Time-Multiplexed Dual-Rail Protocol for Low-Power Delay-Insensitive Asynchronous Communication

Marco Storto and Roberto Saletti

Dipartimento di Ingegneria della Informazione:
Elettronica, Informatica, Telecomunicazioni
University of Pisa – Via Diotisalvi 2, I-56126 PISA (Italy)
Tel.+39-050-568511, fax +39-050-568522, E-mail: saletti@iet.unipi.it

Abstract

A protocol for delay-insensitive asynchronous communication based on the time-multiplexing of two bit on a dual rail line is proposed and analyzed. Compared to conventional dual-rail protocols, it halves the number of bus wires and, according to simulations, significantly reduces energy consumption with small throughput loss and nearly no area overhead.

1. Introduction

As feature size of CMOS technology decreases, the balance between gate and wire delays is going worse, with wiring delays moving into dominance. The dominant wire delay causes many troubles in synchronous designs: since wire delays depend on the final physical design, the system performance strongly depends on physical implementation, so many design cycles are needed to obtain a correct circuit.

Delay-insensitive asynchronous communication can simplify design, reducing sensitivity to circuit delays; nevertheless, conventional delay-insensitive techniques (i.e. dual-rail coding) are expensive in terms of energy [1] and routing complexity. This paper proposes an alternative coding scheme, based on time-multiplexing of two bits on a dual-rail line, which reduces energy consumption and routing complexity, with no area overhead and only a small throughput loss.

The paper is organised as follows. Section 2 describes existent delay-insensitive communication techniques. The proposed protocol is described in Section 3. Section 4

shows the circuits needed to realize an interconnection bus based on such a signalling. Simulation results for the proposed protocol are compared to those for traditional ones in Section 5.

2. Dual-Rail encoding techniques

Delay-insensitive asynchronous communication requires a redundant encoding, in which some symbols represent invalid values (no data). The most common choice is *4-phase dual-rail* encoding [2], where each bit is coded on two wires: '00' is the invalid code (NULL), and '01' and '10' are logical 0 and 1. The transmission of a bit with this code requires a transition from NULL state to one of the logical state and (after receiving an acknowledge) another transition back to the NULL state, so obtaining a 4-phase cycle.

In fact, the NULL state is used as a *spacer* between consecutive data values, so allowing a simple recognition of the arrival of a new data word (i.e. simple circuits). However, the transmission of the spacer after each data reduces throughput and wastes energy, since $2N+2$[1] transitions are needed to send an N bit word.

An alternative technique is to use transition signalling (2-phase), in which the logical 0 and 1 are transmitted as transition on the first and the second wire of the pair (respectively). This code appears more energy-efficient than the previous one, since uses only N+1 transitions to send the same N bit word. However, transition signalling requires state variables in transmitter and receiver to hold the state of the line, leading to very complex circuits. Therefore, transition signalling usually dissipates more power than 4-phase one.

A more efficient signalling method, called *Level-Encoded 2-Phase Dual-Rail* (*LEDR*), has been proposed in [3] and independently in [4] (where it is called *Four-State Coding*). This code still uses two wires to encode one bit, but allows a 2-phase transmission with circuits simpler than those required by transition signalling.

---

[1] Including transitions on acknowledge wire.

In LEDR encoding, as shown in Table 1, *each* logical value (0 and 1) is represented with *two* symbols of *opposite* phase. In a data stream, odd and even symbols must alternate, so one and only one wire changes state for every successive data transmitted (i.e. bus signals appears "Gray-coded"). This code leads to simple *decoder* circuits, since one of the wires always carries the logical value transmitted, and phase changes (signalling arrival of a new word) are easily detected by means of a XOR gate. LEDR signalling may be more energy-efficient than 4-phase dual-rail one, using only N+1 transition to send an N bit word. Nevertheless, the *encoder* circuit is still complex, because it has to hold the previous phase of the line and choose the symbol to send in order to guarantee the correct alternation of phases. The overall energy consumption is strongly dependent on the actual circuit implementation.

Value	Symbol	Phase
0	0 0	even
1	1 1	even
0	0 1	odd
1	1 0	odd

Table 1: LEDR code

All those dual-rail codes uses twice as many wires as data bits, making their application in large buses very expensive in terms of routing area. The 4-phase one is rather wasteful of energy, but is often preferred by designers owing to its simpler implementation [5]. This paper proposes an improvement of the LEDR coding scheme, called Time-Multiplexed LEDR (TM-LEDR), that uses only one wire per bit, by multiplexing in time two bits (adjacent in the word) on a dual-rail line. Besides halving the wire number, the proposed code allows a significant reduction in energy consumption, with small throughput loss and nearly no added circuit complexity, as compared with the 4-phase dual-rail code.

3. Time-Multiplexed dual-rail protocol

The main feature of TM-LEDR signalling is to use *one* dual-rail line to send sequentially *two* adjacent bits of a word, merging two 2-phase cycles in a 4-phase one. The protocol requires transmitter to wait for *ack* wire low, then put the coded value of first half-word (odd bits) on the bus wires (*b0..b3*), toggling the bus into the odd state,

and wait for *ack* to go high. Meanwhile, the receiver waits for bus wire pairs to go into the odd phase, then stores the first half-word and toggles *ack* in high state. When the transmitter sees *ack* high, it puts the coded value of second half-word (even bits) on the bus, toggling it back into the even phase; the receiver waits for this event, then stores the second half-word and toggles *ack* back to low state. See Fig. 1 for a transmission example.

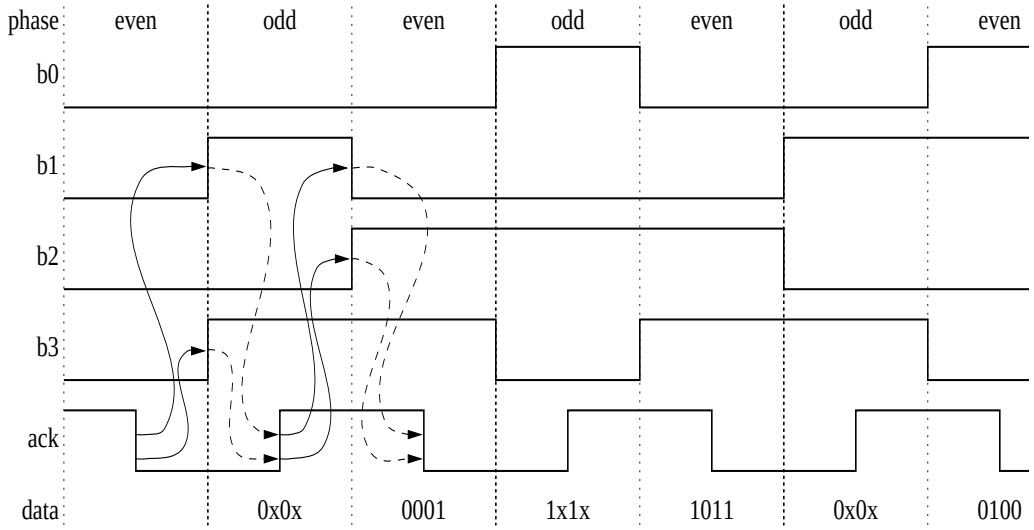

Fig. 1: Transmission example

	Value	symbol
First bit	0	0 1
odd phase	1	1 0
Second bit	0	0 0
even phase	1	1 1

Table 2: TM-LEDR code

The proposed code, shown in Table 2, appears similar to the LEDR one; however, TM-LEDR attributes a different meaning to the phase of the line: *odd symbols represent the first bit, even symbols the second one*. Bus signals still appear "Gray-coded", but after each transmission cycle the line is always brought back in the initial (even) phase, and the relationship between data values and its coded representation is *combinational*, which leads to (encoding) circuits much simpler than LEDR's ones.

Actually, the TM-LEDR code simply trades speed for wire number as compared to LEDR one. Nevertheless, owing to the lower circuit complexity, it significantly re-

duces energy consumption. The use of time multiplexing obviously reduces throughput, as compared with *2-phase* dual-rail. However, TM-LEDR's throughput is only slightly less than 4-phase dual-rail's one, because the time-consuming return-to-zero phase of this one is now used to carry information; in fact, in both protocols a complete cycle requires four sequential bus transitions. The cycle time of a 2-phase protocol is obviously shorter but, when power-efficiency is the main design target, 2-phase protocols are often avoided, due to the greater circuit complexity. If a delay-insensitive communication is needed in an overall 4-phase design, LEDR signalling requires rather inefficient conversion circuits, so a 4-phase protocol may be preferable.

The major drawback of TM-LEDR communication is the increased latency, because data are available to the receiver after three sequential transitions on the bus, whilst in 4-phase dual-rail and LEDR only one bus transition is needed.

4. Circuit implementation

The proposed TM-LEDR communication seems well suited for interconnection between integrated macrocells, where it allows a delay-insensitive communication without doubling the wire number. However, a bundled-data approach can lead to more efficient circuits [1] inside macrocells, so we have designed converters between the proposed protocol and a 4-phase bundled-data one. The chosen protocol assumes

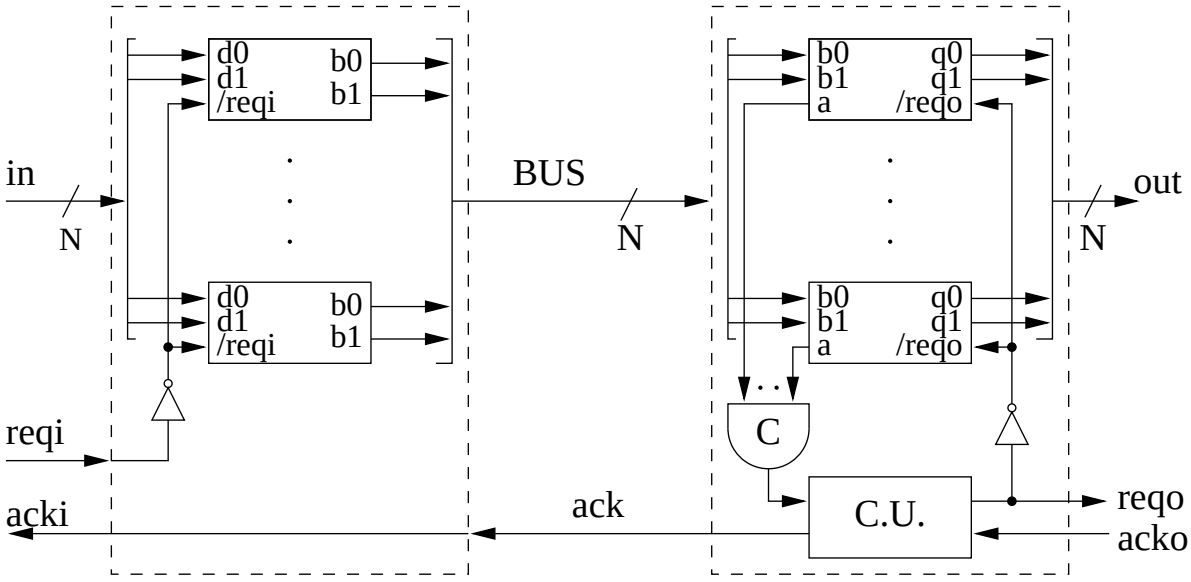

Fig. 2: TM-LEDR bus

data validity when *req* is high, but circuits can easily be adapted to other bundled-data protocols.

Both converters (transmitter and receiver) have a modular structure (Fig. 2), composed by bit pair handlers (encoder in transmitter and decoder in receiver); a simple control unit in the receiver collects the "data ready" signals from each bit pair (by means of a Muller C-gate) and handles the output handshake. At the moment, this control unit is designed as a Burst-Mode state machine [6], but any other design style (namely STG) may be used as well.

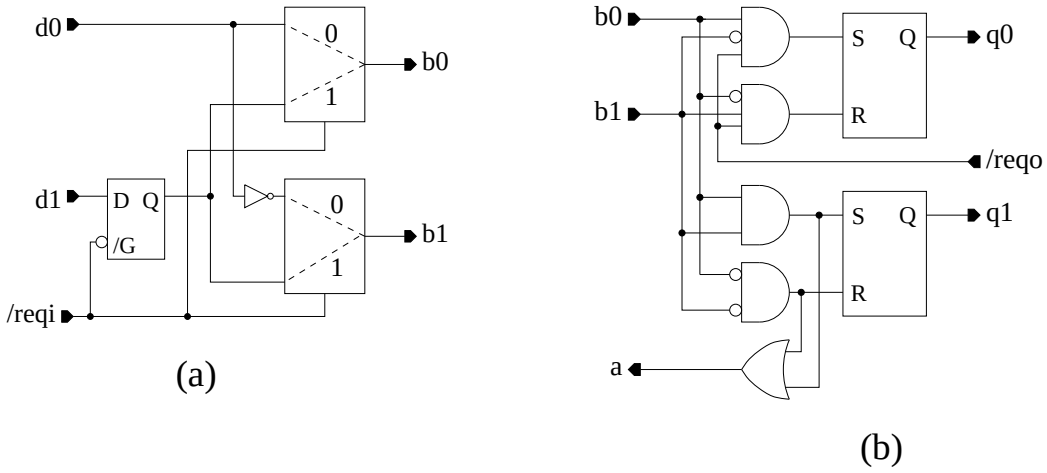

Fig. 3: TM-LEDR encoder (a) and decoder (b)

The combinational nature of TM-LEDR code allows a fairly simple implementation of the bit pair encoder (Fig. 3a); since *ack* signal is directly connected to *acki*, bus phase may be directly associated to the level of *reqi* signal (i.e. *reqi* high -> odd phase, *reqi* low -> even phase). In the bundled-data protocol chosen, data are no longer valid when *reqi* goes low, so we need a latch to hold the second bit during even phase. This circuit uses some timing assumptions: when *reqi* goes high, multiplexers must switch before the latch becomes transparent, and input data must be hold after *reqi* goes low, to satisfy the latch hold time requirement. Moreover, multiplexers must be hazard-free[2].

[2]This is easily obtained using a transmission-gate or tri-state inverter implementation.

The bit pair decoder is simple too (Fig. 3b): it uses two RS latches driven by decoding the two bus wires, and a XOR function to detect the bus phase (*a* in Fig. 3b). Actually, one of the latches is not necessary, but it allows to start the next bus cycle while the output handshake is still in progress, improving throughput (the receiver acts as a pipeline stage). Odd phase decoders are disabled when *reqo* is high accordingly.

5. Simulation results

The proposed circuits have been designed with the AMS 0.8 µm CMOS technology, using the standard-cell library provided by the foundry. To evaluate TM-LEDR performance, we have designed converters between the bundled-data protocol and standard dual-rail ones (4-phase and LEDR), using a similar structure and the same synthesis technique (see Fig. 4). All the designs include latches in the receiver to (partially) decouple bus cycle from output handshake.

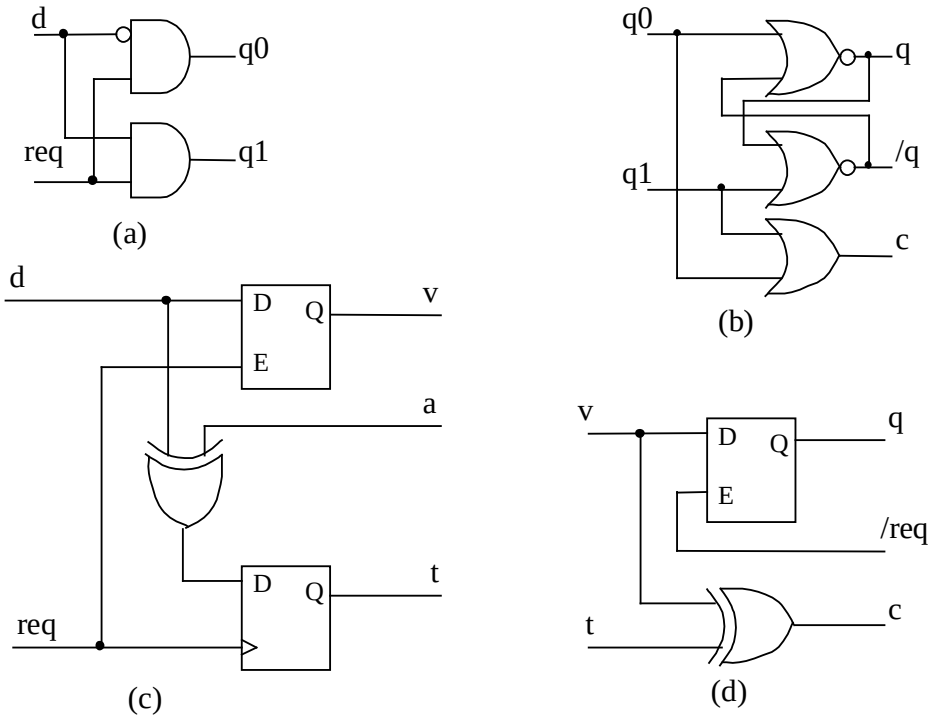

Fig. 4: Reference circuits. (a) DR encoder. (b) DR decoder. (c) LEDR encoder. (d) LEDR decoder

To estimate energy consumption, we have simulated the circuits with SPECTRE, using foundry's CMOS47 (BSIM.3v2) model, V_{cc} = 5V and typical process conditions.

Simulation with a pseudo-random sequence of 2 bit words gave the results shown in Table 3a, from which the consumption for a 32 bit bus can be extrapolated. The comparison between these values (Table 3b) shows that TM-LEDR is more energy-efficient than the other protocols.

Bus load (pF / wire)		1	2	3
Encoder and decoder consumption (pJ / bit)	TM-LEDR	56.5	69.8	83.5
	DR	60.7	86.7	113
	LEDR	76.6	89.1	102
Control units consumption (pJ)	TM-LEDR	142	166	191
	DR	87.0	112	136
	LEDR	181	193	206

(a)

Bus load (pF /wire)	1	2	3
TM-LEDR / DR	96%	83%	76%
TM-LEDR / LEDR	71%	79%	82%

(b)

Table 3: (a) Energy consumed in a transmission cycle. (b) Comparison between energy required to send a 32 bit word

Latency (i.e. time elapsed from transmission to arrival of data word) and cycle time (i.e. time between arrival of consecutive data) for a 32 bit bus have been obtained with VERILOG-XL simulation program, using cell models (provided by the foundry) which provide load-dependent delays. Simulation results (Table 4a) show an increase in latency as expected, but only a moderate one in cycle time, as appears in Table 4b.

Table 5 shows the estimated area occupied by the converter circuits, calculated as the sum of the used cell areas. TM-LEDR circuits have the same complexity as DR ones, while LEDR requires about 50% more area. A further advantage of the TM-LEDR protocol, that it is not weighted at all, is that it uses half the number of wires for the same bus width, so the routing area is halved.

Bus load (pF / wire)		1	2	3
Latency (ns)	TM-LEDR	18.8	20.4	21.9
	DR	6.0	6.5	6.9
	LEDR	9.6	10.1	10.7
Cycle time (ns)	TM-LEDR	19.9	21.9	23.9
	DR	17.2	19.2	21.3
	LEDR	15.3	16.3	17.3

(a)

Bus load (pF / wire)	1	2	3
$\frac{\text{TM-LEDR}}{\text{DR}}$	115%	114%	112%
$\frac{\text{TM-LEDR}}{\text{LEDR}}$	130%	135%	138%

(b)

Table 4: (a) Latency and cycle time. (b) Comparison between cycle times

	TM-LEDR	DR	LEDR
Per bit	4900 μm^2	4830 μm^2	7350 μm^2
Control unit	7700 μm^2	3150 μm^2	16400 μm^2
32 bit bus	100%	96%	150%

Table 5: Comparison between circuit area

6. Conclusions

A protocol for delay-insensitive communication based on time multiplexing of two bits onto a dual-rail line has been proposed. The main feature of the protocol is that two consecutive 2-phase cycles are merged in a single 4-phase one, where two adjacent bits are transmitted. This leads to a reduction in the speed of communication but also allows a significant reduction in energy consumption and circuit complexity. Interface circuits towards a communication bus have been described and simulated, and compared to other delay-insensitive communication protocols.

The main results obtained are that the proposed TM-LEDR communication halves the wire number and, according to simulation, significantly reduces energy consumption with a small throughput loss, as compared to conventional dual-rail protocols.

References

[1] L.S. Nielsen and J. Sparsø. “A low-power asynchronous data-path for a FIR filter bank” *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, March 1996.

[2] Charles L. Seitz. “System Timing” in Carver A. Mead and Lynn A. Conway, editors, *Introduction to VLSI systems*, chapter 7, Addison-Wesley, 1980.

[3] Mark Dean, Ted Williams, and David Dill. “Efficient self-timing with level-encoded 2-phase dual-rail (LEDR)” in Carlo H. Séquin, editor, *Advanced Research in VLSI: Proceedings of the 1991 UC Santa Cruz Conference*, pages 55-70. MIT Press, 1991.

[4] Antony J. McAuley. “Four state asynchronous architectures”, *IEEE Transactions on Computers*, 41(2):129-142, February 1992.

[5] Pedro A. Molina and Peter Y.K. Cheung. “A quasi delay-insensitive bus proposal for asynchronous circuits”, *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, April 1997.

[6] Kenneth Y. Yun and David L. Dill. “Unifying synchronous/asynchronous state machine synthesis”, *Proc. International Conf. Computer-Aided Design (ICCAD)*, pages 255-260. IEEE Computer Society Press, November 1993.