School of Electrical, Electronic & Computer Engineering

UNIVERSITY OF
NEWCASTLE UPON TYNE

Completion Detection Optimisation

A. Mokhov, D. Sokolov and A. Yakovlev

Technical Report Series

NCL-EECE-MSD-TR-2005-109

October 2005

Contact:

Andrey.Mokhov@ncl.ac.uk

Danil.Sokolov@ncl.ac.uk

Alex.Yakovlev@ncl.ac.uk

Supported by EPSRC grant EP/C512812

NCL-EECE-MSD-TR-2005-109

School of Electrical, Electronic & Computer Engineering,
Merz Court,
University of Newcastle upon Tyne,
Newcastle upon Tyne, NE1 7RU, UK

`http://async.org.uk/`

Completion Detection Optimisation

A. Mokhov, D. Sokolov and A. Yakovlev

October 2005

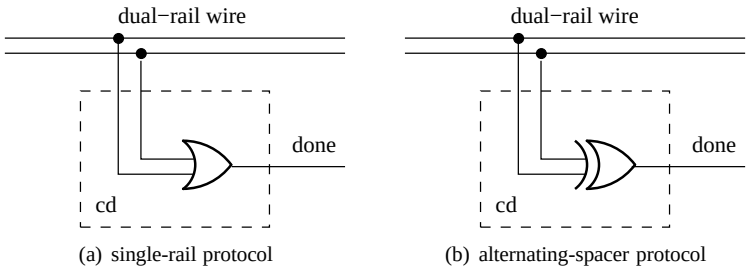
Abstract

This paper presents an algorithm for efficient distribution of completion detection blocks in a dual-rail self-timed circuit to ensure correct computation of the completion signal. Layer-wise optimisation technique is used with the width of layers selected so as to satisfy timing constrains and use the least possible number of completion detection blocks.

1 Introduction

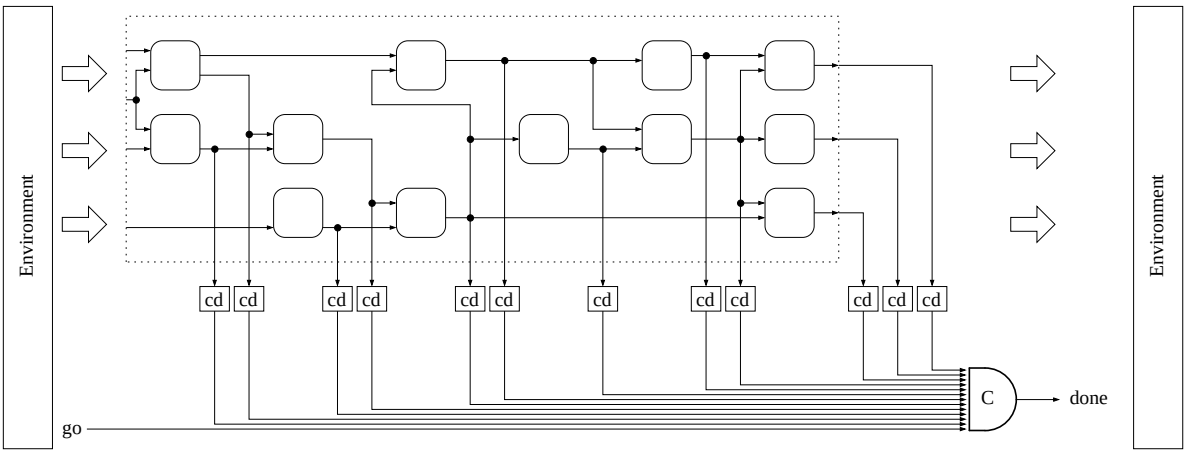
Dual-rail self-timed circuits do not have a clock and require *completion detection* to be used to generate a signal *done* which informs the environment that the circuit has produced correct data on the output and all its internal gates are in stable state. Because of the *early propagation* effect [4] it is not enough only to check that the circuit outputs are ready. The circuit can produce correct output data much earlier than all its internal gates come to a stable state. As a result, the environment can switch circuit inputs from codeword to spacer too early. If the spacer propagates faster than a codeword it can eventually overtake the codeword that leads to hazards in the single-spacer protocol [4] and even to incorrect functionality in the alternating-spacer protocol [4].

A possible implementation of completion detection blocks for the dual-rail logic is shown in Figure 1. As the single-spacer protocol uses all-zeroes combination {00} to represent a spacer and combinations {01} and {10} to encode values 0 and 1 it is possible to use a simple OR gate (see Figure 1(a)) to check if the wire has switched from a spacer to a codeword or vice verse and produce the *done* signal. In case of the alternating-spacer protocol a spacer can also be encoded as combination {11} and hence a XOR gate should be used to distinguish between spacers and codewords as shown in Figure 1(b).

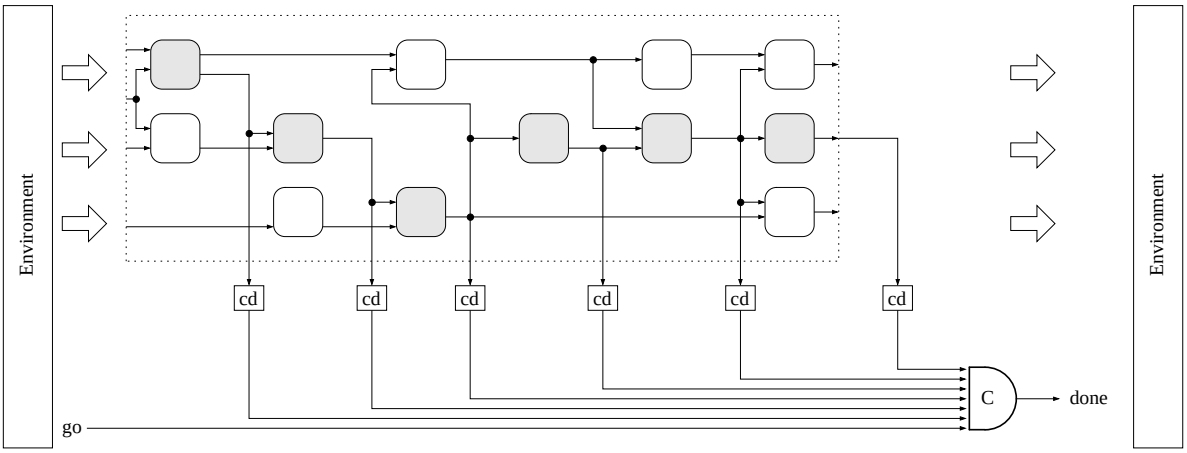


(a) single-rail protocol (b) alternating-spacer protocol

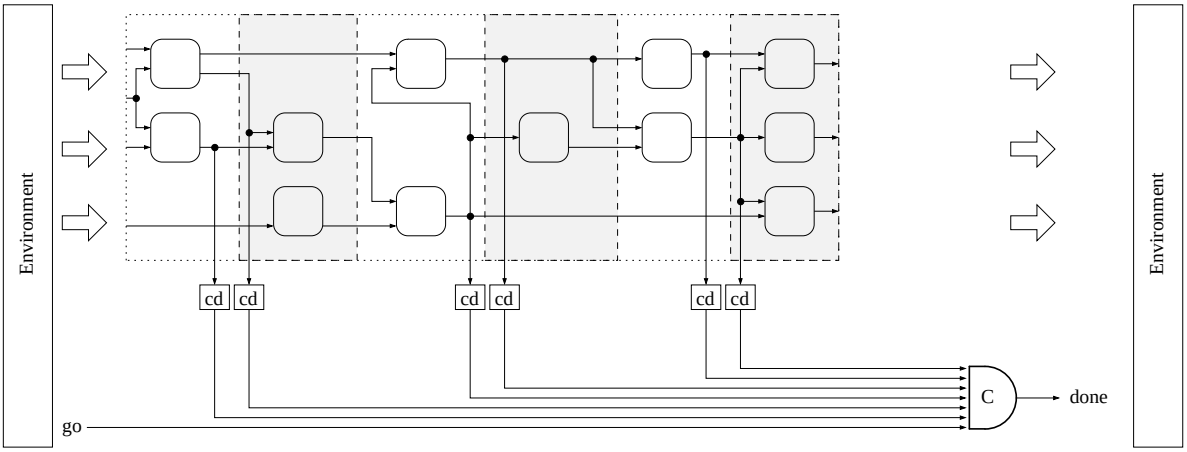
Figure 1: Completion detection blocks



(a) No optimisation



(b) Path-wise optimisation



(c) Layer-wise optimisation

Figure 2: Completion detection optimisation

The obvious way to ensure the correct computation of the completion signal is to check not only the outputs of the circuit but the outputs of all the internal gates. This simple way is very inefficient albeit *speed-independent* [3]. The example circuit in Figure 2(a) needs 12 completion detection blocks using this approach (actually same as the total number of gates in the circuit). Completion detection blocks are marked as *cd*. Efficient implementations of a multi-input C-element that is used to produce the *done* signal can be found in [1]. Note that inputs of the circuit needn't be checked with completion detection blocks as the environment confirms their stable state with the *go* signal.

There are several possible ways to decrease the number of completion detection blocks used for correct computation of the *done* signal [3]. They are based on relative timing constraints, therefore the circuit is no longer speed-independent. One of these methods is *path-wise optimisation*. The method assumes that the longest delay of circuit stabilisation is determined by its critical path. Therefore it is enough to put completion detection blocks only on the outputs of the gates on the critical path. The result of application of this optimisation to the same circuit is shown in Figure 2(b). Only 6 completion detection blocks are used here as the circuit has a critical path composed of six gates (they are marked as gray in the figure). Unfortunately the assumption of the method is incorrect in general case as for some input transitions shorter paths can have longer delay of stabilisation. To increase the reliability of the method it is possible to select several paths for completion control instead of the critical one only. But still this won't guarantee the correct behaviour of completion detection logic without computationally complex dynamic timing analysis [2].

Another method of optimisation, *layer-wise optimisation*, is more reliable and promising. It relies on relative timing of gates in the circuit and skips some layers of gates without completion detection, as shown on Figure 2(c). The layers of gates that were skipped are marked with gray (6 gates were skipped in total). For example, the three gates in the last layer needn't be checked for completion because the multi-input C-element producing *done* signal is very slow in comparison with simple boolean gates and therefore these three gates will stabilise and produce output before the completion signals from the previous gates propagate the through completion detection blocks and the C-element. Same observation is used to skip completion detection in the other two layers. In [3] the width of layers is kept almost constant throughout the circuit but in fact careful timing analysis of given circuit allows the width to be increased from layer to layer and eventually obtain substantial reduction to the number of used completion detection blocks. This paper presents an algorithm for refined layer-wise optimisation that uses layers of width potentially growing in geometric progression.

2 Layer-wise optimisation refinement

To start description of the algorithm two timing functions should be introduced. Let $f(C)$ denote the maximum time for a *circuit* C to stabilise on all possible input transitions. Also let $g(C)$ denote the minimum time for the *circuit* C *outputs* to stabilise on all possible input transitions. Note that $f(C)$ is responsible for stabilisation of all the internal gates of C while $g(C)$ is responsible for outputs of C only. Both functions should take *early propagation effect* into account. The least possible delay of the multi-input C-element will be denoted as Δ_C. Knowledge of the implementation of the multi-input C-element can be used to calculate the delay more accurately for different layers. This paper however uses only conservative lower bound on the delay Δ_C for the C-element. The algorithm needs only minor modifications for any given particular implementation of the multi-input C-element.

Now it is possible to derive the width of the first layer L_0 (layers are counted from the right side of the

circuit). Its width is determined by the following inequality:

$$f(L_0) \leq \Delta_C$$

It is quite easy to understand the reasoning behind: the first layer L_0 has no completion detection blocks attached to it and therefore it has to stabilise completely before the completion signal from the previous layer propagates through the C-element. After observing that we can derive a bit more complex constraint for the second layer L_1:

$$f(L_1) \leq \Delta_C + g(L_1)$$

Here layer L_1 must stabilise not later than its outputs stabilise ($g(L_1)$ term) and completion detection from them passes through the C-element (Δ_C term). Up to now there is no difference between the usual layer-wise optimisation and the new refinement. But the constraints for the next layer L_2 are already different. In the usual layer-wise optimisation we bound the width of L_2 by:

$$f(L_2) \leq \Delta_C + g(L_2),$$

which is similar to layer L_1. But the refinement is that the constraint for L_2 width can be relaxed in the following way:

$$f(L_2) \leq \Delta_C + g(L_1 \cup L_2)$$

The reasoning behind is that layer L_2 must stabilise not later than all outputs of L_1 stabilise ($g(L_1 \cup L_2)$ term) and the completion detection from them passes through C-element (Δ_C term). It is hard to realise now why there should be completion detection blocks after layer L_2 if they are not mentioned in *any* constraint at all. The reason is that these blocks are used not for correct computation of completion signal for layer L_2 but for correct computation of that for L_1. When stating the constraint for layer L_1 it was assumed that its inputs will change simultaneously. But that is not true if the layer is not the leftmost in the circuit. Completion detection blocks standing before the layer implicitly solve this problem. They guarantee correctness of either completion signal from L_1 or completion signal from L_2.

The general inequality for layer L_n can be now easily derived by induction:

$$f(L_n) \leq \Delta_C + g(\bigcup_{k=1}^{n} L_k)$$

The above inequality implies that the width of layer L_n is greater than the width of L_{n-1}. The exact growth factor is determined by particular circuit but it can be estimated to be greater than one. That can optimise the number of used completion detection blocks significantly. Figure 3 shows the result of application of the refined layer-wise optimisation to the example circuit. Only 4 completion detection blocks are used here that is the best result obtained with different optimisation techniques.

Algorithm for the refined layer-wise optimisation (shown in Algorithm 1) can be easily written to have $O(H \cdot n)$ complexity where n is the number of gates in the circuit and H is its average *height*, i.e. the average number of gates that can be attached to a cut of the circuit without violation of the topological order of gates in the circuit. In the example circuit H is equal to 3 approximately. Note that the algorithm uses conservative estimations of functions f and g as their exact evaluation is too time consuming and practically intractable. The estimations used are based on finding critical paths.

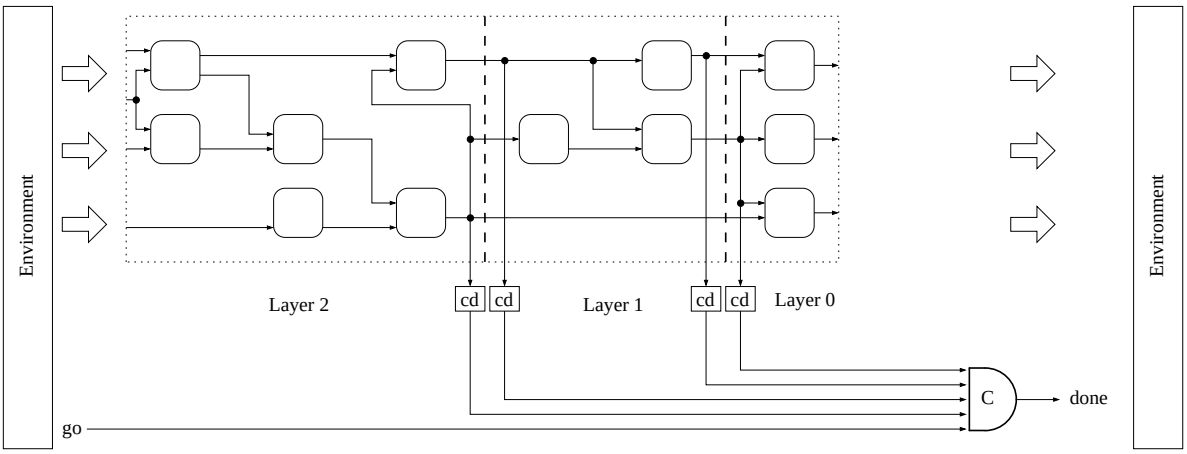


Figure 3: Refined layer-wise optimisation

The algorithm works on dual-rail circuits and therefore operates on *dual pairs of gates*. A dual gate is a pair of gates in the dual-rail circuit whose outputs form a dual wire.

The algorithm is iterative. In each iteration a dual pair is added to the partially constructed circuit. The pair to add is selected from candidates that can be added to the partially constructed circuit without violation of the topological order of gates in the circuit. Among these pairs the one that minimises $f(L_n) - g(\bigcup_{k=1}^{n} L_k)$ is selected. If no dual pair of gates can be added without violation of constraint $f(L_n) \leq \Delta_C + g(\bigcup_{k=1}^{n} L_k)$ the current layer is considered finished and new layer starts. This iterative technique allows the estimations of functions f and g to be obtained in constant time for every gate pair added to the layer, thereby reducing time complexity.

Algorithm 1 Refined layer-wise optimisation

```
Given : G - set of gates of the circuit;
Result: L - set of layers for completion detection;

n=0; // current layer
while(G ≠ Ø)
{
  Let Q = ⋃_{k=1}^{n} L_k;
  <Select pair (x,y) of dual gates such that:>
    1) x ∈ G and y ∈ G;
    2) all outputs of x and y are in Q ∪ L_0;
    3) f(x ∪ y ∪ L_n) − g(x ∪ y ∪ Q) is minimised.

  if f(x ∪ y ∪ L_n) ≤ Δ_C + g(x ∪ y ∪ Q) then
  {
    <Add x and y to current layer L_n>;
    <Delete x and y from G>.
  }
  else
  {
    n++; // Layer L_n is finished.
  }
}
```

3 Conclusions

A refinement to the layer-wise optimisation technique of [3] is presented. Theoretical results appear to be promising but their practical use is to be determined yet. The key feature of the technique is that it is based on relative timing characteristics of circuit paths and makes use of delays in signal propagation through layers and a multi-input C-element.

The presented algorithm for layer-wise optimisation is quite fast that allows it to be used for large circuits. It is not linear though with respect to circuit size and further optimisations are most probably possible.

Acknowledgement

This work is supported by EPSRC project NEGUS at the University of Newcastle upon Tyne. Thanks go to Dr. Alexandre Bystrov for useful discussions.

References

[1] Fu-Chiung Cheng. Practical design and performance evaluation of completion detection circuits. In *IEEE International Conference on Computer Design (ICCD)*, 1998.

[2] Andrey Mokhov. Software development for dynamic timing analysis of monotonic combinatorial circuits. Master's thesis, Kyrgyz-Russian Slavic University, 2005.

[3] Danil Sokolov. *Automated synthesis of asynchronous circuits using direct mapping for control and data paths*. PhD thesis, University of Newcastle upon Tyne, 2005.

[4] Danil Sokolov, Julian Murphy, Alexander Bystrov, and Alex Yakovlev. Design and analysis of dual-rail circuits for security applications. *IEEE Transactions on Computers*, 54:449 – 460, 2005.