Pipeline Implementation of Secure Hash Algorithm (SHA-1) for Cryptographic Application in Network Security

Mohamed Khalil Hani Ahmad Zuri Sha'ameri Chong Wei Sheng

Microelectronic & Computer Engineering Dept. (MiCE)
Faculty of Electrical Engineering, Universiti Teknologi Malaysia,
80310 UTM Skudai, Johor, Malaysia
tel: 07-5505003 fax: 07-5566272
Email: chongweisheng@nadi.fke.utm.my

Abstract- **This paper presents a pipeline implementation of the Secure Hash Algorithm, SHA-1. This design is targeted for the Altera Flex10KE50 Field Programmable Gate Array (FPGA) on a PCI card. All input output data interface and internal operations are performed in 32-bit width. The major advantage of this design is the higher throughput gained from the pipeline operations, therefore double speed performance is obtained compared to the equivalent software implementation on Sun SPARC 20/71. The higher throughput shows the potential of this design to support critical network security system, for example the digital signature in e-commerce.**

I. INTRODUCTION

Hash function is a compression function that produces a digest of an input message. Cryptographic hash function is a one-way and collision free hash function. It is related to the notion of checksum, because it produces a checksum that is infeasible to be inverted [6]. The secure digest can be used for password storage, digital signature and authentication. Therefore cryptographic hash function is essential for network security, for instance e-commerce and remote access.

SHA-1 [2] is a strong one-way hash function that is being employed in various standards, algorithms and products such as Pretty Good Privacy (PGP), Secure Hash Standard, Yarrow and Intel Pro/100 S server adapter. It is a block-chained digest algorithm, computed over the data in phases of 512-bit blocks. The first block is initialized with an 160-bit initial seed, the seed for the next block is generated from current block. Digest of the last block would be the 160-bit message digest of the entire stream. Conceptual flow of SHA-1 hashing operation for an input message is depicted in Figure 1 while the SHA-1 algorithm that is related to this paper is shown in Appendix A.

Although SHA-1 is a strong cryptographic hash function, its operation is rather computational demanding, thus cause a performance bottleneck in the overall system [1, 6]. In the paper entitled " SHA: A Design for Parallel Architectures?", Bosselaers, Govaerts and Vandewalle (1997) have shown that SHA-1 offers a number of parallelisms [1]. Based on their research, this paper proposes a pipeline architecture to exploit these parallelisms. The next section explains the pipeline SHA-1 architecture. Section III investigates the pipeline operations of SHA-1. The simulation results and discussion are presented in section IV followed by the conclusion in section V.

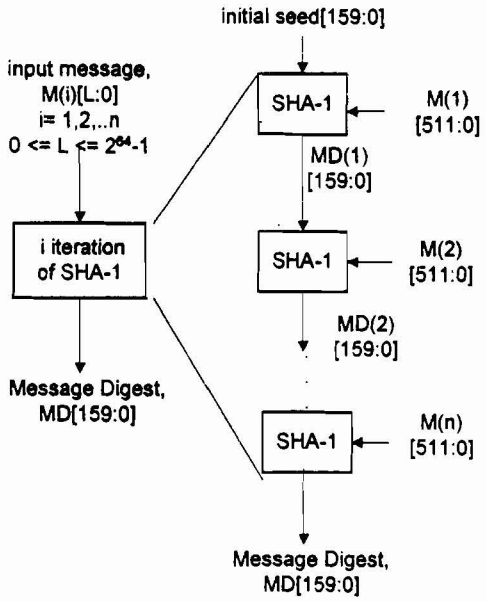


Figure 1: Flow of Iterative Operations in SHA-1 Hashing

II DESIGN ARCHITECTURE

Pipelining is one of the design principles in high-speed hardware design, it exploits parallelisms for shorter clock cycle and higher throughput. Thus pipeline becomes the design principle of this SHA-1 implementation [1].

Besides that, this design adopts 32-bit buswidth and big-endian design architecture to comply with the data structure of SHA-1. In general, control unit (CU) and

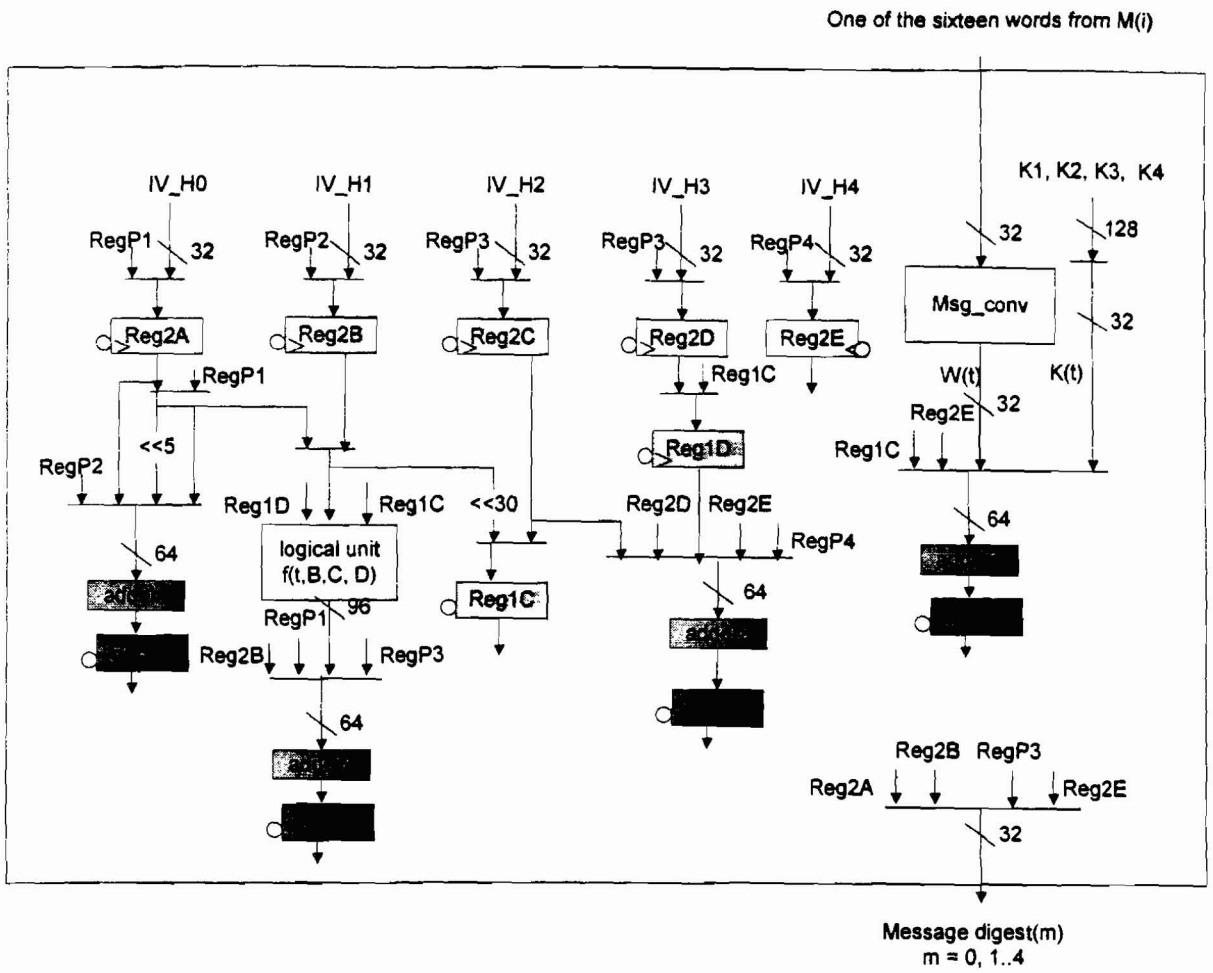


Figure 2. Datapath Unit of Pipeline SHA-1

datapath unit (DPU) (Figure 2) are the main modules in this design.

i. Datapath Unit

The DPU consists of functional units and storage elements. The functional units are logical units, modulo 32 bit adders, multiplexers and barrel shifters; whereas registers are the storage elements.

For each M(i) processing of SHA-1, M(i) is fed into DPU in the sequence sixteen 32-bit words and output message digest is fed out as sequence of five 32-bit words. Initialization vectors, IV_H's and constants, K's are hardwired. Reg2's stores the current H's for each processing block (Appendix A). Since these H's are used in the beginning and last few steps in each block, thus registers are needed for their storage. Reg1 keeps variables that are read and updated in each step. Msg_conv converts input messages, M(i) into eighty words, W(t) (Appendix A: equation a and b). Outputs of modulo-32 bit adders are pipelined with RegP's. Logical units generate its 32-bit output according to the logical functions in Appendix A. A 4:1 multiplexer selects output message digest.

For each block of input message M(i), the 80-step operations are performed in six rounds (Figure 3). Since the first to fourth round perform the same operations (Appendix A: equation d), they are grouped as intermediate rounds. There are a few overlapping among these six rounds due to the available parallelisms in SHA-1 and the pipeline architecture in this design. These overlapping reduce the number of clock cycles in pipeline SHA-1 implementation.

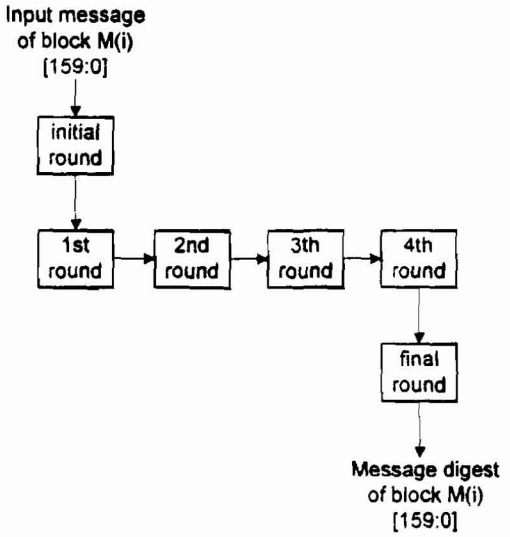


Figure 3: Sequence of SHA-1 operation

ii. Control unit

This design applies counter-based control unit to generate timing signals that coordinate the operations in DPU. Although there are only eighty steps in SHA-1

algorithm, mod-84 counter is used because of the latency in the pipeline operations. The state diagram of the CU is attached in Figure 6 (Appendix B). Initial round of first message block follows the state diagram in Figure 6.a. After this initial round, all coming rounds in the every message block use common state diagram (Figure 6.b) as they have similar operations. As shown in Figure 1, SHA-1 processing of M(2) to M(n) takes the message digest of previous block and produces the seed or the final message digest, therefore they also share the same state diagram (Figure 6.b). The message digest is delivered starting from the final round of final message block (Figure 6.b). 160-bit message digest of entire message stream is completed within two steps after the processing of final block (Figure 6.c).

III PIPELINE OPERATIONS

Data dependency is one of the criteria in pipeline design. Since SHA-1 is an iterative algorithm, data dependency between each step becomes the focus of our pipeline design. Critical path analysis on consecutive steps in SHA-1 has shown that updated variable *A* in current step is to be circular shifted and added before becoming the next variable *A* [1]. In this design , since the fixed circular shifting is implemented through barrel shifter, the operations of modulo 32 addition become the only critical path operation to be pipelined.

In SHA-1, There are a total of five 32-bit variables to be added. These additions are pipelined as shown in Figure 4, parallel operations in one clock cycle are drawn in the same horizontal level and operations of the same step are shown between dotted lines.

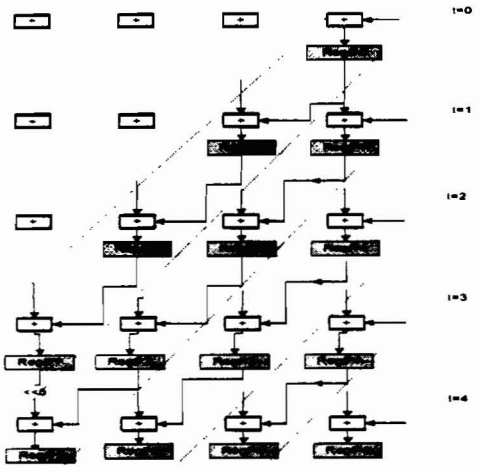


Figure 4: Pipelining Modulo-32 bits Adders in SHA-1

Due to the latency of four clock cycles, some adders are not occupied at the final round, which means they are available for the addition of the seeds in each message block (Appendix A).

i. Initial Round and Final Round Operations

Two operations in the initial round are the initialisation of Reg1's and Reg2's. Reg1C and Reg1D are always initialised with Reg2C and Reg2D respectively. For the first message block, Reg2's are initialised with the IV_H's. In the final round of every block, updated H's are stored in the Reg2's to become the message digest of current block and the seed of coming block, thus the Reg2's of coming block are initialised. (Appendix A: equation e).

ii. Intermediate Rounds Operations

During the intermediate rounds (Figure 6.b), Msg_conv supplies W(t) by selecting either the 32-bit input message or the circular shifted data, which is the xor of four previous W(t)s (Appendix A: equation a and b), In current design, register are used to store the these previous W(t)s. Since up to W(t-16) is needed, minimum registers required are sixteen (Figure 5). The Reg1's and RegP's are updated in the intermediate rounds (Figure 6.b) according to equation d (Appendix A). K(t) is selected from K's according to the sequence of rounds (Appendix A).

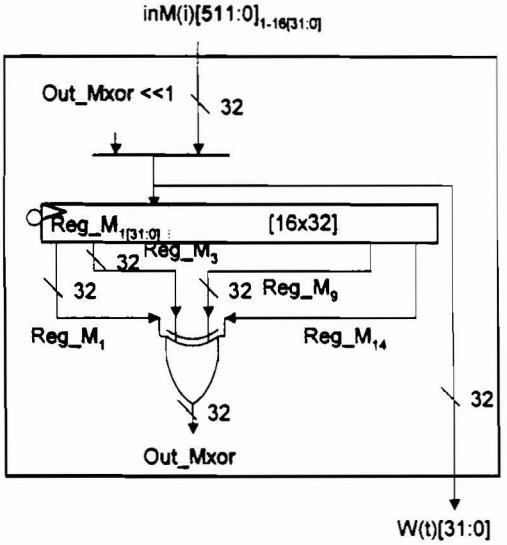


Figure 5 Structural Diagram of Msg_conv

IV SIMULATION RESULTS & DISCUSSION

The timing simulation results are shown in Appendix C. The operating frequency is 10Mhz on Flex10KE50 at speed grade of –3 [4]. It takes a total of 84 clock cycles to compute the message digest of 512-bit message block, thus the speed of 61 Mbps is obtained. This speed is about double compared to the software implementation on Sun SPARC 20/71 [5].

This hardware optimization design excludes the operations that can be efficiently handled by software. For examples, padding the input message, breaking the

input message into 512-bit blocks, dividing each block into 32-bit inputs and cascading 5 words into a message digest. The design is coded in 800 lines of VHDL using behavioral description. On the Flex10e50 FPGA, this implementation consumes hardware resources of 1191 logic cells including 879 flip-flops.

Parallelism of consecutive steps and block are exploited by pipeline architecture. To optimize the critical path operations, zero cost barrel shifters and hardware consuming carry look ahead adders are used.

V CONCLUSION

Beside achieves the security strength of SHA-1, this design also optimizes the parallelisms in the SHA-1 using pipeline architecture and the design flexibility of Altera Flex10KE50. Positive preliminary result indicates its potential to support critical network security applications, such as e-commerce and secure file transfer. For current design, storage elements of Msg_conv are implemented in flip-flops, however for future design, embedded RAM in Flex10KE50 would be used to obtain area efficiency. Future design will also use another 32-bit pipeline register to store the outputs of logical function for next immediate step in SHA-1. This extra pipeline should reduce the delay of feeding the logical function output to the adder.

ACKNOWLEDGEMENT

This research was supported in part by Altera Corp. (M) under research grant Vot. no. 68838.

REFERENCE:

[1] Bosselaers, Govaerts, Vandewalle (1997), "SHA: A Design for Parallel Architectures?". ftp://ftp.esat.kuleuven.ac.be/pub/COSIC/bosselae/parallel.ps.gz

[2] National Institute of Standards and Technology(1995) , "Secure Hash Standard", April 17, USA: http://www.itl.nist.gov/fipspubs/fip180-1.htm

[3] Schneier (1996), " Applied Cryptography" USA: John Wiley & Sons

[4] Altera Inc. (1999), "Device Data Book 1999"

[5] J. Touch, "Performance Analysis of MD5", Proceedings of ACM SIGCOMM'95, Comp. Comm. Review, Vol.25, No. 4, 1995, pp. 77-86. ftp://ftp.isi.edu/pub/hpcc-papers/touch/sigcomm95.ps.Z

[6] K.S. McCurley (1994), "A Fast Portable Implementation of the Secure Hash Algorithm III", Technical Report SAND93-2591

[7] Sherigar, Mahadevan, Kumar, David(1997) "A Pipelined Parallel Processor to Implement MD4 Message Digest Algorithm on Xilinx FPGA" http://dlib.computer.org/conferen/vlsid/8224/pdf/82240394.pdf

APPENDIX

A . SHA-1 Operations for Pipeline Architecture
(following the standard notation in [2])

Partial initial round of first block

Before processing any blocks, the H's are initialized.

Message conversion by Msg_conv

Divide M(i) into 16 words W(0), W(1), . .. , W(15), where W(0) is the left-most word. (a)

$$S^n(X) = (X << n) \text{ OR } (X >> 32-n).$$

For t = 16 to 79 let

$$W(t) = S^1(W(t-3) \text{ XOR } W(t-8) \text{ XOR } W(t-14) \text{ XOR } W(t-16)). \quad (b)$$

Initial and intermediate rounds operations all blocks

Let A = H0, B = H1, C = H2, D = H3, E = H4. (c)

For t = 0 to 79 do

TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t);
E = D; D = C; C = S^30(B); B = A;
A = TEMP; (d)

Final round of all blocks

Let H0 = H0 + A, H1 = H1 + B,
H2 = H2 + C, H3 = H3 + D,
H4 = H4 + E. (e)

Message digest,
MD[159:0] = H0 H1 H2 H3 H4

Ouput of of logical unit and K(t)

First round
f(t;B,C,D) = (B AND C) OR ((NOT B) AND D)
K(t) = 5A827999 hex (K1)
(0 <= t <= 19)

Second round
f(t;B,C,D) = B XOR C XOR D
K(t) = 6ED9EBA1 hex (K2)
(20 <= t <= 39)

Third round
f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D)
K(t) = 8F1BBCDC hex (K3)
(40 <= t <= 59)

Forth round
f(t;B,C,D) = B XOR C XOR D
K(t) = CA62C1D6 hex (K4)
(60 <= t <= 79).

B. State Diagram of Pipeline SHA-1

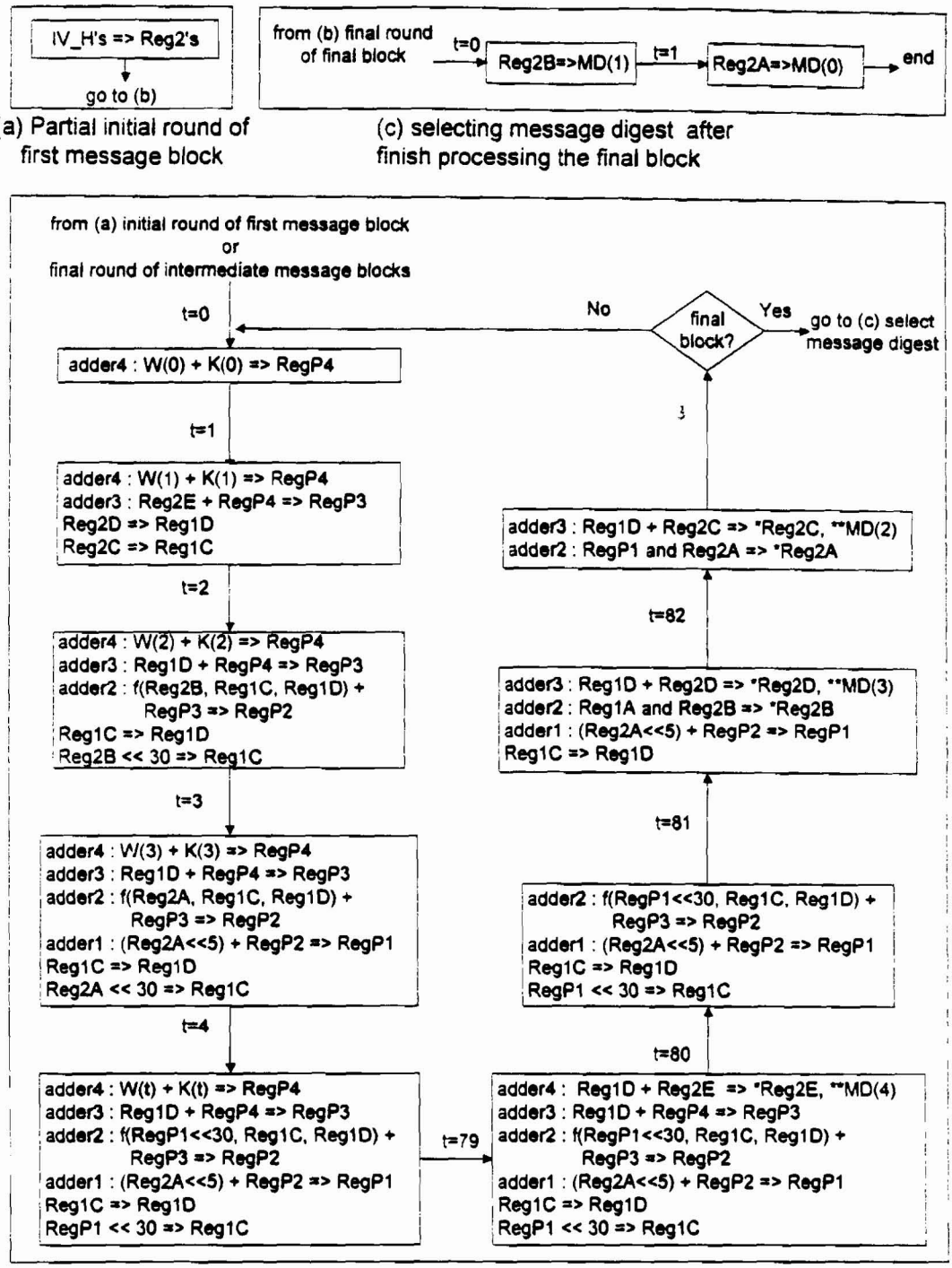
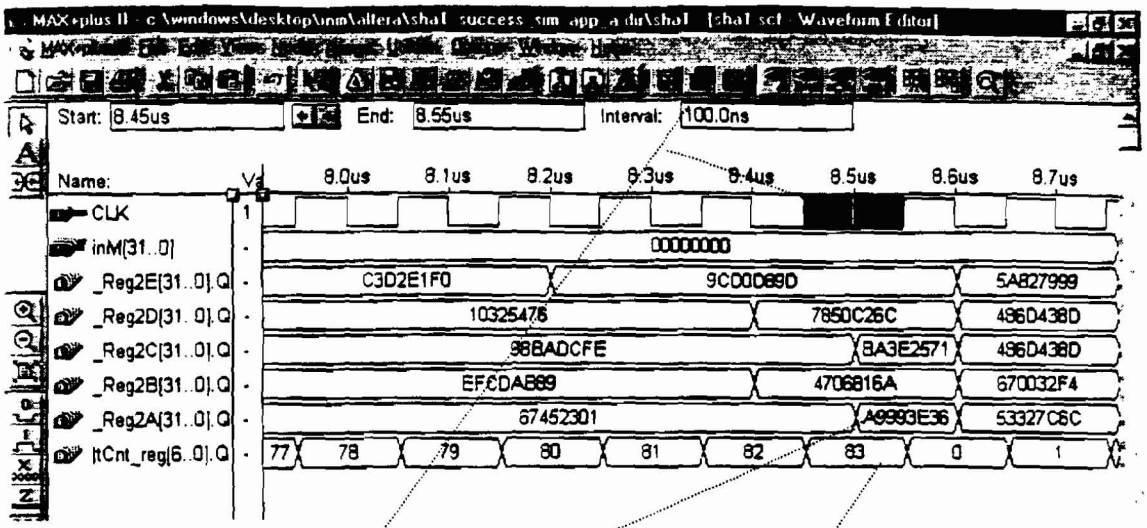


Figure 6 : State Diagram of Pipeline SHA-1

C. Timing Simulation Results of Pipeline SHA-1



Specification of the timing simulation :

Using the test vector from the Appendix A of [2]

Message digest = A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D

Clock frequency = 10Mhz on Flex10KE50 speed –3.

Total clock cycles to process 512 bit message = 84 clock cycles

Speed = 512/ (84*0.1us) bit/s
= 61 Mbps

Figure 7: Timing Simulation Results of Pipeline SHA-1