A Pipelined Parallel Processor to Implement MD4 Message Digest Algorithm on Xilinx FPGA

M. Bhaskar Sherigar * **A.S. Mahadevan**+ **K. Senthil Kumar**+ **Sumam David**#

* *Presently working in Armedia Labs Pvt Ltd, Bangalore ,INDIA*
\+ *Presently working in Central Research Laboratory, BEL, Bangalore, INDIA*
\# *Presently working in Karnataka Regional Engineering College, Surathkal, INDIA*

ABSTRACT ***The paper presents a Pipelined Parallel Processor Architecture design to implement MD4 Message digest Algorithm which computes the message digest or the fingerprint of 128 bit fixed length, for any arbitrary length of input message. The processor implements the arithmetic, logic and circular shift operations by Pipelined Parallel Process. The architecture is designed to suit the design flexibility of the XILINX Field Programmable Gate Arrays (FPGA). The Processor reads the message from an external RAM, 16-bit at a time and internal operations are performed with 32-bit data. The major advantage of the design is increased speed of computation and minimum hardware. The processor computes the digest with a speed approximately three times faster than the software version implemented in DSP processors.***

1. INTRODUCTION

In the quest for the new design alternatives FPGAs has come out as promising one with reduced design turn-around time. **XILINX FPGAs** provide ultimate flexibility due to their re-programmability. These devices can be re-configured to change the logic function while they are embedded in the system. Hardware can be changed as easily as software.

SRAM based XILINX FPGAs has a modular architecture, rich in registers and powerful function generator. In the present design Xilinx **XC3090** chip is used which has **320 CLBs** [8]. Each CLB consists of a *five input* combinational block, *two flip-flops* and two *outputs.*

The perimeter of *configurable* Input Output Blocks (**IOB**) provides a programming *interface* between the internal logic array and the device package pins. The interconnect resources are *programmed* to form the network. The Logic Configurable Array (**LCA**) functions are established by *configuration program* which is loaded into an internal, distributed array of configuration memory cells.

Digital Signature [4] fulfil the need for *authenticity* in secure communication systems. Several schemes [4] are available for digital signature, but has limitations in the speed of *computation* or *transmission bandwidth.* These requirements are met by using *Hash functions* [4]. A hash function **H** accepts a variable size message **M** as input and outputs a *fixed* representation **H(M)** of **M** called **message digest.** In general **H(M)** of **M**, is much smaller than **M**. A digital signature may be applied to **H(M)** in relatively quick fashion. i.e. **H(M)** is signed rather than M. Both **M** and signed **H(M)** may be encapsulated in another message which may then be *encrypted* for *secrecy*

MD4 message digest algorithm [3] takes an input of arbitrary length and produces an output of 128-bit Fingerprint or message digest in such a way that it is computationally infeasible to produce two messages having the same message digest or to produce any message having a given pre-specified target message digest.

2. PROCESSOR ARCHITECTURE

A major concept that received considerable attention in the design of high speed computation is ***Pipelining.*** In order to meet the indigenous characteristics of the message digest algorithm, the processor utilises a Pipelined Parallel

Processing Architecture, as shown in the Figure.1. The operation are performed on 32-bit data by reading the message from an external RAM, 16-bit at a time. The set of operations defined by the algorithm are performed on a block of 512-bits message.

2.1 Execution Unit

The processor functional blocks implements Logic, Arithmetic and Circular shift operations.

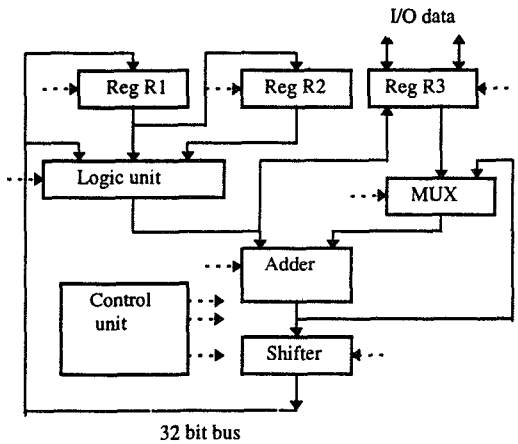

Figure 1. Processor Architecture

In addition, the registers store the intermediate results and initialise the constants A, B, C and D, as well as the magic constants for round two and round three of the algorithm[1]. Register R1 initialises the constant B and Register R2 initialises the constants A, B, C, magic constant 1 and magic constant 2. Register R3 stores the external message to be processed by reading 16-bits data twice corresponding to a single word of 32 bit length. All Registers are 32-bit wide in length.

A multiplexer with output Register, Selects and stores the contents from R3 or from the adder. The arithmetic unit performs addition of two 32-bit words by using a cascaded carry look ahead adder (CLA) of four bits forming a sequential adder of eight stages. This reduces the delay in propagation of carry from LSB to MSB.

The Logic unit performs three auxiliary functions defined by the algorithm given by

$f(x, y, z) = xy \vee (\neg x)z$

$g(x, y, z) = xy \vee xz \vee yz$

$h(x, y, z) = x \oplus y \oplus z$

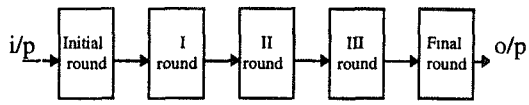

Figure 2. Sequence of Operations

The functions f (x, y, z), g (x, y, z), h(x, y, z) are performed in round one, round two and round three respectively. A multiplexer at the output stage selects any one function and stores the result in output register LUR.

A sequential shifter, circularly shifts left in round one two and three with different shift counts. The number of the shifts are given by the algorithm. It is different for each step in all the three rounds.

The operations performed according to the algorithm are shown in Figure. 2. The constants A, B, C and D are initialised during the initialisation round. Simaltaneously the constants are stored in the external RAM for further processing in the final round. The addition of the constants and the result of the third round operations are performed in final round which may be the required message digest or may be partial result. This partial result forms the constants for the next 512 bits message. Round one , two and three corresponds to the rounds mentioned in the algorithm.

2.2 Control Unit

The control unit generates timing signals as shown in the Figure 3, it is a hardwired control

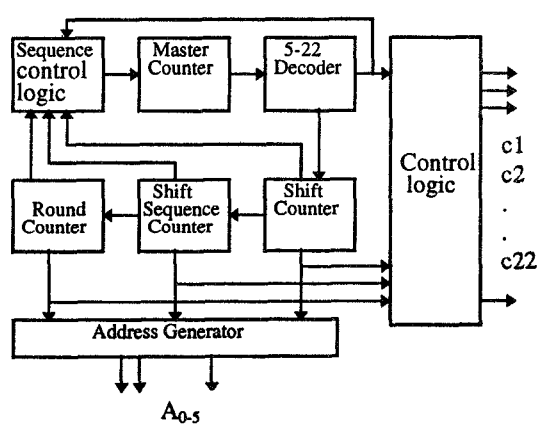

Figure 3. Control Unit

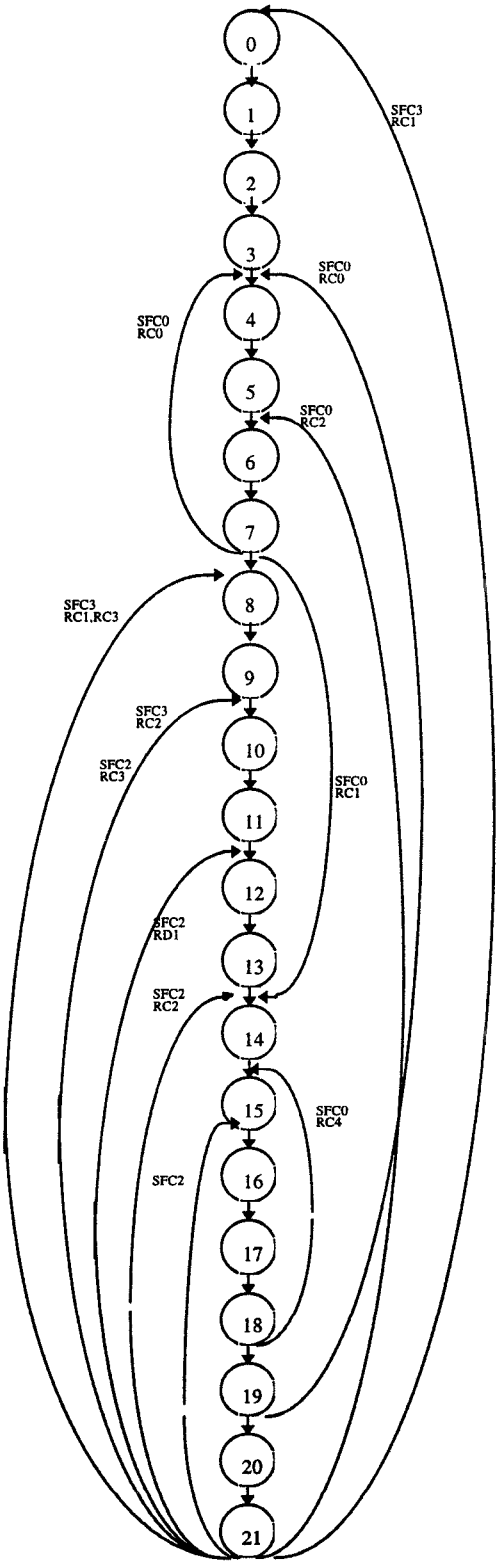

Figure 4. State diagram of Master Counter

unit. A MOD 22 master counter is decoded to generate the control signals in control logic unit. Three counters shift counter, shift sequence counter and round counter keeps the track of the operations taking place from initial round to the final round. The sequence of the operations are controlled by the sequence control logic depending on the status condition of the shift counter, shift sequence counter and round counters. The state diagram of the control unit is as shown in Figure 4.

The state diagram of the control unit is split into four state diagrams which reduces the complexity of the control unit without affecting the hardware requirements. These state machines corresponds to shift ,shift sequence and round counters along with master counter. The outputs of these counters are denoted as SFC_{0-3} ,SFS_{0-3},and RC_{0-4} respectively.

An address generator points the memory address from which the next data is to be read. There are six address lines which carries the address of the data to be read or stored. The initial constants are read in final round. The data read is in sequential order from consecutive memory locations for round one where as for round two and round three the data read is not in regular fashion. The number of shifts in each step repeats with a regular fashion.

3. PIPELINED OPERATIONS

Three operations are Pipelined and processed independently and Simaltaneously as shown in the Figure 5. In addition to this, initialisation of the magic constants for round

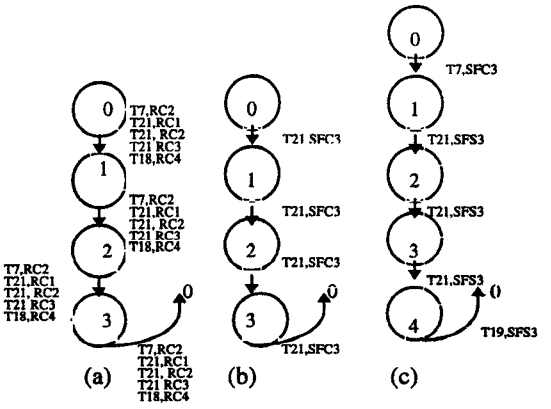

Figure 4. (cntd..) Control unit state diagram
(a) Shift counter
(b) Shift sequence counter
(c) Round counter

two and round three and reading the message from the external RAM is done Simaltaneously.

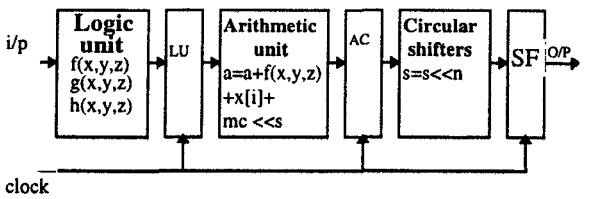

Figure 5. Pipelined Process

3.1 Initial Round Operations

First step is to initialise the constants A, B, C and D. Second step is to store these constants in memory. These values are retrieved at the final round to perform the addition. The first step is performed only once in the computation of the message digest, but second step is repeated for a block of every 512 bits data. The flow chart of initial round is shown Figure 6.

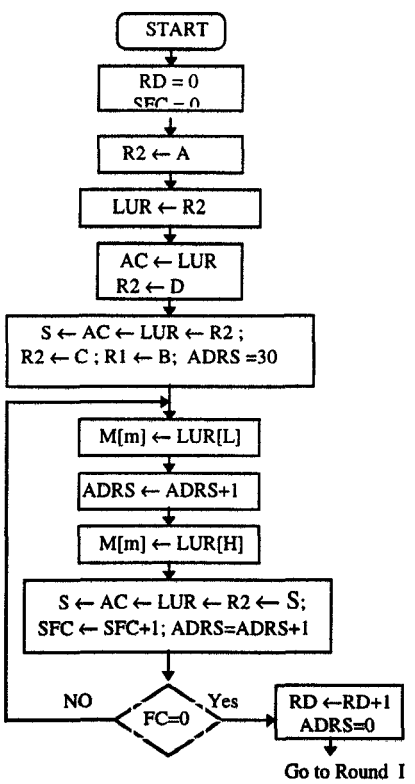

Figure 6. Initial Round Operations

3.2 First Round Operations

Flow chart in Figure 7. shows the operations performed in round one. There are 16 steps in this round and each step is dependent on the previous result. Circular shift operation of each step is performed in the next step parallely during which the addition, logic function and reading the external data are in progress. This minimises the delay.

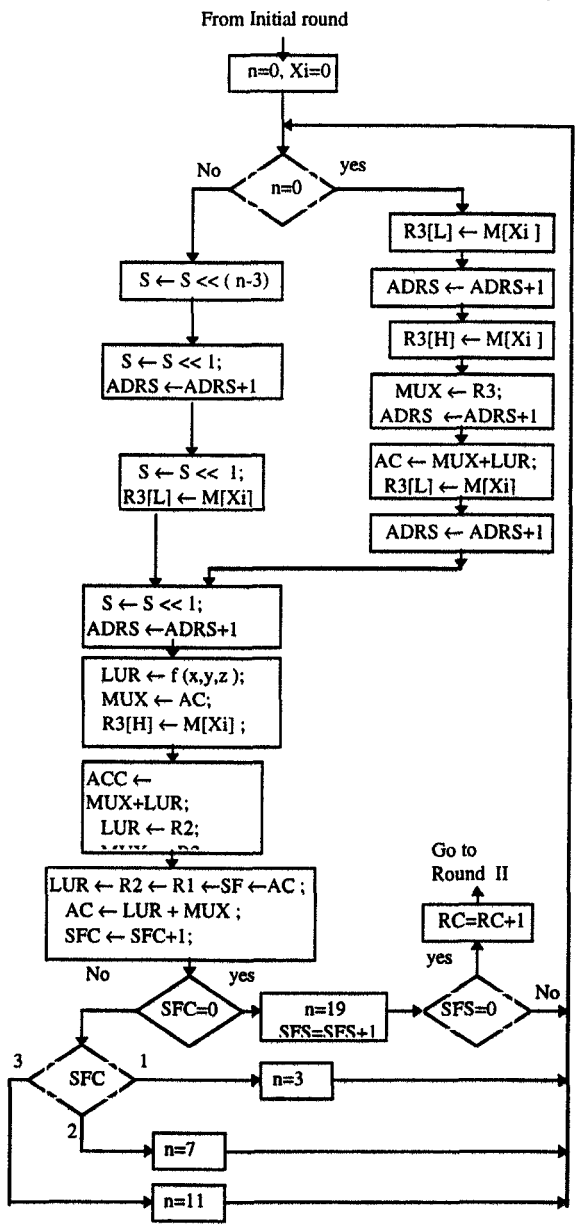

Figure 7. First Round Operations

3.3 Second Round Operations

The second round operations are similar to the first round operations except for Magic constant, shift count and the sequence in which the message blocks are read. The message is read in a fixed irregular fashion. Flow chart of second round is shown in Figure 8. A typical operation taking place in this round is given below.

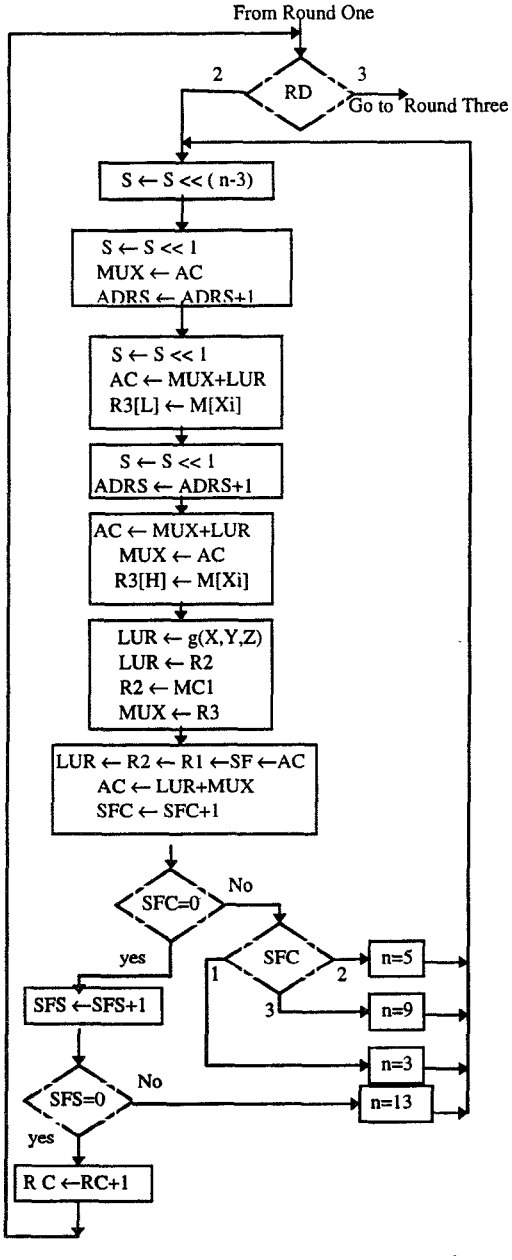

Figure 8. Second Round Operations

$A = A + g (B, C, D) + X[i] + MC1 <<< s$

A,B,C, D → Register contents at each step which are the intermediate results.

X[i] → Message block ,

MC1 = 5A 82 79 99 →Magic constant.

3.4 Third Round Operations

The sequence of operations taking place in third round are same as round two except for the memory contents read, the Magic constant MC2= 6E D9 EB A1[3] and the shift sequence. The flow chart is same as round two operations.

3.5 Final Round Operations

At the end of round three the end result is added to the initial constants stored in external RAM. This operation may be the last operation

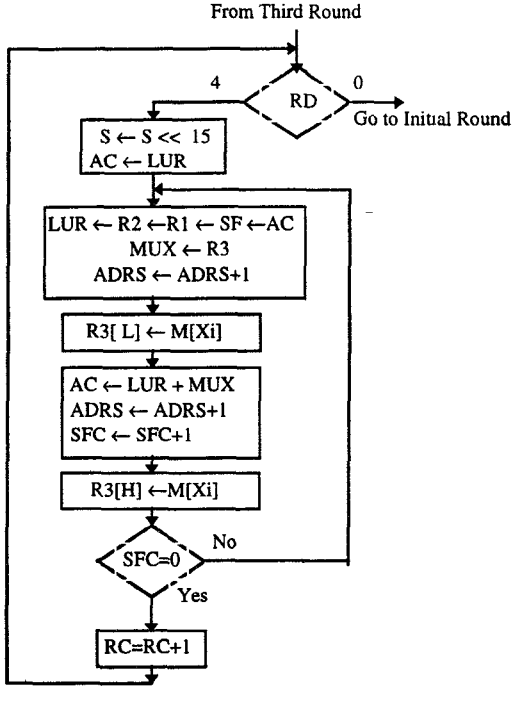

Figure 9. Final Round Operations

or may be of any intermediate part. The result after addition thus may be the Digest or the initial constants for next set of operations. Flow chart in Figure 9, depicts the sequence of the operations.

4. SIMULATION RESULT

A Pipelined parallel processor architecture is designed and the design is implemented on XILINX XC3090PG175-125 FPGA chip. The processor assumes that the padding the message as defined by the algorithm is done externally and is available at RAM. The design is simulated and the Processor works with a frequency of 6.67 MHz. It takes 94.07 microseconds to compute the digest for the message block of 512-bits with the chip having a speed grade of 125[8]. The speed of operation of the present design is almost three times faster than the implementation of the algorithm on DSP Processor. The ADSP 2101 processors takes 260 μs to compute the digest for 512 bits message block with a clock period of 50 ns.

The specification of the processor is summarised below.

- Frequency of operation **6.67 MHz**
- FPGA IC used **XC3090PG175**
- Number of IOBs used **42**
- Number of CLBs used **252**
- Number of address lines **06**
- Number of data lines **32(external 16)**

SOFTWARE TOOLS USED

- Design entry **FutureNet**
- Design interface **XILINX XDM/XACT**
- Design verification **P/C SILOS**

5. APPLICATIONS

The processor has its main application in Crypto-Communication field. Authenticity of the message is the prime requirement in applications like remote banking, defence etc. Since hardware implementation reduces the computation speed the software can be totally replaced by the processor which has simple interfacing and low cost. The message digest can be signed and appended to the original message before encryption for secrecy.

6. DISCUSSION & CONCLUSION

The processor is designed to meet the goals of the MD4 Message Digest Algorithm, utilising the design flexibility of the Xilinx FPGAs adapting a Pipelined architecture to achieve higher computation speed. The Execution unit consumes the major part of the hardware totalling of 148 CLBs and control unit consumes 104 CLBs. The control unit is simplified by state splitting technique. In the design a cascaded carry look ahead adder is used which consumes more hardware but delay is reduced by 75%compared to the sequential adder which consumes only 32 CLBs. If delay is not important then sequential adder is preferred. Another unit where time delay can be reduced is the Circular Shifter. A better choice would be a barrel shifter if hardware is not a constraint for reduced delay designs. The architecture can be modified to achieve more parallelism but the routing congestion will increase. The frequency of the operation can be increased by implementing the design in Xilinx 4000 series devices. An ASIC based on this design would give optimum performance, because a FPGA has general architecture.

7. ACKNOWLEDGEMENT

We are grateful to Mr. H. Ramakrishna *Chief Scientist* and Mr. Sethuraman *Fellow* at CRL-BEL for the exchange of ideas and discussion. We wish to thank the staff members at CRL-BEL, K.R.E.C., Surathkal, V.V.N. Sudhakar Reddy at CRL and *Armedia Labs* Bangalore for their support.

8. REFERENCES

[1]. FutureNet *Schematic Designer* Data I/O Corporation, Washington.

[2]. P/C SILOS *Logic Simulator User Manual* SIMUCAD Incorporation, California.

[3]. Revest R.L ***The MD4 Message Digest Algorithm,*** Lecture notes in Computer Science, Advances in Cryptography CRYPTO '90.

[4]. Simmons G.J *Contemporary Cryptography* the science of information integrity IEEE Press, New York 1991.

[5]. XILINX *User guide and Tutorials* Xilinx Incorporation, California.

[6]. XACT 2000/3000 *Programmable Gate Array* Development System, Xilinx Incorporation, California.

[7]. XACT *Design Interface User Guide* Xilinx Incorporation, California.

[8]. Xilinx *Programmable Logic Data Book* Xilinx Incorporation, California. ■