max numbers of elementary layers crossed by the signals

Min and Max numbers of elementary layers crossed by the signals					
Dual-rail Cell	**Proposed Pseudo-static style**	**Proposed Static Style**	**AO222 based style**	**Majority based style**	**[5]**
Or2 / And2 / Nor2/Nand2	4/2	4/2	4/2	4/2	2
Or3 / And3 / Nor3 /Nand3	4/2	4/2	8/4	8/4	4
Or4 / And4 / Nor4 /Nand4	4/2	4/2	8/4	8/4	4
Xor2 / Xnor2	4/2	4/2	4/2	4/2	2
Xor3 / Xnor3	4/2	4/2	8/4	8/4	4
Xor4 / Xnor4	4/2	4/2	8/4	8/4	4
AO21/AOI21	4/2	4/2	8/4	8/4	4
AO22/AOI22	4/2	4/2	8/4	8/4	4

III.b Speed Performances

If the number of elementary layers crossed by the signals is a good first order indicator of the speed performances, we wanted to quantify them more precisely. Therefore we simulated the propagation delays of various cells under different loading and controlling conditions. More precisely, we applied on theirs inputs linear ramps of various durations (10, 50, 100, 200 and 500ps) and we varied the values of the output loading capacitance (5, 10, 20, 50, 100fF). Table 6 summarizes the results obtained taking the AO222 based design style as reference.

Table 6. Average propagation delay reduction

Average propagation delay reduction				
Cell	**Proposed Pseudo Static Style**	**AO222 based style**	**Majority based style**	**[5]**
Or2	5.2 %	0	22.6 %	18.1 %
Or3	38.7 %	0	38.1 %	26.6 %
Or4	35.8 %	0	38.9 %	25.3 %
Xor2	7.3 %	0	27.2 %	19.0 %
Xor3	39.0 %	0	38.9 %	28.1 %
Xor4	30.7 %	0	37.7 %	27.4 %
AO21	36.8 %	0	34.3 %	28.0 %
AO22	28.5 %	0	31.8 %	26.9 %

As expected from table 5, for 2-input cells the propagation delays of our cells are equivalent to those obtained for the AO222 and Majority based cells while they are smaller for 3 and 4 input gates.

III.c Energy Consumption

If the integration cost, evaluated as the number of transistors necessary to realize of boolean function, is a first order metric of the energy consumption, we characterized it more precisely. Therefore, we simulated the energy consumption for various cells controlled by voltage ramps of different durations (10, 50, 100, 200 and 500ps) and loaded by different capacitances (5, 10, 20, 50, 100fF). Table 7 gives the average energy consumption reduction of the various implementation styles considered herein taking the AO222 based design style as reference.

Table 7. Average energy consumption reduction

Average energy consumption reduction				
Cell	**Proposed Pseudo Static Style**	**AO222 based style**	**Majority based style**	**[5]**
Or2	-30.3 %	0 %	17.0 %	14.7 %
Or3	7.2 %	0 %	26.6 %	31.9 %
Or4	11.3 %	0 %	29.2 %	34.6 %
Xor2	-36.1 %	0 %	17.6 %	19.1 %
Xor3	2.2 %	0 %	29.1 %	32.5 %
Xor4	6.1 %	0 %	30.7 %	33.5 %
AO21	5.5 %	0 %	29.8 %	27.7 %
AO22	7.5 %	0 %	30.9 %	27.0 %

As expected, the 3 and 4-input gates that we proposed have smaller energy consumption than the AO222 based cells have. However, this is not the case for our 2-input cells that consumes significantly more than the other implementation styles. This is mainly explained by the energy dissipated in the 2-input OR gates generating the internal signals. They do have, indeed a high toggling rate that penalizes the energy consumption.

IV Conclusion

We have introduced a method to quickly design dual-rail CMOS asynchronous primitives. The primitives obtained thanks to this design technique have better or similar performances than those of more traditional implementation styles such as the AO222 based style. Moreover, this technique allows designing 3 and 4 input CMOS dual-rail cells. To our knowledge, this was not possible up to now. As a result, the cells obtained with the design techniques introduced in this paper are good candidates for the design of dual-rail asynchronous and synchronous circuits.

References

1. T. Chelcea, A. Bardsley, D. A. Edwards, S. M. Nowick" A Burst-Mode Oriented Back-End for the Balsa Synthesis System." Proceedings of DATE '02, pp. 330-337 Paris, March 2002

2. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev "Logic synthesis of asynchronous controllers and interfaces ", Springer-Verlag, ISBN: 3-540-43152-7 (2002)
3. M. Renaudin et al, "TAST", Tutorial given at the 8th international Symposium on Advanced Research in Asynchronous Circuits and Systems, Manchester, UK, Apr. 8-11, 2002.
4. C. Piguet, J. Zhand "Electrical Design of Dynamic and Static Speed Independent CMOS Circuits from Signal Transistion Graphs" PATMOS '98, pp. 357-366, 1998.
5. P. Maurine, J. B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin "Static Implementation of QDI asynchronousprimitives", PATMOS 2003
6. J. Sparso, S. Furber "Principles of Asynchronous Circuit Design – A Systems Perspective", Kluwer Academic Publishers (2001)