Modelling the Crypto-Processor from Design to Synthesis

W.P. Choi and L.M. Cheng

Department of Electronic Engineering, City University of Hong Kong,
Tat Chee Avenue, Kowloon,
Hong Kong S.A.R., P.R.C.
wpchoi@ee.cityu.edu.hk, eelcheng@cityu.edu.hk

Abstract. In this paper, the modelling of a Crypto-processor in a FPGA chip based on the Rapid Prototyping of Application Specific Signal Processors (RASSP) design concept is described. By using this concept, the modelling is carried out in a structural manner from the design capture in VHDL code to design synthesis in FPGA prototype. Through this process, the turnaround time of the design cycle is reduced by above 50% compare to normal design cycle. This paper also emphasises on the crypto-processor architecture for space and speed trade-off; design methodology for design insertion and modification; and design automation from virtual prototyping to real hardware. In which above 60% of spatial and 75% of timing reduction is reported in this paper.

1 Introduction

The design flow and the techniques of modelling a crypto-processor [1] in FPGA chip based on the RASSP [2,3,4,5] are described in this paper. The modelling is made use of the VHDL platform. This platform has provided the perfect simulation and synthesis media for rapid prototyping. As well as, it also facilitated the design methodology of RASSP which promoting the design upgrades and re-uses. The modelled crypto-processor is designed for use in embedded digital systems which requiring area/speed/power trade-off, as crypto-processor is now commonly used in nowadays' digital devices, such as in Electronic Fund Transfer (EFT) systems and electronics wallet using smart cards.

This paper highlighted the procedures of modelling the crypto-processor from design to synthesis as in the following sections. In section 1.1 & 1.2, the background of the RASSP and the modelled crypto-processors are introduced. In section 2, the design process based on the VHDL virtual prototyping is described from the design specification, executable specification to detailed design. In section 3, the detailed design methodology of the crypto-processors is demonstrated. In section 4, the observations and results of this study are reported. In section 5, conclusions are made.

© Springer-Verlag Berlin Heidelberg 1999

1.1 Scope of RASSP

Rapid Prototyping of Application Specific Signal Processors (RASSP) [2,3,4,5] is a modern methodology of designing embedded digital system nowadays. It supports the design of processor through a structural framework. The framework of RASSP mainly emphasises on the top-down design, design re-use and model-year design concepts [11]. Implementing these design concepts will result in shorter time-to-market and first-time silicon success fabrication.

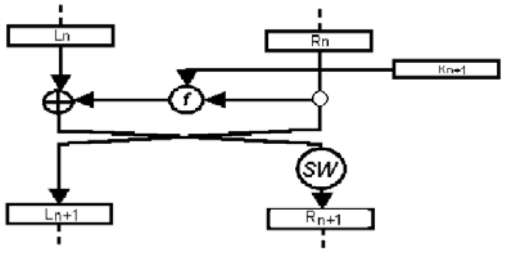
In this study, those concepts are demonstrated by using the VHDL modelling via multi-level of abstraction, with all component objects defined in a standard open interface and technology independent specification. Based on this, it is not only provides the architecture reuse library components, but also supports the rapid insertion of a new element into an existing design for upgrades or modifications.

1.2 Cryptography and Crypto-Processors

Nowadays, cryptography is commonly used in commercial and banking sectors as Electronic Commerce created these urgent needs in Electronic Fund Transfer (EFT) application. In this paper, main focus is put on the modelling of symmetric crypto-processor which encrypt fixed-length of data block. The Data Encryption Standard (DES) [6] is often used as a basic building block in the existing cryptosystem, that difference applications are used in different ways. On the contrary, attacks on DES using linear cryptanalysis and differential cryptanalysis, as well as exhaustive search are also well known. Therefore, in order to strengthen the security level of the existing cryptosystem, various kinds of modification and upgrade of the DES algorithm are proposed which using DES components as a building block. Hence, modelling the DES algorithm in a RASSP design framework helps the rapid prototyping of a new design. This is benefited from the reuse of design information and functional block library from previous design, for instance, the Randomised-DES [7,8,9] proposed by T. Kaneko, K. Koyama and R. Terada and the Extended-DES [10] proposed by H.S. Oh and S.J. Han. These are DES-based cryptosystem which used DES components as a building block.

Randomised-DES (RDES) [7,8,9] is a cryptosystem with an n-round DES in which a probabilistic swapping, *SW(Rn, Sn)*, is added onto the right half output of each round as shown in Fig. 1. It has been claimed that the n-round RDES is stronger than the n-round DES against differential cryptanalysis.

Extended-DES [10] is a cryptosystem utilising the iteration F-function of the DES to extend the property of the algorithm in form of a matrix. It defines the input plaintext as 96-bits and the key size as 128-bits, as well as the order of the S-box is randomly arranged. The 128-bits key is divided into two independent key, K_1 and K_2, and used the same key scheduling algorithm of DES for generating the subkeys. The encryption and decryption formulas of EDES are shown in Table 1. With this extended configuration, it is verified to be less vulnerable to attack by differential cryptanalysis.



Where $\boldsymbol{SW} = SW\,(R_n, S_n)$
$= SW\,(R_{nL}, R_{nR} \mid S_n)$

$$= \begin{cases} (R_{nL}, R_{nR}) & \text{if } S_n=0 \\ (R_{nR}, R_{nL}) & \text{if } S_n=1 \end{cases} \qquad S_n = G_0\,(R_n)$$

Fig. 1. The Randomised-DES (RDES)

Table 1. Encryption and Decryption Formulas of EDES

Encryption	**Decryption**
$A_i = B_{i-1}$	$A_{i-1} = C_i \text{ Xor } f(A_{i-1},K_{2,i})$
$B_i = C_{i-1} \text{ Xor } f(B_{i-1},K_{2,I})$	$B_{i-1} = A_i$
$C_i = A_{i-1} \text{ Xor } f(B_{i-1},K_{1,I})$	$C_{i-1} = B_i \text{ Xor } f(A_i, K_{1,i})$

2 Design Process by VHDL

VHSIC Hardware Description Language (VHDL) provides a media of vendor, platform and technology-independent design method of describing, simulating, and documenting complex digital system. It helps the rapid prototyping application-specific simulatable and sysnthesisable VHDL models of various signal-processing functions. The support of multi-level of abstraction, as well as working at a higher-levels of abstraction, facilitates the design transfers from the system level algorithm to structural implementation. Through out the modelling process in VHDL, it supports a cost-effective means for rapid exploration of area, speed, and power requirements of the processor. It also facilitates the functional trade-offs of algorithm and architectural design alternatives at the very early stages in the design process. The design process of VHDL can be divided into three parts as shown in Fig. 2: they are design specification, executable specification and detailed design.

2.1 Design Specification

Design specification captures customer requirements and converts these system-level needs into processing requirements (functional and performance) by VHDL description. Functional and performance analyses are performed to properly decompose the system level description. The system process has no notion of either hardware functionality or processor implementation. It also specifies an appropriate set of parameters specifying the performance and implementation goals for the processor (size, weight, power, cost, etc.). The traditional approach is to utilise text-based files in a specific format to support extraction of key parameters by the appropriate tools. Nowadays, VHDL is regarded as the unifying design representation language and tool integration approach for describing the design specification. Eventually, the design specification is translated into simulatable functions, which refers to an executable specification.

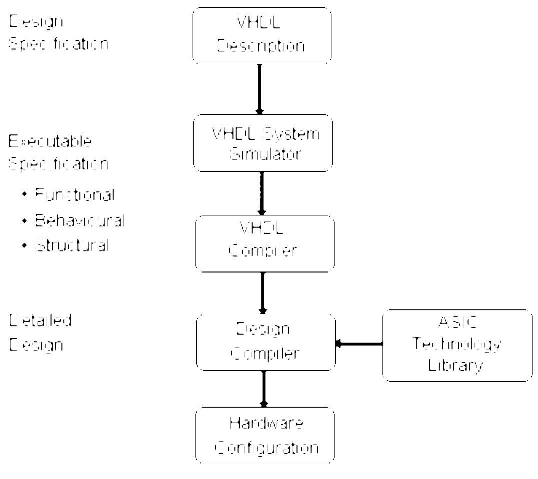


Fig. 2. The Design process by VHDL

2.2 Executable Specification

An executable specification [12] is a behavioural description of a component or system module without describing a specific implementation. The description reflects the particular function and timing of the intended design as looking on the component's interface level. During this process, the system level processing requirements are allocated to functional modules and each module is then verifying its specified functionality against the system requirements. The module is then integrated with other components of the system and to test whether an implementation of the entire system is consistent with the specified behaviour in the design specification. Finally, a virtual prototype is resulted in a detailed behavioural description of the processor hardware.

In this stage, an extensive simulation of all components is carried out in any form of the above models which can be described as functionally, behaviourally or structurally. Simulation is carried out by using the VHDL system simulator and VHDL compiler. It is intended to verify all of the codes during this portion of the processor design. After this process, all modules are fully tested and resulted in a detailed behavioural description of the processor hardware. Thus, the result of the executable specification is the virtual prototype describing the custom modules down to individual components at the behavioural level with emphasis on interface behaviour rather than internal chip structure.

2.3 Detailed Design

With the above processes, the design is modelled and verified through a set of extensive functional and performance simulations using integrated simulators in VHDL platform. At the completion of those simulations, the design is in the form of a fully verified virtual prototype of the system and the timing is also verified to ensure proper performance against the design specification. For the design to be realised in a physical hardware, in this stage, the executable specification of the processor is transformed into detailed designs in Register Transfer Level (RTL) and/or logic level which specifying the actual implementation technology. This process resulted in a detailed technology-dependent hardware layout and artwork, netlist, and test vectors of the entire processor. Making uses of that information, the processor can be put into real hardware for integration, as well as used for silicon fabrication. It is accomplished by using the VHDL design compile and the specific ASIC technology library to generate the vendor-specific hardware configuration details.

3. The Crypto-Processor Model

The crypto-processors are synthesised using the Synopsys VHDL integrated simulator and implemented in a Xilinx FPGA chip. The main task of the synthesis tool is to transfer the design into a virtual prototype with simulation and debugging of system functionality. The implementation tool is to realise the design in real hardware and used for design verification. In this section, the detailed modelling of the baseline algorithm, DES, is demonstrated. The reuse concept is also exercised in the RDES and Extended-DES models. Finally, some observations and results are shown.

3.1 Top-Down Modelling of the DES

To rapidly prototype the DES in VHDL, the procedures described in section 2 is deployed. First of all, the design specification is defined, i.e. the mathematical representations of the algorithm. Then, the algorithm is partitioned into functional modules for synthesis, in which, the algorithm is simulated in form of functional, behavioural and structural models. The modules are refined into smaller component which is implementable in FPGA architecture. Finally, the virtual prototype is transformed to detailed design of FPGA configuration netlist.

Design Specification
The design specification of the DES is the standardised algorithm defined in International Standard document [6]. As stated in the document, the DES algorithm is making use of a series of permutation, substitution and exclusive-or operations to scramble the data depending on a binary key. The core of the algorithm computation includes the Initial Permutation (*IP*), the Expansion Box (*E-box*), the Substitution Box (*S-box*), the Permutation Box (P-box), the Inverse Initial Permutation (*IP^{-1}*) and the Exclusive-OR (*XOR*) operations. By combining the *E-box*, *S-box* and *P-box* with the associated *XOR* operations, it forms the iteration function (*F-box*) which is the core

computation unit of the DES. In addition, the Key Schedule (*KS*) associated with the algorithm provides the 48-bits subkeys used in each round of iteration. The KS includes the Permutation Choice-1 and -2 (*PC-1, PC-2*), and a series of shift operations.

According to the DES specifications, all computation of the above units follows a set of operation tables defined in the standard [6]. To capture the design for implementation, each operation table specified in the standard is coded as a functional entity in VHDL description. Eventually, a DES VHDL package, which translates the textual specifications into synthesisable VHDL code, is modelled. The package is then used for program coding, design validation and system integration.

Executable Specification

In this stage, the functionality of the algorithm is validated by simulating and testing the algorithm in VHDL simulator, ultimately completed with a fully verified virtual prototype of the algorithm. To achieve this, the process is conducted through a combination of functionality partitioning and synthesis at all levels of abstraction. The partitioning of the model into smaller modules also facilitated the reuse concept.

The DES algorithm is partitioned into four top-level functional modules, including the *F-box*, the *IP*, *IP*$^{-1}$ and the *KS*. In those functional modules, their interfaces between sub-modules, as well as the resource requirements (performance/area) for each component module are specified. Probably, the functional module is further decomposed into lower behavioural level model, so as to form a layered-architecture. This layer approach made the module more manageable, understandable, reusable, and maintainable. This helps to facilitate the design reuse concept and to build an encapsulated library. For instance, the functional module, *F-box*, with the defined interface is shown in Fig. 3, and the partitioned behaviour module of the *F-box* is shown in Fig. 4, with each box represents a behavioural component model in VHDL description entity.

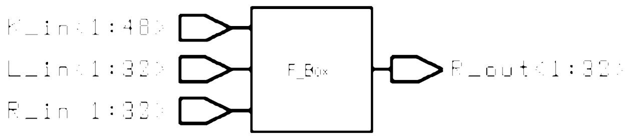


Fig. 3. Interface of F-box

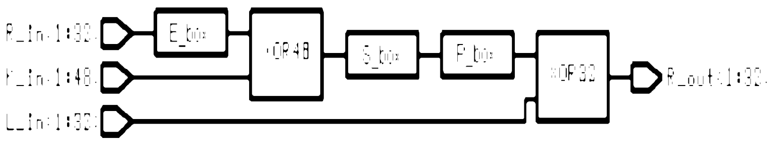


Fig. 4. Partitioned behavioural modules of F-box

The behavioural VHDL description of each module is then designed and tested individually using the pre-defined VHDL DES-algorithm package as the component library. At this level, further model partitioning is also conducted for each sub-module, until a sufficient detail is resolved for physical construction, performance and area requirements in the specific technology platform.

In this case, the S-box is further partitioned in smaller sub-modules with respect to satisfy the area and speed requirements. With this partitioning, the large function is implemented in a minimum of area and the delay in signal path, and thus achieves performance increase both in spatial and speed requirements. Finally, the entire algorithm is integrated by a structural model which interconnects all verified modules together.

In this stage, the functionality of all modules is verified and the timing analysis is computed. This ends up with a technology-independent and fully functional verified virtual prototype of the algorithm. For it is synthesisable in real hardware, say a FPGA chip, it needs to forward to the VHDL design compiler with the specific FPGA technology library to generate the detailed FPGA configuration netlist.

Detailed Design

Detailed design will match the design into a physical reality. The virtual prototype synthesised in the pervious stage is ready for realisation in FPGA. In this stage, the entire design is converted into the FPGA netlist by the VHDL design compiler and the associated FPGA technology library. As a result, the Xilinx Netlist Format (xnf) file is generated. By making use of this netlist in the automated development environment, logic mapping, placement, and routing are done automatically and finally a FPGA configuration file is generated. Then the configuration file is stored inside an EPROM for programming the FPGA in the real hardware prototype.

Finally, the DES algorithm is transformed into the FPGA configuration file. The spatial requirement of the algorithm in pipeline mode occupied 2,176 Configuration Logic Block (CLB) with a signal path delay of 164.96 ns. Detailed synthesis results of all modules are shown in Section 4.

3.2 Design Re-use for the Randomised-DES and Extended-DES

Randomised-DES [7,8,9]

RDES is the case of design insertion in the existing DES algorithm. It is an extension of the DES by inserting a special modular, *SWAP*, in the algorithm as illustrated in Fig. 1. By the modular-design concept applied in the DES design, the VHDL DES-algorithm package library, as well as the verified functional, behavioural and structural models, are reused as the components for constructing the RDES.

In the virtual prototyping stage, only the functionality of the *SWAP* module needs to verify as it is stated in the specification. The insertion of the *SWAP* module only affected the internal structural of the *F-box* which is retained with the same interface structure. Thus, all structural models remained in its defined interface, as well as their original interconnection between modules as in the DES model. The modified structural of the *F-box* model is shown in Fig. 5.

As a result, all of the structural models designed in the pervious DES model is reused. For the entire RDES algorithm design, modification is made only in the structural model of the *F-box*. By this design reuse of the DES algorithm component library, the RDES algorithm is rapidly prototyped within one-man week. This is achieved by the top-down, model-year structural design methodology applied in the designing of the DES algorithm.

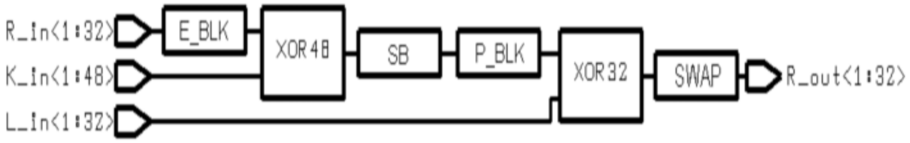


Fig. 5. Modified Structural Model of F-box for RDES.

Extended-DES [10]

EDES is the case of modifying the existing DES algorithm. EDES is just an extension of the DES by increasing its data block length to 96-bits and the key size to 128-bits, in addition with a special arrangement in the order of *S-box*. In such, the processing block size is remained in 32-bits using the same DES iteration *F-box* as its core, and the same key-scheduling algorithm that is used in DES. In this case, only the top-level structure model is needed to re-design and the *S-box* sub-module is needed to re-structure. In the *S-box*, since the functionality of all the smaller components in the lower-level module is verified in the DES-algorithm library. It only needs to re-program the structural model according to the EDES design specification and then reinsert it back to the encapsulated library. The *F-box* module and all other sub-modules within the *F-Box* are not affected at all.

Therefore, the modelling of the EDES required in this case is just to modify the structural level models. All functional models of the processing units are using the standard DES modules extracted from the encapsulated library built during the previous design. As a result, the virtual prototype of EDES re-defined the interconnection between modules in a structural model and this re-design is prototyped by one-man week.

4 Observations & Results

4.1 Space and Speed Requirements

To prototype a design into a physical hardware, such as in FPGA chip, the space and speed constraints in physical device are not negligible. Since all electronic technologies deliver finite spatial resources for building functions and wiring resources for communications which are especially tight with FPGA. By using the top-down design concept to partition design functionality into small modules has facilitated the design optimisation against those constraints.

During the modelling of DES algorithm, the following results are observed. By transforming the functional model directly to detailed design, the resultant requirement in space and delay are higher. In the partitioned behaviour model which module is in form of a small component, the resultant requirement is much lower. The results of the DES modules are tabled in Table 2.

Table 2. Synthesis results of the DES module[1]

Module	Un-partitioned Functional Model		Partitioned Behavioural Model	
	CLB	Timing(ns)	CLB	Timing (ns)
IP	0	0	0	0
IP-1	0	0	0	0
E-box	N/A	N/A	0	0
P-box	N/A	N/A	0	0
S-box	230	33.92	96	6.31
XOR32	N/A	N/A	16	3.26
XOR48	N/A	N/A	24	3.26
F-box	323	42.38	136	10.31
KS	0	0	0	0
SWAP	N/A	N/A	16	3.62

From the table, it is found that the spatial requirement of the partitioned s-box is reduced by 60% and the delay is reduced by over 80%. (CLB's propagation delay is reduced from 7 stages to 2 stages). While in the partitioned F-box, a 60% reduction is achieved. CLB's propagation delay is reduced from 9 stages to 4 stages, above 75% reduction is accomplished. With this result, the algorithm is more feasible for implementation in FPGA chip with benefits in both spatial and timing requirements. Those benefits are also encountered in the case of RDES and EDES implementations.

[1] *Timing is measuring under the Xilinx Xfpga_4025e-3 library parameters: path_full, delay_max, max_paths, and WCCOM operation conditions*

4.2 Design Insertion and Modification

Through designing the DES algorithm in model-year architecture, by defining the module with open interface and partitioning functionality into small components, it can facilitate the rapid design insertion and modification. Like the cases of modelling the RDES and EDES, the turnaround time to prototyping those algorithms are reduced rapidly. In modelling of RDES and EDES, above 70% and 40% of the development time is eliminated respectively. This is achieved by the result of using the encapsulated library, as most of the functionality verification is exempted. Thus, the design reuse concept of the encapsulated component library has shown its advantage and significance in this aspect.

4.3 Design Automation

Beside the design methodology, the platforms for simulation, debugging, synthesis, logic placement, routing, test vectors generation and hardware implementation the design are also important. Any one of those elements cannot be omitted in the process of processor design and prototyping. Therefore, a standardised integrated development environment is essential for designer, so as to speedup design process and reduce design transfer/translation cumbersome. In this study, the use of Synopsys VHDL integrated platform has helped a lot in the design automation aspect from the design capture to the synthesis in hardware.

4.4 Hardware Prototype

The designs are realised in a 25,000 logic-gates FPGA chip for testing and integration. Those algorithms are synthesised in both recursive and pipeline mode of operations. The hardware prototype is as shown in Fig. 6.

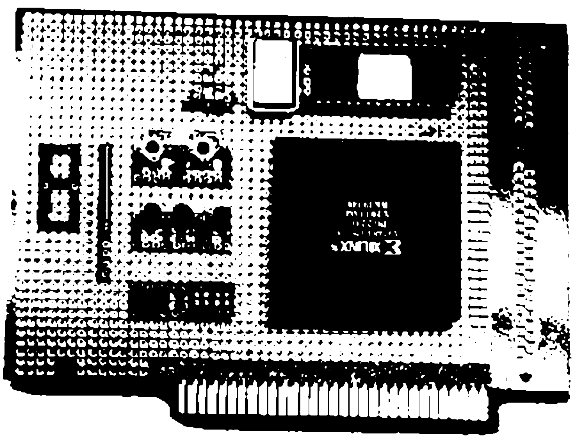


Fig. 6 Prototype of the Hardware

5. Conclusions

Through out the modelling of the crypto-processors in this study, the design concept of rapid prototyping the application-specific signal processors is practised. By the described approach, it not only verifies design functionality early in the design process, but also provides the key to rapid prototyping and upgrading of signal processors, as the same time reduces the development time and costs significantly.

Deployment of model-year design concept in rapid prototype has provides the use of previous models as a baseline for further developments. As in the cases of modelling the RDES and EDES algorithm, which using DES as a baseline, allowed the modification of the functional models in the virtual prototyping stages and allowed partitioning and re-targeting design during the synthesis activities. In this case, above 50% of development time is reduced in modelling the RDES and EDES algorithms.

On the other hand, in the VHDL development environment, design can automatically converts a VHDL description to a gate-level implementation in a given technology; and can automatically transform a synthesis design to a smaller or faster circuit through partitioning. In this experience, above 60% of spatial and 75% of timing reduction is achieved. In addition, capturing the design in VHDL technology-independent functional models for the virtual prototype also enhances reuse of functional primitives and generates the design in different technology. Simultaneously, it also provides a technology-independent documentation for a design and its functionality.

To conclude, the modelling is carried out in a structural manner from the design capture in VHDL code to design synthesis in FPGA prototype. Through those prototyping procedures, the turnaround time of the design cycle is reduced; and through the modular design concept, the feasibility of design upgrade and modification is enabled.

6. References

1 Rita C. Summers, “Secure Computing: Threats and Safeguards”, McGraw-Hill, Chapter 5, 1997.

2 Mark A. Richards, Anthony J. Gadient and Geoffrey A. Frank, “Rapid Prototyping of Application Specific Signal Processors”, Kluwer Academic Publishers, Boston, February 1997.

3 Mark A. Richards, “The Rapid Prototyping of Application Specific signal Processors (RASSP) Program: Overview and Accomplishments”, Proceedings of the 1st Annual RASSP Conference, pp.1-10, August 1994.

4 Jeffrey S, Pridmore and W. Bernard Schaming, “RASSP Methodology Overview”, Proceedings of the 1st Annual RASSP Conference, pp.71-85, 1994.

5 Carl Hein, Paul Kalutkiewicz, Todd Carpenter and Vijay Madisetti, “RASSP VHDL Modelling Terminology and Taxonomy – Revision 2.3”, RASSP Taxonomy Working Group (RTWG), June 23, 1997.

6 “Data Encryption Standard”, Federal Information Processing Standard (FIPS) 46, Nat. Bur. Stand., Jan. 1977.

7 K. Koyama and R. Terada, "How to Strengthen DES-like Cryptosystems against Differential Cryptanalysis", IEICE Trans. Fundamentals, Vol. E76-A, no. 1, Jan. 1993.
8 T. Kaneko, et. al., "Dynamic Swapping Schemes and Differential Cryptanalysis", IEICE Trans. Fundamentals, Vol. E77-A, no. 8, Aug. 1994.
9 Y. Nakao, et. al., "The Security of an RDES Cryptosystem against Linear Cryptanalysis", IEICE Trans. Fundamentals, Vol. E79-A, no. 1, Jan. 1996.
10 Haeng-Soo Oh and Seung-Jo Han, "Design of the Extended-DES Cryptography", Proceeding of 1995 IEEE International Symposium on Information Theory, pp.353, 1995
11 Jeff Pridmore, Greg Buchanan, Gerry Caracciolo and Janet Wedgwood, "Model-Year Architectures for Rapid Prototyping", Journal of VLSI Single Processor 15, 83-96 (1997).
12 Allan H. Anderson and Gray A. Shaw, "Execute Requirements and Specifications", Journal of VLSI Single Processor 15, 49-61 (1997).