A Design Flow for 12.8 GBit/s Triple DES using Dynamic Logic and Standard Synthesis Tools

S. Flügel, F. Grassert, M. Grothmann, M. Haase, P. Nimsch,
H. Ploog, D. Timmermann, A. Wassatsch

University of Rostock, Dep. ETIT
Institute of Applied Microelectronics and CS
Richard-Wagner-Str. 31; 18119 Rostock, Germany

{gf94,hp,wa11}@e-technik.uni-rostock.de

ABSTRACT

In this paper, we propose a 12.8Gbit/s Triple DES processor using 0.6um/5V AMS. A conventional static CMOS implementation cannot fulfill the requirements of high throughput digital designs. In contrast, dynamic logic has the principle advantage of speeding-up logic designs. In particular True Single Phase Clock (TSPC) Logic shows a robust cell characteristic, and therefore a standard cell implementation using TSPC is possible. Current high level synthesis tools do not support automatic synthesis of dynamic logic, therefore this implementation also utilizes a modified design flow and uses a toolset DYNAMIC, which applies a transformation of the combinational circuit into a pipelined structure. Due to these aggressive changes in the design representation, verification in each design step is strictly required. In addition, the design was verified against the FIPS 46.3 reference model implemented with the use of CLI. Furthermore, we present the results of the design orientated library optimization process.

Utilized Synopsys Tools: Design Compiler, Library Compiler, CLI-Interface, VSS, Cyclon, Formality

1. Introduction

This paper describes the development of a 12.8GBit/s Triple Data Encryption Standard (DES) processor. Fast data encryption is becoming a more important requirement for applications such as secure networking. Using DES cannot always ensure high security, therefore the data can be encrypted three times with the same algorithm to realize Triple DES. Through the utilization of dynamic TSPC, a fast realization can be achieved. The synthesis of dynamic logic is difficult, because there are no synthesis tools which support such logic styles.
In this paper, the main focus is on the design flow of the DES chip. It discusses the basics for encryption and dynamic logic and explains the performance goals and the resulting chip structure briefly. Further, the single steps for the design of the processor are introduced trying to keep the relationship between the syntheses and the verification. We conclude with a discussion of the results and estimate comparisons with other chips.

2. Fundamentals

The following two sections give a basic understanding of the DES encryption principle and the main concepts of dynamic logic.

2.1. DES encryption

DES [8] is a symmetric (private-key) encryption. To be used, both parties must agree on a secret key which must be kept private. The algorithm processes a message of 64 bits and encrypts this with a 56 bit key to a 64 bit cyphertext. The incoming data and key are passed through input and output permutations. Between these permutations, the data is passed 16 times through a Feistel network, while 16 keys are generated simultaneously through permutations on the base key.
The data is split into two parts, L and R. R_{i-1} is modified by the function f and XORed again afterwards with L_{i-1} to build the new R_i. R_{i-1} is directly passed through to L_i. Figure 1 shows one round of the DES algorithm. During the f-function data is expanded and XORed with the sub-key k_i. The received value is fed through eight S-boxes, which are simple table lookups. In this design, this loop was unrolled, and therefore it is ideal for an implementation as a pipeline.
Decryption is done using the same algorithm as encryption except using a reverse key schedule.
It is widely known that the standard DES algorithm was broken with the help of the internet several months ago. Therefore, this chip realizes a triple DES. In triple DES, data is encrypted with the first key, decrypted with a second key, and then encrypted again with a third key.

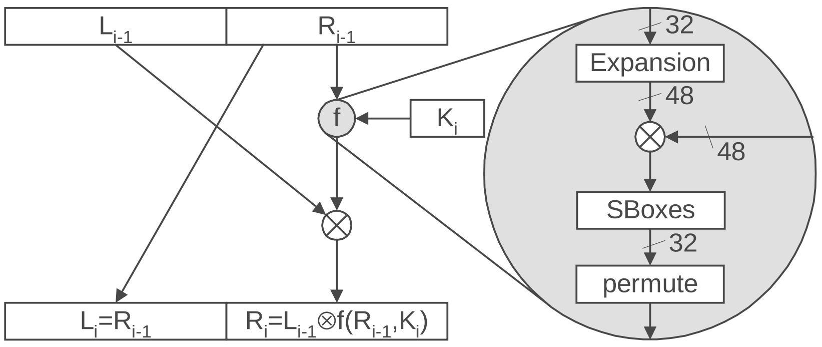

Figure 1: One round of DES

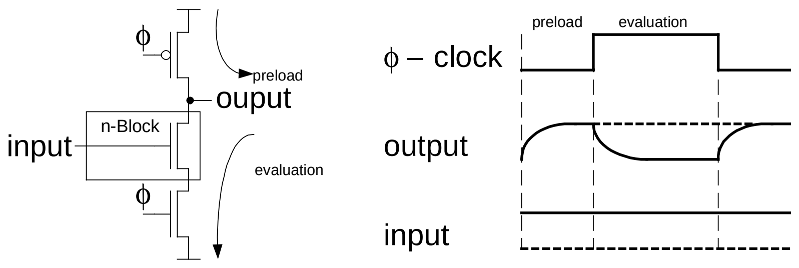

Figure 2: Function of dynamic logic with precharge and evaluation period

2.2. Dynamic Logic

The use of dynamic logic is state-of-the-art for high speed data processing applications. For the realization of circuits like the Compaq Alpha microprocessor [1] or the experimental IBM PowerPC integer microprocessor, this circuit style was used extensively to achieve frequencies approacing 1GHz. In addition, pipelining is used to speed up the circuits.

In static CMOS, the output stays stable as long as the input signals do not change. A logic function consists of complementary blocks of N- and P-transistors. Either the N-network connects the output node to ground or the P-network connects the output to VDD. A static CMOS circuit doesn't need a clock signal, the functionality is always given, and the power consumption is determined by the number of changes of output nodes.

Dynamic logic styles differ from static CMOS logic because they work with two phases: the evaluation and the precharge phase. This is illustrated in Figure 2. Dynamic logic always needs a clocking signal, because the output nodes have to be precharged in every cycle. There are two possibilities in precharging: the output node is precharged high through a P-transistor or the output is precharged low through an N-transistor. In the evaluation period the output settles to the correct value depending on the input signals by connecting the output node to ground through an N-network (precharged high) or connecting the output to VDD through a P-network. After the loss of the charge from the output node, there is no possibility for it to be reestablished. Therefore, false discharging has to be prevented by changing the input signals only from low to high for an N-block and vice versa for a P-block during the evaluation. The power consumption does not mainly depend on the change of the input signals, because the output node can be charged and discharged every clock cycle even when the input signals stay stable.

A special dynamic circuit is the TSPC structure presented in [5]. As Figure 3 shows an N-block and an N-latch alternate with a P-block and a P-latch. This regular structure makes this logic

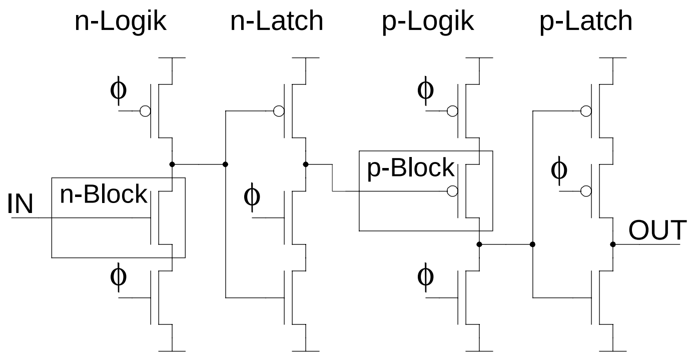

Figure 3: Structure of True Single Phase Clock Logic (TSPC)

style easier to use and prevents false discharging of the output nodes. The logical functions are mostly implemented in the N-block because of the higher speed. The N-part of TSPC works inversely to the P-part, so that the N-blocks are in the precharge phase while the P-blocks are in the evaluation period and vice versa. Therefore, this structure implements a logic and a register function at the same time.
The advantages of dynamic logic are less area, because the logic function is realized in N-transistors or P-transistors only rather than with complementary networks as for static CMOS, and higher speed than static CMOS (approx. 1.5), because the N-logic is faster in evaluation. On the other hand, dynamic logic always needs a clocking signal and has a higher power consumption than static CMOS. Also, there are no synthesis tools that facilitate the use of dynamic logic. But due to the uniform structure of TSPC-logic, the partial use of conventional tools is possible through a special design flow with the use of an additional external toolset (DYNAMIC) [2,3].

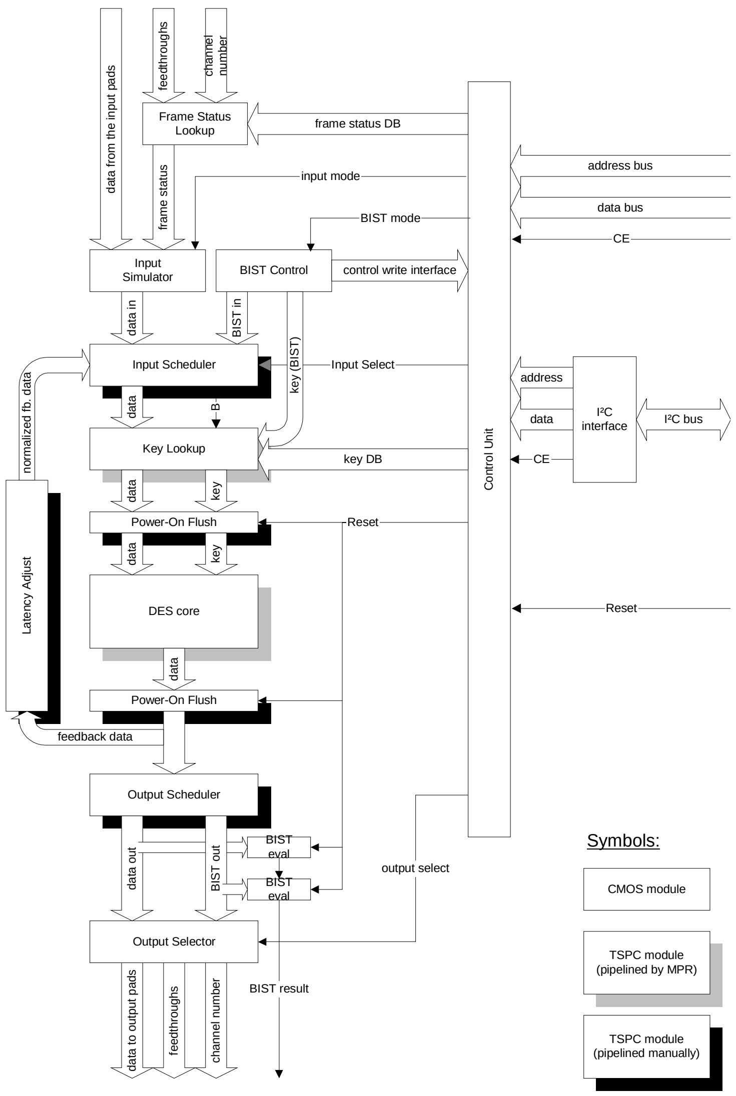

Figure 4: Structure of the Triple DES chip

3. Chip Specification

3.1. Performance of the DES-Chip

The development of the DES chip was driven by its functionality, the requirements of which are discussed briefly. Two working modes of encryption are implemented: the simple straight forward DES encryption and the triple DES mode. The chip also has a ROM based Built In Self Test (BIST) unit. Although the functionality of the device includes a cipher feedback (CFB) mode, this is not implemented in the layout due to the area overhead that it incurs.

The triple DES encryption works for 64 Bit parallel at a clock frequency of 200 MHz and the throughput is 12.8Gbit per second. Because the pads of this evaluation chip are limited to 200 MHz an increase of the internal clock frequency to 800 MHz is necessary. The increase of the clock allows a single DES encryption to be looped three times to achieve triple DES encryption. The time for the fourth slot is used for functional chip monitoring and permits additional research of this chip.

3.2. Structure of the DES-Chip

Figure 4 illustrates the logical structure of the DES chip whose implementation meets the required goals of performance and functionality. The device can be conceptually divided into two sections on the basis of performance: the control interface and self test parts which operate at 200MHz, and the DES core along with the scheduler, which operate at 800MHz. These two sections are implemented with different logic styles, the fast section using dynamic logic and the slow section using static CMOS.

In addition the device can be further subdivided according to synthesis requirements. The mask ROM contains data necessary for the BIST. The static CMOS section includes the control unit, parallel interface, PC interface and input selection. The remaining two sections constitute the dynamic logic. Of these the DES core synthesis can be automated, however the input/output scheduler must be synthesized manually due to its special timing behavior. Since this last part is small the cost of hand coding is acceptable.

4. Design Flow

The design flow (Figure 5) differs for each of the four parts and is adapted to suit the logic style. Nearly every synthesis step has an associated verification step to ensure the correct functionality. The following sections 4.1 and 4.2 describe Synthesis and Verification respectively, with cross references where appropriate.

4.1. Synthesis

The most time consuming parts for synthesis are the dynamic ones, because they require several steps: library development, creation of a combinational netlist, reorganization to a pipeline structure and verification steps. This automatic pipeline realization is only possible for circuits with no recursive logic. Where recursive logic exists, it needs to be separated from the combinational logic or, alternatively, it needs to be manually synthesized.

Development of a design library

For the dynamic parts, all necessary gates and functions have to be included in a new Synopsys [6] synthesis library. The library compiler was used to create a library containing the simulated results as timing values, structure, and logical function. The number of included different gates influences the length of the resulting pipeline. An expansion of the library reduced the number of

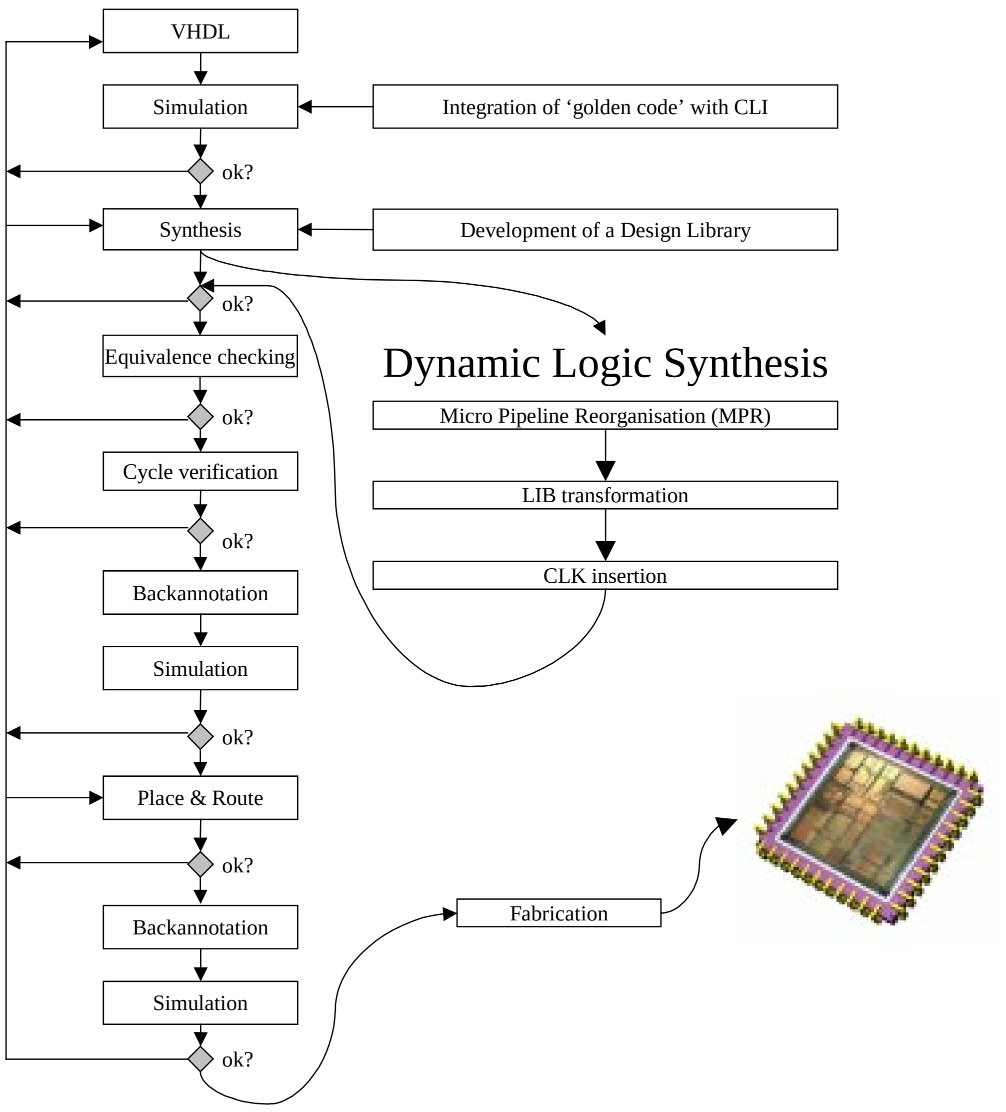

Figure 5: Design-Flow for the evaluation of the chip

pipeline stages from over 500 to 243. Now, the compilation of combinational logic and the verification is possible. At this point, the included register functionality of every gate is not important and the clocking cannot be simulated.
For the static CMOS parts, a standard library (AMS 0,6u cua [7]) can be used.

Design of the static CMOS parts

A standard library can be used to synthesize the static CMOS parts. The compilation of the VHDL description and the verification can be done using the design compiler and VHDLSim making this an easy step, because no changes to the design flow were made.

Automatic design of dynamic pipeline parts

The automatic synthesis of the dynamic pipeline parts requires some additional steps. In Figure 6, the strategy for a small example is shown. The basic VHDL description includes the combinational functionality only and the verification of this description is easy. Next, the compilation of these parts with the design compiler using the developed library with

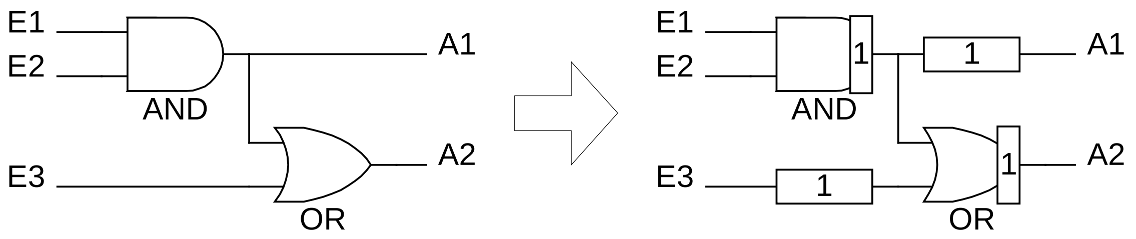

Figure 6: Example of the conversion of combinational logic into a pipeline

corresponding information can be made. At this point, a netlist exists with combinational logic only and without any clocking signal (left side of figure 4). This netlist has to be processed with the Micro Pipeline Reorganization (MPR) tool. Because every TSPC gate includes a register function, the tool can handle each logic gate as a pipeline stage. It replaces the combinational gates with equivalent TSPC gates. Then it includes simple registers in single wires to ensure the timing behavior, because each signal has to be registered in every clock cycle (right side of figure 4). Further, the pipeline tool generates a clock tree and inserts the buffers and connections to the gates. It finally generates a netlist for the fully pipelined logic with clock tree and references to the TSPC library.

The condition for the use of the pipeline tool is the existence of a combinational netlist without any recursive connections. This means that no combinational feedback may exist, because the tools need a well defined start point and end point for each signal path. Where a combinational loop can be split into a forward part and a feedback part, the forward part can also be processed with the MPR tool. After this step, the verification of behavior and cycle functionality can follow.

Hand design of the dynamic parts with recursive logic

For dynamic parts with recursive logic the automatic synthesis steps do not work, which is why this has to be performed in schematic VHDL. The scheduler was hand coded, too, because its function is strictly associated with timing information. A separation of combinational logic and register function is not possible. The power-on flush and the latency-adjust is too simple to use the MPR tool. In addition to the design of the logic, a clock tree has to be built. After that, the following verification step is important to ensure the correct functionality.

4.2. Verification

Verification of behavior with VHDLSim

The behavioral verification of the VHDL descriptions can be realized using VHDLSim. Parts or the whole design can be stimulated with special input values and the response is compared with expected values. For single parts of the chip, this strategy is acceptable. For large sections, the generation of test vectors and the corresponding results becomes very time consuming.

Usage of the C Language Interface (CLI)

The functionality of the DES core is fully programmed in C – the ‘golden code’. The CLI was used to integrate the reference of the DES core into the verification step. The C code can then be used to verify the simulated VHDL description. Both, the compiled C program and the VHDL DES core, were stimulated with the same test vectors and the output signals were compared.

Because the amount of input data is too big to test all combinations, a special group of test vectors which provides high functional coverage was taken. The simulation and comparison process between the developed VHDL core and the reference C software can run automatically and provides a comprehensive error log.

Cycle Verification with Cyclon

A behavioral and a special cycle verification are necessary for the pipelined parts. In this step, the output values of the registers at every cycle are the point of interest. Because of the great number of registers, the pure cycle based verification approach of Cyclon has a better performance than the event driven VHDLSim.

Usage of Formality for verification of the dynamic parts

The structure of the dynamic pipeline parts were verified with the Formality tool. The corresponding static logic can be used with inserted registers to build a graph for comparison. The evaluation of both, the created graph for the pipeline and the graph from the static logic, shows the equivalence of the correct combinational structure and the generated pipeline.

5. Conclusions

The throughput of the developed DES chip of 12.8Gbit per second is very high. Comparable results are described in [4]. However the chip area is very large due to the high number (243) of pipeline stages for the DES core. All dynamic gates have a maximum of only two inputs. This fact minimizes the library but makes the design very inefficient in terms of area. Using a library with optimized gate models would significantly reduce the area.

6. Acknowledgments

This paper has been greatly improved by the SNUG technical committee member Robert Fairlie.

7. References

[1] P. E. Gronowski, W. J. Bowhill, R. P. Preston, M. K. Gowan, and R. L. Allmon, „High-performance microprocessor design”, Journal of Solid State Circuits, Vol. 33, No. 5, pp. 676-686, May 1998.

[2] A. Wassatsch, D. Timmermann, „DYNAMIC - A Java Based Toolset For Integrating Dynamic Logic Circuits Into A Standard VLSI Design Flow”, International Cadence User Group Conference (ICU'2000), San Jose (CA) USA, pp. SIG IC - ic6, September 2000.

[3] A. Wassatsch, D. Timmermann, „Untersuchung zum Einfluß der speziellen Anforderungen dynamischer Schaltungstechnik auf den Systementwurf”, ITG/GI/GMM Workshop: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, Frankfurt/Main(Germany), VDE-Verlag, S. 278-287, 28.2.-1.3. 2000.

[4] I. Kim, C. S. Steele, J. G. Koller, „A Fully Pipelined, 700MByte/s DES Encryption Core”, Proceedings of the 9th IEEE Great Lakes Symposium on VLSI, March 4-6, 1999.

[5] J. Yuan, I. Karlsson and C. Svensson, „A True Single Phase Clock Dynamic CMOS Circuit Technique”, IEEE Journal of Solid-State Circuits, Vol. SC-22, 1987, pp. 899-901.

[6] Synopsys, Inc., 700 East Middlefield Road, CA 94043-4022 United States of America. Synopsys Synthesis and Simulation Tools, 2000 edition.

[7] Austria Micro Systeme International AG (AMS), Austria, 0.6 um CMOS process cua, v3.12.

[8] National Bureau of Standards FIPS Publication 46, „DES modes of operation”, 1977.