Synchronous to Asynchronous Conversion

A Case Study: The Blowfish Algorithm Implementation

J. Alcântara*, S. Salomão*^o, E. Granja*, V. Alves*, F. França*

*COPPE – Federal University of Rio de Janeiro, POBox 68504
21945-970 Rio, Brazil

^oIME – Military Institute of Engineering,
Praça General Tibúrcio, 80
22290-270 Rio, Brazil

Key words: Asynchronous logic, cryptography, VLSI.

Abstract: This work introduces a novel methodology that eases the synchronous to asynchronous conversion of existing digital circuits. Synchronous single-phased circuits may have its performance improved with the use of a variable rate clock generator if the conversion is done on some key circuits. This methodology is used to improve the performance of a soft-core implementation of the Blowfish cryptographic algorithm.

1. INTRODUCTION

This work introduces the use of a distributed and deterministic mechanism on the asynchronous timing of selected digital networks. Our approach shows that any digital system can be redesigned as a synchronous system with a varying clock rate. Hardware overhead is small and will not add significantly to the real state needs of the original system.

The conversion methodology is based on Scheduling by Edge Reversal (SER), a distributed algorithm^[i] ^[iii] that is a potentially optimal solution to Dijkstra's *Dining Philosophers* problem^[iii] under heavy load. SER gives a fair and distributed solution to this problem and has been used in the implementation of many distributed processing paradigms.

All information exchanged between any two distinct neighboring digital subcircuits must then be treated in a *mutual exclusion* style. SER can be used to control the asynchronous timing among parts of a target digital system

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35498-9_57](https://doi.org/10.1007/978-0-387-35498-9_57)

and can be seen as a generalization of the handshake protocol.^[iv] SER is not topology constrained.^[v] Theoretical background for the distributed mechanism behavior is introduced in section 2. Section 3 presents the conversion methodology. Section 4 describes the Blowfish algorithm. Section 5 details the implementation and the performance achieved. Concluding remarks follows in section 6.

2. SCHEDULING BY EDGE REVERSAL – SER

Relevant aspects of the SER methodology are shown in this section. For an in-depth review of the theory involved the reader is referred to^{[vi] [vii] [viii]}. The SER methodology^[ix] is composed of three basic steps.

1. Representing the target digital circuit. Consider a neighborhood-constrained system composed of a set of processes and a set of shared atomic resources represented by a connected graph $G = (N, E)$ where N is the set of processes and E , the set of edges defining the interconnection topology. Each process is represented by one node. An edge is present between any two nodes if and only if these two processes share at least one resource.

Figure 1. A graph G under SER, with $m = 1$ and $p = 3$

Figure 1 shows an example of G . SER works in the following way: starting from any acyclic orientation ω on G there is at least one *sink* node, i.e., a node that has all of its edges directed to itself. All sinks are allowed to operate while other nodes remain idle. After operation, a sink node will revert the orientation of its edges, becoming a *source*, and thus releasing resources to its neighbors. The whole process is then repeated for the new set of sinks. Note that the reversible edges in the definition graph G do not imply any data flow, they simply sign resource availability.

2. Construction of the asynchronous timing network. Let $G = (N, E)$ represent the target SER network defined to drive the timing of the target circuit C . Every node $N_i \in N$, is connected to N_{ipost} , where $N_{ipost} = N(i+1) \bmod N$, if and only if the corresponding elements are neighbors.

3. Initialization of G . An appropriate acyclic orientation on G can be obtained if an edge between any two neighboring nodes $N_i, N_j \in N$ is oriented from N_i to N_j if $i > j$. Using this particular orientation node 0 is the first *sink* to happen under SER.

3. CONVERSION METHODOLOGY

In a synchronous digital system any connection topology may be defined, so that stages include but are not limited to pipelines. *Figure 2* shows an example of a possible target circuit composed of four functional stages, and its data communication. Stages having no interaction between the elements of the output vector have an algorithmic complexity on time of $O1$ and may include decoders, multiplexers and random logic. For circuits performing arithmetic operations, such as adders and multipliers, the mean complexity on time will be different from the worst case. An n -bit ripple carry adder, for instance, has a worst case complexity of On while having a mean complexity of only $O(\log_2 n)$.^[x]

Figure 2. Synchronous stages and data communication of the target circuit

A great part of the system can then be treated just like any synchronous system and just a few stages will need to be redesigned. The result will be a faster clock for the majority of operations.

Definition. Synchronous to Asynchronous Conversion methodology.

(i) *Representation of the target digital circuit C.* The target digital circuit is represented by a graph $C=(B,D)$, where B is the set of its synchronous stages and D is the set of edges used to time their operation. A synchronizing node S is then added to the circuit as shown in *Figure 3*.

(ii) *Construction of the asynchronous timing network.* Let $G=(N,E)$ represent the target self-oscillatory SER network defined to drive the timing of the target circuit C . Nodes NI_j control the length of the *high* clock level. For every stage in the circuit there is an edge between nodes $N0$ and NI_j . As each stage ends its operation, it reverses its edge towards $N0$. When all the stages are done, $N0$ becomes a sink and produces a clock transition from *high* to *low*. Then it operates for a very short time and prepares for a new cycle by signaling all the stages with an edge reversal.

(iii) *Reduction of the asynchronous timing network.* In order to save circuitry, and also to simplify the conversion process, only a few stages need to be asynchronously timed. Stages with a complexity $O1$ have a nearly constant t_d , no matter how its input vary. In this case, the stage i , for which $t_{dmin_i} > t_{dmax_j}$ for all $j \neq i$, can be used to time itself and all

the j stages. Other stages, according to the instantaneous values of their inputs, will have varying t_d . Those stages must be timed separately and ideally they will be the candidates for an asynchronous redesign.

Figure 3. The target circuit as $C=(B,D)$ and as $G=(N,E)$

(iv) *Initialization of G .* An appropriate acyclic orientation on G is easy to be obtained if we define $N0$ as the initial state. All the edges are initialized pointing to $N0$, which becomes a sink.

SER Hardware Implementation. The implementation comprises two basic circuits, namely the *Node Controller* and the *Edge Controller*, both operating on self-timed functional blocks. Note that each token represents the orientation of each edge on G .

Figure 4. Schematic for the Node and Edge Controllers

Node Controllers simply evaluate the presence of tokens and, after detecting that the node has become a *sink*, they allow the target local circuit to operate. This implementation relies on a transition sensitive *end of operation* signal that is sent to all the neighboring Edge Controllers. Every time the node controller for nodes Nl_i is activated, all the stages start their operation. Node $N0$ is simply a synchronizer. As shown in Figure 4, a node starts operation according to an AND gate that samples the incident tokens and as many inputs as the number of arrowheads pointing to the node.

Edge Controllers shown in Figure 4 connect two nodes using a differential output XOR gate as a token generator. They also contain logic used to guarantee the initial acyclic orientation. Each Edge Controller will reset the token for the node that ended its operation and at the same time will send that token to a neighbor node.

4. CASE STUDY

Blowfish Cryptographic Algorithm. Blowfish^{[xii] [xi]} is a symmetric variable-length key, 64-bit block cipher cryptographic algorithm based on Feistel Networks^[xii] developed by Bruce Schneier^{[xii][xi]}. Data encryption is performed through a 16-round Feistel network. Each round consists of a key-dependent permutation, and a key- and data-dependent substitution. All operations are done over 32-bit words.

Subkeys: These keys must be pre-computed^{[xii][xi]}. They consist of a P-array with 18x32 subkeys and four S-boxes with 256x32 entries each.

Encryption: As stated previously, the Blowfish is a 16 round Feistel network, as seen in Figure 5(a). The input is a 64-bit data element, X.

Figure 5. The structure of the Blowfish algorithm

The algorithm can be stated as:

Divide x into two 32-bit halves: X_L, X_R

For $i = 1$ to 16: $X_L = X_L \oplus P_i$; $X_R = F(X_L) \oplus X_R$; Swap X_L and X_R ; end

$X_R = X_R \oplus P_{17}$; $X_L = X_L \oplus P_{18}$; Recombine X_L and X_R

Function F , see Figure 5(b): Divide x_L into four eight-bit quarters: a, b, c , and d . Then $F(X_L) = ((S_a^1 + S_b^2 \bmod 2^{32}) \oplus S_c^3) + S_d^4 \bmod 2^{32}$

Decryption: Same algorithm, with $P_1, P_2, \dots, P_{18}$ in the reverse order.

Implementations of Blowfish that require the fastest speeds should unroll the loop and ensure that all subkeys are stored in embedded RAMs^{[xii][xi]}. Examining one round in the data flow shown in Figure 5, one can identify groups of data independent operations, pointing out temporal and spatial parallelism implemented as one round pipeline shown in Figure 6. FIFOs included into the pipeline hold data forwarded between non-adjacent stages.

S2A Conversion. The first step in the conversion methodology consists of circuit partitioning. An **S2A** system (*Figure 7a*), with 4 nodes corresponding to nodes $N1_i$ and the extra $N0$ node was created. A single timing node $N3$ corresponds to the circuit blocks having a nearly constant delay. The delay corresponding to node $N3$ has been determined from critical path extraction. Nodes $N1$ and $N2$ correspond to asynchronous adders **ADD1** and **ADD2** respectively.

Figure 6. The pipeline structure

The clock signal is derived from node $N0$ and indicates that nodes $N1$, $N2$ and $N3$ have completed their respective task. This varying clock signal will time the pipeline described in *Figure 6*. The clock period for the converted system will vary from a minimum corresponding to the operation of $N3$ to the adder worst case just described. In fact, edge reversal is not instantaneous, but it imposes just a very low overhead to the cycle time.

Rules and constraints. S2A conversion methodology is very simple and powerful. However, some care must be taken regarding hardware implementation: Blocks with variable operating time have to be redesigned in order to provide an end of operation signal and end of operation signals must be free of glitches. Also, circuit outputs must be stable after end of operation until the end of the cycle. Thus, depending on the specific circuit implementation, it may be necessary to latch its outputs.

Figure 7. Asynchronous conversion

5. IMPLEMENTATION AND PERFORMANCE

Circuit description. Adder units were designed from 32_bit ripple carry adders. There are four 32_bit pipeline registers. The subkeys are stored in 6 embedded RAMs. Scheduling by Edge Reversal is used for correct circuit timing is depicted in *Figure 7*. Edges A1 to A3 are simply implemented with a XOR gate and an inverter. DELAY1, used in stages 2 and 4, represents the worst case timing for the S-boxes, while DELAY2 is the worst case obtained from critical path extraction for the remaining pipeline stages.

Table 1. Performance comparison

	Synchronous version	Asynchronous version
Number of transistors	43280	48771
Minimum cycle time	-	13ns
Average cycle time	20ns	16ns
Maximum cycle time	-	25ns
Average throughput	200Mbps	250Mbps

Performance. The asynchronous circuit obtained from S2A conversion was described in VHDL, synthesised with a 0.7µm CMOS standard cell library and simulated with Synopsys CAD tool. Variable clock generation and SER operation are shown in *Figure 8*. The circuit was validated by comparison with a C program executing the Blowfish algorithm. *Table 1* compares the performance of synchronous and asynchronous versions.

6. CONCLUSIONS

We have presented an alternative approach to the conversion of synchronous circuits into asynchronous. The use of SER's methodology simplified the conversion, as it allowed a viable implementation of a variable rate clock generator requiring very low area overhead. A pipelined implementation of the Blowfish cryptographic algorithm, its VHDL description and the resulting asynchronous circuit were simulated and synthesized, showing a reasonable performance improvement.

Figure 8. Simulation results

7. REFERENCES

- [i] Barbosa, V., Gafni, E. "Concurrency in heavily loaded neighborhood-constrained systems". *ACM Transactions on Prog. Languages and Systems*, vol. 11, no. 4, Oct 1989.
- [ii] Barbosa, V. C., "An Introduction to Distributed Algorithms", MIT Press, 1996.
- [iii] Dijkstra, E. "Cooperating Sequential Processes" In *Programming Languages*, F. Genuys, Academic Press, New York, 1968, pp 43-112.
- [iv] Meng, T. "Synchronization design for digital systems" Kluwer Academic, 1991.
- [v] Jou, S. and Chuang I. "Low-Power Globally Asynchronous Locally Synchronous Design Using Self-Timed Circuit Technology" *IEEE Int'l Symp. on Circuits and Systems*. Hong Kong, June 1997.
- [vi] França F., Alves V., Granja E. – *Edge Reversal Based Asynchronous Timing Scheme* – International Symposium on Circuits and Systems, Monterey, USA, 1998.
- [vii] França F., Alves V., Granja E. – *A Multi-phase Asynchronous Timing Scheme* – SBCCI, Gramado, Brazil, 1997.
- [viii] Alves V., França F., Granja E. – *A BIST Scheme for Asynchronous Logic* – Asian Test Symposium, Singapore, 1998.
- [ix] França, F., Alves, V., Granja, E.- *Patent Req INPI BR*.
- [x] Franklin, M., Pan T. "Performance Comparison of Asynchronous Adders" in *Int'l Symp. on Advanced Research in Asynchronous Circuits and Systems*. November 1994.
- [xi] Schneier, B.- Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish) *Cambridge Security Workshop Proceedings*, Springer-Verlag, Dec 1993.
- [xii] Schneier, B. – *Applied Cryptography* – John Wiley and Sons, New York, 1996.