Synthesis of multiple rail phase encoding circuits

Andrey Mokhov, Crescenzo D'Alessandro, Alex Yakovlev
Microelectronics System Design Group, School of EECE, Newcastle University, UK
{andrey.mokhov, crescenzo.dalessandro, alex.yakovlev}@ncl.ac.uk

***Abstract*—Multiple rail phase encoding data communication protocol introduced recently has several unexploited advantages over traditional encodings. The main difficulties in using it arise from the absence of practical and scalable implementations of controllers for phase encoded data transmission.**

This paper focuses on techniques for generating efficient circuits for multiple rail phase encoders, decoders and repeaters. The circuits are specified and synthesised using Conditional Partial Order Graph model which provides robust and scalable area-efficient gate-level implementation of the controllers.

I. Introduction

The design of the on-chip interconnect fabric is a crucial part of the design of large complex Systems-on-Chip (SoCs). The many-layered set of design requirements imposed by the ever-increasing performance constraints require the identification of fast, reliable and scalable data/control signalling techniques. Networks-on-Chip (NoCs) are one such method, responding to the need of modularity and scalability, and also adaptability: the network can be designed a priori, freeing the designer team from the task of designing ad-hoc solutions for their designs. Mentioning layers in the preceding text is apt in the context of NoC: these are typically designed using a layered approach (see, for instance, [1]), which identify and separate the requirements and characteristics of physical communication, node-to-node interaction all the way to application-specific requirements. The physical layer of a NoC, and more generally on-chip signalling, is the underlying motivation of this paper.

D'Alessandro et al. in [4] introduced the concept of phase encoding for on-chip signalling, where the information is encoded into the sequence of events over a number of lines: this provides a way to concentrate information into symbols more than by using binary encoding, with the added advantage of reliability to single-event upsets [3]. However, the previous work does not describe a satisfactory method to generate encoders and decoders for this communication scheme: the structures described are limited to small number of wires (*rails*), and their scalability is not clearly described.

While conventional control logic specification and synthesis methods based on STGs [2] or on Burst-mode FSMs [7] have certain advantages, they cannot be directly applied to the problem of phase encoders circuitry synthesis as shown in [6]. In particular, the size of specification of *matrix phase encoder* (see Section III-B) is exponential w.r.t. the number of output rails in these models. To overcome this, the paper defines and solves the problem of specification and synthesis of multiple rail phase encoding circuits using the model of *Conditional Partial Order Graphs* which was recently introduced in [6], providing efficient gate-level implementations for the circuits.

II. Phase encoding essentials

Phase encoding protocol was introduced by D'Alessandro *et al.* in [4]. The initial idea was to encode an information bit into phase difference between two switching signals. The idea was further extended into *multiple-rail phase encoding* [3] which uses several wires for communication and data is encoded in the order of occurrence of transitions on the communication lines. Figure 1 shows an example of 4-wire phase encoding communication channel. The order of rising signals on wires $\{a, b, c, d\}$ indicates that permutation $abdc$ is being sent. In total it is possible to send $n!$ different permutations over an n-wire channel. This makes the multiple rail phase encoding protocol very attractive for its information efficiency.

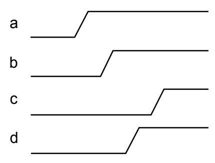
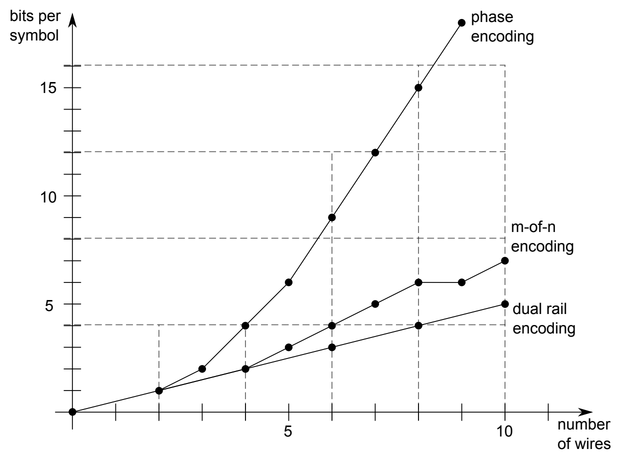


Figure 1. Data symbol in multiple-rail phase encoding channel

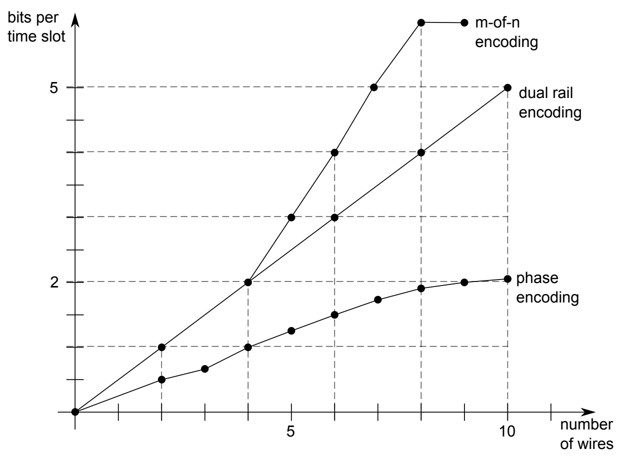
The amount of information that can be sent in a symbol in n-wire channel grows faster than linearly. This is due to the fact that

$$\log_2(n!) = \sum_{k=1}^{n} \log_2 k \approx \int_1^n \log_2 x dx = \Theta(n \log_2 n) \qquad (1)$$

The numeric comparison of multiple rail phase encoding protocol with *dual-rail* and *m-of-n encoding protocols* [8] for up to 10-wire communication channels is shown in Figure 2. The results of m-of-n encoding are calculated for the most informative case when $m = \lfloor \frac{n}{2} \rfloor$. Subfigure (a) shows information efficiency with respect to a symbol size, while Subfigure (b) - with respect to a time slot. Notice that phase encoding is dominating on the first graph and shows rather bad results on the second. However this should not be misleading: although n-wire phase encoding protocol needs n time slots to send a data symbol these time slots can be significantly shorter than that of dual rail or m-of-n protocols because each wire switches only once in these n time slots. Therefore the sending and receiving circuitry of a particular wire can work at a speed n times slower than the communication channel as a whole. It allows the time slots to be compressed much more than for dual rail and m-of-n encoding protocols and achieve higher information density over time. Thus phase encoding is potentially optimal in terms of area (number of bits per wire), speed (number of bits per time interval) and power (number



(a) number of bits per symbol



(b) number of bits per time slot

Figure 2. Numeric comparison of DI communication protocols: information efficiency

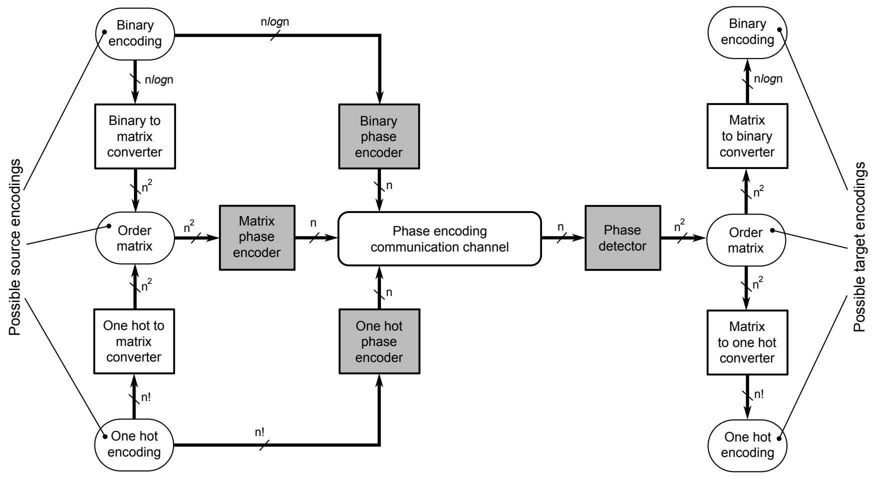

Binary encoding
n*log*n
Binary to matrix converter
Binary phase encoder
n
Order matrix
n^2
Matrix phase encoder
Phase encoding communication channel
Phase detector
One hot to matrix converter
One hot phase encoder
n!
One hot encoding
Matrix to binary converter
Matrix to one hot converter
Possible source encodings
Possible target encodings

Figure 3. Phase encoding communication circuitry (numbers of wires are shown beside communication channels; implementation of functional units that are drawn tinted is covered in the paper)

of signal transitions per bit). Asymptotic comparisons of the protocols can be found in [5].

All these comparisons are theoretical and need experimental refinement. However existing implementations of multiple rail phase encoding circuitry are area inefficient and usually designed by hand for a particular number of wires. This work presents a number of techniques for generating circuits for multiple-rail phase encoding senders/receivers/repeaters. Figure 3 shows the overall phase encoding communication circuitry. Rectangular boxes represent functional units for conversion between different data encodings. The paper covers implementation of the units in tinted boxes. Most of the phase encoding circuits presented in the work are synthesised using Conditional Partial Order Graphs introduced in [6]. The details of the synthesis approach and the used CPOG encoding schemes can be found in [5].

III. Phase encoding repeater

The first multiple rail phase encoding circuit that we are going to synthesise is *phase encoding repeater* [3] – a circuit able to regenerate the deteriorating phase difference between signals in phase encoding communication channel.

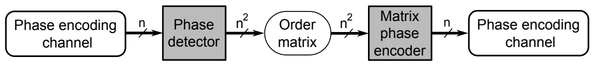


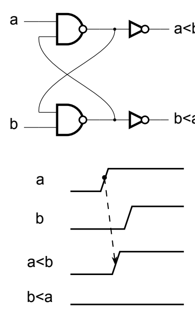
Figure 4. Phase encoding repeater circuitry

Phase encoding repeater consists of two functional parts: a receiver (a *phase detector*, which determines the order of the incoming transitions) and a sender (a *phase encoder*, which has to generate a series of transitions in the order they were received) as shown in Figure 4.

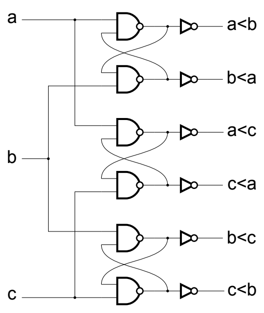
It should be noted that we assume here that the phase encoded symbols arriving via the communication channel to the repeater are correct i.e. all transitions are ordered with appropriate time slot condition. The issues of error behaviour and noise tolerance have been addressed in [3].

A. Phase detector

Phase detector for n-wire communication channel consists of $\binom{n}{2}$ *mutual-exclusion* (*mutex*) *elements* [3]: each for every pair of wires. A possible implementation of a mutex is shown in Figure 5(a): it consists of a pair of cross-coupled NAND gates (an SR-latch) and a simple metastability filter constructed from two inverters. To determine the order of n transitions it is possible to compare their arrival times pairwise (see Figure 5(b) for an example of 3-wire phase detector).



(a) mutex



(b) 3-wire phase detector

Figure 5. Phase detection

The result of phase detection can be seen as a *control matrix* [5] with zero diagonal elements. Thus the subsequent phase encoder should be synthesised using *matrix encoding scheme* [5] to avoid additional encoding conversion circuitry.

B. Matrix phase encoder

Given control matrix $X = \{x_{jk} | j = 1...n, k = 1...n, j \neq k\}$ containing pairwise comparisons of arrival times of n transitions *matrix phase encoder* should generate n output transitions in the specified order.

The control matrix X coming from the phase detector has $n!$ different possible values assignments $\psi_k : X \rightarrow \{0,1\}, k = 1...n!$ each of them corresponding to a partial order of a particular scenario. Conditional partial order graph $H(V, E, X, \phi)$ containing all of them as its projections has the following generic description:

$$\begin{array}{l} V = \{e_j | j = 1...n\} \\ E = \{(e_j, e_k) | j = 1...n, k = 1...n, j \neq k\} \\ X = \{x_{jk} | j = 1...n, k = 1...n, j \neq k\} \\ \phi(e_j) = 1, j = 1...n \\ \phi((e_j, e_k)) = x_{jk}, j = 1...n, k = 1...n, j \neq k \end{array} \qquad (2)$$

Example of such a CPOG for the case of 3 wires is shown in Figure 6.

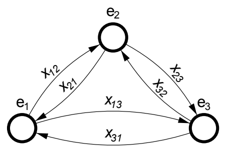


Figure 6. 3-wire matrix phase encoder specification

Having synthesised the CPOG it is possible to derive Boolean equations for physical controller implementation. The controller should have $2\binom{n}{2} = n^2 - n$ inputs $X = \{x_{jk} | j = 1...n, k = 1...n, j \neq k\}$ and n outputs $T = \{t_1, t_2, ..., t_n\}$. Output transition t_k is enabled to fire if all the preceding (w.r.t. to the partial order specified by control matrix X) transitions have already fired i.e.

$$t_k = \phi(e_k) \cdot \bigwedge_{\substack{1 \leq j \leq n \\ j \neq k}} (\phi(e_j) \cdot \phi((e_j, e_k)) \Rightarrow t_j) \qquad (3)$$

This generic equation can be simplified taking into account the particular CPOG specification (2):

$$t_k = \bigwedge_{\substack{1 \leq j \leq n \\ j \neq k}} (x_{jk} \Rightarrow t_j) = \bigwedge_{\substack{1 \leq j \leq n \\ j \neq k}} (\overline{x_{jk}} + t_j)$$

Another optimisation opportunity is to exploit the fact that the control matrix X specifies a *total order* (a special case of partial order $P(V, \prec)$ such that $(a \prec b) \Leftrightarrow \neg(b \prec a)$ i.e. every pair of elements in V is ordered). In our case it means that $\overline{x_{jk}} = x_{kj}$:

$$t_k = \bigwedge_{\substack{1 \leq j \leq n \\ j \neq k}} (x_{kj} + t_j)$$

As the phase encoder should maintain a certain time separation Δ between the generated transitions it is necessary to modify the above equation to take this fact into account:

$$t_k = \bigwedge_{\substack{1 \leq j \leq n \\ j \neq k}} (x_{kj} + t_j^{\Delta})$$

where t_j^{Δ} represents signal t_j delayed for Δ time units. For the purpose of resetting the controller into the initial state after generating the desired sequence of transitions we should also add control signal *go* that would serve as an initiating and resetting signal:

$$t_k = go \cdot \bigwedge_{\substack{1 \leq j \leq n \\ j \neq k}} (x_{kj} + t_j^{\Delta}) \qquad (4)$$

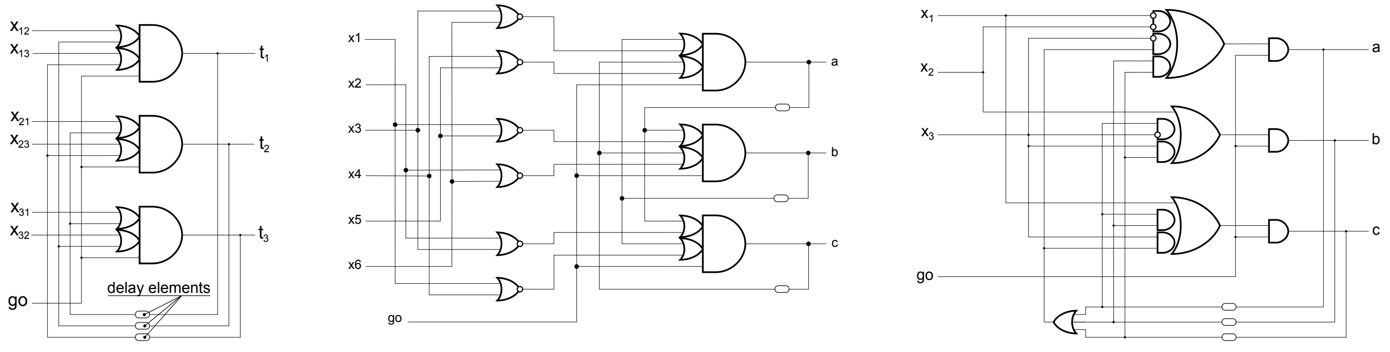
The gate-level implementation of the controller specified with equation (4) is shown in Figure 7(a).

IV. ONE HOT PHASE ENCODER

One hot encoding can be used to specify the order of signal transitions for small values of n (for large values of n the method is inappropriate because it needs $n!$ wires). To send data presented in one hot encoding it is possible to convert it first into matrix form using *one hot code to matrix converter* and then to send the result using matrix phase encoder. Alternatively, to avoid unnecessary conversions it is possible to send one hot data directly using *one hot phase encoder* as shown in Figure 3.

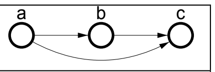
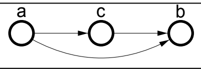
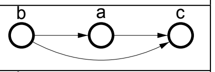
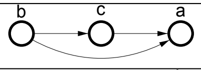
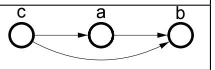
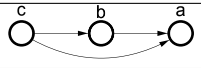
There are $n!$ partial orders $\mathcal{P} = \{P_1, P_2, ..., P_{n!}\}$ specifying the $n!$ scenarios. Using *one hot encoding scheme* [5] it is possible to synthesise CPOG containing all of them.

Consider the following example of synthesis of 3-wire one hot phase encoder. There are 6 one hot control signals $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$ and 6 partial orders corresponding to the possible permutations of output transitions $T = \{a, b, c\}$:

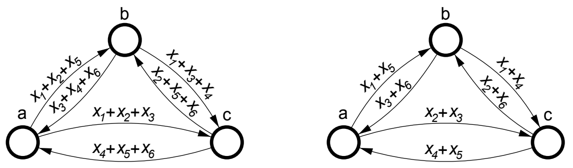


(a) Matrix phase encoder (b) One hot phase encoder (c) Binary phase encoder

Figure 7. 3-wire phase encoding circuits

#	permutation	one hot encoding	partial order
1	(a, b, c)	$\psi_1 = (1, 0, 0, 0, 0, 0)$	
2	(a, c, b)	$\psi_2 = (0, 1, 0, 0, 0, 0)$	
3	(b, a, c)	$\psi_3 = (0, 0, 1, 0, 0, 0)$	
4	(b, c, a)	$\psi_4 = (0, 0, 0, 1, 0, 0)$	
5	(c, a, b)	$\psi_5 = (0, 0, 0, 0, 1, 0)$	
6	(c, b, a)	$\psi_6 = (0, 0, 0, 0, 0, 1)$	

The synthesised CPOG is shown in Figure 6(a). Using CPOG logical optimisation [6] it is possible to simplify it into slightly smaller CPOG shown in Figure 6(b).



(a) synthesised CPOG (b) optimised CPOG

Figure 8. CPOGs for 3-wire one hot phase encoder

The gate-level implementation of 3-wire one hot phase encoder specified with the obtained optimal CPOG is shown in Figure 7(b).

V. Binary phase encoder

Binary encoding is traditionally used for data transmission. The CPOG synthesis process is the same as for one hot phase encoding with the only exception that the *binary encoding scheme* [5] is used. For the case of 3-wire binary phase encoder, the following set of Boolean equations for output signals $T = \{a, b, c\}$ is eventually derived:

$$\begin{cases} a = ((\overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}x_3) \Rightarrow b^{\Delta})((\overline{x_1}x_2x_3 + x_1\overline{x_2}\,\overline{x_3}) \Rightarrow c^{\Delta}) \\ b = ((\overline{x_1}\,\overline{x_2}\,\overline{x_3} + x_1\overline{x_2}\,\overline{x_3}) \Rightarrow a^{\Delta})((\overline{x_1}\,\overline{x_2}x_3 + x_1\overline{x_2}x_3) \Rightarrow c^{\Delta}) \\ c = ((\overline{x_1}\,\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3}) \Rightarrow a^{\Delta})((\overline{x_1}\,\overline{x_2}\,\overline{x_3} + \overline{x_1}x_2x_3) \Rightarrow b^{\Delta}) \end{cases}$$

Taking into account binary assignment set $\Psi = \{(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 1)\}$ the above equations can be simplified into

$$\begin{cases} a = \overline{x_1}\,\overline{x_2} + b^{\Delta}c^{\Delta} + \overline{x_3}(b^{\Delta} + c^{\Delta}) \\ b = x_2 + \overline{x_3}a^{\Delta} + x_3c^{\Delta} \\ c = x_1 + a^{\Delta}b^{\Delta} + x_3(a^{\Delta} + b^{\Delta}) \end{cases}$$

These resultant equations can now be mapped into gates to produce physical implementation of the binary phase encoder as shown in Figure 7(c) (signal *go* is added for start/reset purposes).

VI. Conclusions

The paper discusses the benefits of using multiple rail phase encoding protocol and compares it with some other self-synchronous communication protocols. It also presents a CPOG model based approach for specification and synthesis of phase encoding multiple-rail controllers (phase detector, repeater, variety of phase encoders for different source encodings). The obtained solutions are more robust and scalable than in the previously published approaches.

Acknowledgement

This work was supported by EPSRC grant EP/C512812/1.

References

[1] William J. Bainbridge. *Asynchronous System-on-Chip Interconnect*. PhD thesis, University of Manchester, 2000.

[2] Jordi Cortadella, Michael Kishinevsky, Alex Kondratyev, Luciano Lavagno, and Alexandre Yakovlev. *Logic synthesis of asynchronous controllers and interfaces*. Advanced Microelectronics. Springer-Verlag, 2002.

[3] Crescenzo D'Alessandro, Andrey Mokhov, Alex Bystrov, and Alex Yakovlev. Delay/Phase Regeneration Circuits. In *Proc. of International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2007.

[4] Crescenzo D'Alessandro, Delong Shang, Alexandre V. Bystrov, and Alexandre Yakovlev. PSK signalling on NoC buses. In *PATMOS*, pages 286–296. Springer, 2005.

[5] Andrey Mokhov, Crescenzo D'Alessandro, and Alex Yakovlev. Multiple rail phase encoding circuits. Technical report, University of Newcastle upon Tyne, May 2008.

[6] Andrey Mokhov and Alex Yakovlev. Conditional Partial Order Graphs and Dynamically Reconfigurable Control Synthesis. In *Proceedings of Design, Automation and Test in Europe (DATE) Conference*, 2008.

[7] Steven Nowick. *Automatic Synthesis of Burst-Mode Asynchronous Controllers*. PhD thesis, Stanford University, 1993.

[8] Tom Verhoeff. Delay-insensitive codes - an overview. *Distributed Computing*, 3(1):1–8, 1988.