Chapter 7
VLSI Implementation of Cryptographic Algorithms

7.1 Introduction : Until very recently, all encryption products were in the form of specialized hardware. These encryption/decryption devices are plugged into a communications line and encrypted all the data going across that line. Although software encryption is becoming more popular today, hardware is still the embodiment of choice for military and serious commercial applications. The NSA(National Security Agency) of USA only authorizes encryption in hardware. There are strong reasons for hardware implementation of cryptographic algorithms. As far as 3G algorithms are concerned, in the User Equipment (UE),the algorithm may be implemented as hardware ,while in the Radio Network Controller(RNC) it may also be implemented in software on a general purpose processor. For hardware implementations, the working assumption was such that it should be possible to implement one instance of the algorithm using less than 10000 gates [89]. Kernel algorithm KASUMI has been designed so that it can be efficiently implemented in hardware. Moreover, we want to compare DSP and VLSI implementations of three algorithms KASUMI,f8 and f9 and hence this chapter describes the VLSI implementation using Xilinx and Altera synthesis tools.

7.2 Advantages of Hardware Implementation: Speed is the most important advantage of hardware implementation. Encryption algorithms consist of many complicated operations on plaintext bits. These operations cannot run efficiently on general purpose processors. While some cryptographers have tried to design algorithms more suitable for software implementation, specialized hardware always wins speed race. Additionally, encryption is often a computation intensive task. Tying up computer's primary processor for this purpose is inefficient. Moving encryption to another chip (it can be another processor even) makes the whole system faster.

Security is another reason for hardware implementation. An encryption algorithm running on a generalized computer has no physical protection. Anyone

can access and modify the algorithm with various debugging tools without anyone ever realizing it. Hardware encryption devices can be securely encapsulated to prevent this.

7.3 Previous work on Hardware Implementation of 3G Security Algorithms :

Several research papers describe the work done in the area of Hardware (VLSI) implementation of KASUMI algorithm [38], [46]. Earlier publications related to hardware implementations of block ciphers were related to the Data Encryption Standard (DES) or to the block cipher IDEA. Speeds of IDEA implementations ranging from 2.8 Mbps to 528 Mbps [46] have been reported. This high throughput could only be achieved in a full pipeline architecture requiring up to four Xilinx XC4000 FPGAs. DES implementations delivering up to 400 Mbps have also been shown. Extended performance evaluation of the five AES algorithm finalists have been reported by C.Paar et al.Throughput varying between 126 Mbps (RC6) and 444 Mbps(Serpent) under different optimization constraints were mentioned. By introducing pipeline stages in the round functions itself, throughput in the range of 7.5 to 16.8 Gbps were demonstrated for the block ciphers Twofish, Rijnadael, Serpent and 3DES.However,throughput beyond the gigabit barrier is only possible under extensive use of pipeline techniques.

When compared to other modern block ciphers such as Rijndael, KASUMI presents a rather regular structure which allows for several optimization approaches. In a simple implementation , encryption will be done by just feeding the input in the Feistel-network and looping it back eight times to achieve the total required eight rounds. This approach is not very efficient in terms of throughput. Since even and odd rounds differ only in the order of execution of the functions FO and FL, one could implement one type of both and feed the result backward to compute other type. Combining an even and an odd round of KASUMI in a single combinational unit allows us to calculate two rounds of KASUMI within one single clock cycle. This significantly reduces the amount of iterations and is known as loop unrolling. Besides increasing throughput, loop

unrolling also decreases the maximum possible clock frequency since it introduces longer critical paths.

The high area requirement is principally due to the substitution boxes. This number can be reduced by an additional circuitry that enables substitution box sharing between rounds. But the high additional wiring that this implies is not worth the effort. Further, the full loop unrolled architecture essentially requires a full asynchronous design which is a very tedious task. KASUMI algorithm was specified to be used in counter and output feedback mode within f8 and f9 functions.

In one of the research papers on Low-power UMTS Encryption [46], the authors present an architecture which presents the best throughput/area ratio since only 24 substitution boxes were required instead of 96 for a full loop unrolled architecture. In this implementation, total gate count using this architecture varies between 17000 and 8000 gates depending on the architecture of the underlying substitution boxes.

Another research paper by Tomas Baldreas et. al [44] describe an efficient FPGA architecture for block ciphering in Third Generation Cellular Networks. This paper presents a novel hardware implementation of the KASUMI block cipher using the principle of reuse of components. The architecture is a good balance between high performance and low complexity in area as a result of taking advantage of certain features present in modern FPGAs and some design strategies. The main features of the architecture proposed in this paper are: reuse of higher-level components of the block cipher, which reduces the number of total cycles needed to carry out the process, mapping of the S-boxes to embedded dual port memory blocks and the design of a simple key scheduler that takes advantage of a clock –division technique.

7.4 Previous work on VLSI Implementation of IDEA

One research paper published by Oskar Mencer et al [36] from Stanford University describe hardware software tri-design of encryption for mobile communication units. This paper explores the design space of Field Programmable Gate Arrays ,Processors and ASICs-Hardware-Software Tri

design in the framework of encryption for hand-held communication units. IDEA is used to show the tradeoffs for these technologies. The parameters chosen for comparison of different options are performance, programmability and power.

The conclusions drawn in this paper are as follows:

(i) Power consumption is directly proportional to frequency of the circuit. Hence the technology with the highest MOPS/Watt and Mbps/Watt rating yields the lowest power consumption for a given bit rate.

(ii) DSP TI TMX320C6x gives 53.1 Mbps speed and DSP DEC SA-110 gives 32 Mbps speed.

(iii) FPGA implementation on XC 4000 XL gives 528 Mbps speed whereas ASIC implementation on "VINCI" gives 180 Mbps speed.

7.5 : VLSI Implementation of KASUMI,f8 and f9: KASUMI, f8 and f9 algorithms are coded in VHDL . The simulation is done in Modelsim and synthesis is done using Xilinx and Altera synthesis tools.

7.5.1 KASUMI Implementation :

The KASUMI algorithm consists of the following blocks:

- Key scheduler generates eight 16-bit sub keys (KI1, KI2, KI3, KO1, KO2, KO3, KL1 and KL2) according to the 128-bit ciphering key for eight stages (e.g. the first L sub key of the fifth round is KL51). Therefore total 64 sub keys are created.
- Odd and even stages include FL function, FO function, and 32-bit XOR operation. The stages are completely similar, except the order of the functions which is reversed in the next stage.
- FL function divides the 32-bit input into two 16-bit data paths. The block utilizes the KL sub keys and basic mathematical operations like AND, OR, and XOR. The block contains only combinational logic.
- FO function divides the 32-bit input into two 16-bit data paths. The block utilizes KO sub keys, FI functions, and XOR operations. The block contains only combinational logic.
- FI function is a sub function of the FO function. Each FO function includes three FI functions and six XOR operations. The FI function divides the 16-

bit input into 7-bit and 9-bit wide data paths. The block utilizes FI sub keys that are divided into 7-bit and 9-bit wide sub keys, S7 and S9 boxes, and XOR operations. The block contains only combinational logic.

- S7 and S9 boxes are utilized by the FI function. They are practically look-up tables that produce a 7-bit (9-bit) output from a 7-bit (9-bit) input. The boxes are fully combinational.

7.5.2 Synthesis results for KASUMI:

The test data and summary of the synthesis results are given below:

Key : 2B D6 45 9F 82 C5 B3 00 95 2C 49 10 48 81 FF 48
Input: EA 02 47 14 AD 5C 4D 84
Output: DF 1F 9B 25 1C 0B F4 5F

Key schedule:

Sub. Keys	1	2	3	4	5	6	7	8
KLi1	57AC	8B3E	058B	6601	2A59	9220	9102	FE91
KLi2	0B6E	7EEF	6BF0	F388	3ED5	CD58	2AF5	00F8
KOi1	B3E8	58B0	6016	A592	2209	1029	E91F	7AC5
KOi2	1049	8148	48FF	D62B	9F45	C582	00B3	2C95
KOi3	2910	1FE9	C57A	E8B3	B058	1660	92A5	0922
Kli1	6BF0	F388	3ED5	CD58	2AF5	00F8	0B6E	7EEF
Kli2	7EEF	6BF0	F388	3ED5	CD58	2AF5	00F8	0B6E
Kli3	CD58	2AF5	00F8	0B6E	7EEF	6BF0	F388	3ED5

Table 7.1 –KASUMI test data

KASUMI Synthesis Results

1. Chip: Altera

   Family: APEX20KC

   Device: EP20K1000CB652C7

   Total logic element: 6715/38400(17%)

   Total pins: 150/488(30%)

   Speed: 32 MHz

   Throughput: 2048 Mbps

2. Chip: Xilinx
   Family: Spartan3
   Device: xc3s1500l-4fg676
   No. of slices: 4037/13312(30%)
   No. of slices FF: 1193/26624(4%)
   4 input LUTs: 7787/26624(29%)
   Bonded Tobs: 150/487(30%)
   Speed: 50 MHz
   Throughput: 3200 Mbps

7.5.3 f8 Implementation: The interface has separate 64-bit wide data buses for input and output bit streams. The write enable must be set active during the initialization procedure. The operation of the f8 function has been made very simple. After the reset has been deasserted, the f8 interface waits for the active write enable. After the write enable has been set active, the interface goes to the receive state. In the receive state the IF reads count, bearer, direction of transmission, confidentiality key, and message length during the next four clock cycles. After necessary parameters have been read, the interface initializes the KASUMI core with the modified keys and the feedback register is initialized as well. Then one KASUMI operation is performed to the feedback register. After the feedback register has been updated, the interface initializes the KASUMI core again using the confidentiality key. After a block has been processed, the results are sent to the output bit stream. At this moment, the interface sets the interrupt signal (INT) active when the encoded/decoded block can be read from the output bit stream (OBS).

7.5.4 Synthesis results for f8

The test data and summary of the synthesis results are given below:

Key = 2BD6459F82C5B300952C49104881FF48

Count = 72A4F20F

Bearer = 0C

Direction = 1

Length = 798 bits

fout Plaintext: 7EC61272743BF161 4726446A6C38CED1 66F6CA76EB543004
4286346CEF130F92 922B03450D3A9975 E5BD2EA0EB55AD8E
1B199E3EC4316020 E9A1B285E7627953 59B7BDFD39BEF4B2
484583D5AFE082AE E638BF5FD5A60619 3901A08F4AB41AAB 9B134880

Initial A = 72A4F20F64000000
Key used = 7E8310CAD790E655C0791C451DD4AA1D
Modified A = 34222BC8F7C39416
Key now = 2BD6459F82C5B300952C49104881FF48

BLKCNT	Kasumi input	Keystream	enc/dec data
0	34222BC8F7C39416	AF24CC029AC39D08	D1E2DE70EEF86C69

Table 7.2 : f8 test data

f8 Synthesis results

1.Chip: Altera
Family: APEX20KC
Device: EP20K1000CB652C7
Total logic element: 8113/38400(21%)
Total pins: 132/488(27%)
Speed: 32 MHz
Throughput: 218 Mbps

2. Chip: Xilinx
Family: Spartan3
Device: xc3s1500l-4fg676
No. of slices: 4500/13312(33%)
No. of slices FF: 1943/26624(7%)
4 input LUTs: 8586/26624(32%)
Bonded Tobs: 132/487(27%)
Speed: 50 MHz
Throughput: 340 Mbps

7.5.5 f9 Implementation:

The interface is again as simple as possible. The interface has separate 64-bit input bit stream and 32-bit MAC buses. The write enable must be active during the initialization procedure. The operation of the f9 algorithm is almost similar to the f8 algorithm. After the reset has been deasserted, the algorithm waits for the active write enable. After the write enable has been set active, the algorithm reads the necessary initialization data during the next seven clock cycles (count, fresh, length, and integrity key). After the message has been processed, the KASUMI core is initialized with the modified key and the contents of the MAC register (register B) is selected. Then only a single KASUMI operation is performed. After the operation has been performed, the interface asserts the interrupt signal, and the MAC can be read from the output bit stream.

7.5.6 Synthesis results for f9

The test data and summary of the synthesis results are given below:

Key = 2BD6459F82C5B300952C49104881FF48
Count = 38A6F056
Fresh = 05D2EC49
Direction = 0
Length = 189 bits
Message:
6B227737296F393C 8079353EDC87E2E8 05D2EC49A4F2D8E0
New Key: 817CEF35286F19AA3F86E3BAE22B55E2
final step: F1BEEC15B964E3F2 F63BD72C702EBC7A
MAC-I: F63BD72C

Table 7.3 : f9 test data

Synthesis Results:

1.Chip: Altera

Family: APEX20KC

Device: EP20K1000CB652C7

Total logic element: 7937/38400(20%)

Total pins: 100/488(27%)

Speed: 32 MHz

Throughput: 217 Mbps

2. Chip: Xilinx

Family: Spartan3

Device: xc3s1500l-4fg676

No. of slices: 4439/13312(33%)

No. of slices FF: 1847/26624(6%)

4 input LUTs: 8471/26624(31%)

Bonded Tobs: 100/487(20%)

Speed: 50 MHz

Throughput: 339 Mbps

7.6 Implementation of IDEA: The IDEA algorithm was implemented using ALTERA's QUARTUS tool for VLSI synthesis. The major blocks implemented are

(i) 16 bit XOR: Bit by Bit exclusive OR

(ii) 16 bit adder: Addition of integers modulo 2^{16} with inputs and outputs treated as unsigned 16 bit integers

(iii)Modulo 16 multiplier: Multiplication of integers modulo 2^{16} +1 with inputs and outputs treated as unsigned 16-bit integers ,except that a block of all zeros is treated as representing 2^{16} .

The results for implementation are given below:

Fitter report for ide8rnd
Thu Aug 18 18:59:12 2005
Version 4.1 Build 181 06/29/2004 SJ Full Version
Fitter Summary

Fitter Status	Successful - Thu Aug 18 18:59:11 2005
Quartus II Version	4.1 Build 181 06/29/2004 SJ Full Version
Revision Name	ide8rnd
Top-level Entity Name	ide8rnd
Family	APEX20KE
Device	EP20K1500EBC652-1
Timing Models	Production
Total logic elements	43,249 / 51,840 (83 %)
Total pins	257 / 488 (52 %)
Total memory bits	0 / 442,368 (0 %)

7.7 Simulation Waveforms for KASUMI, f8 and f9: Modelsim simulation waveforms for timing analysis of KASUMI, f8 and f9 are drawn here. They have been used to determine latency and throughput for various algorithms.

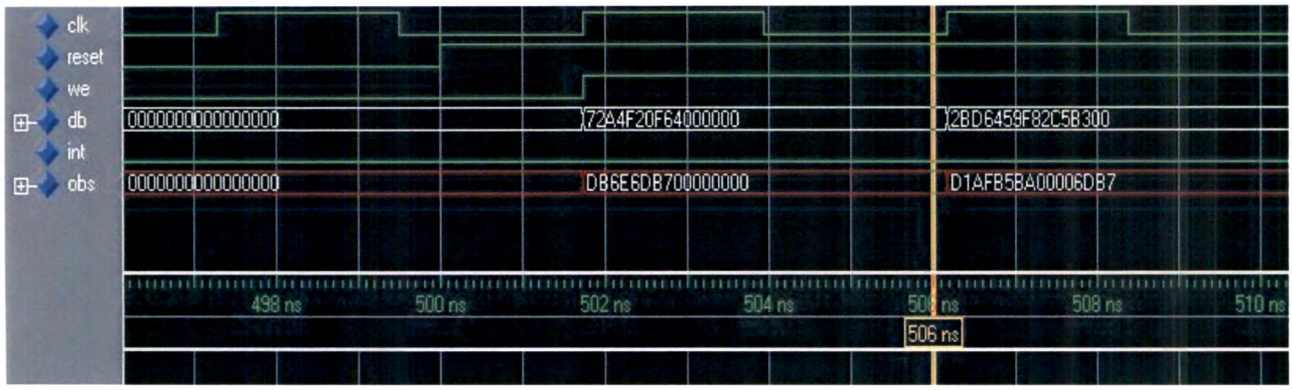

Figure 7.1(a) Simulation waveforms for f8 (Initial A)

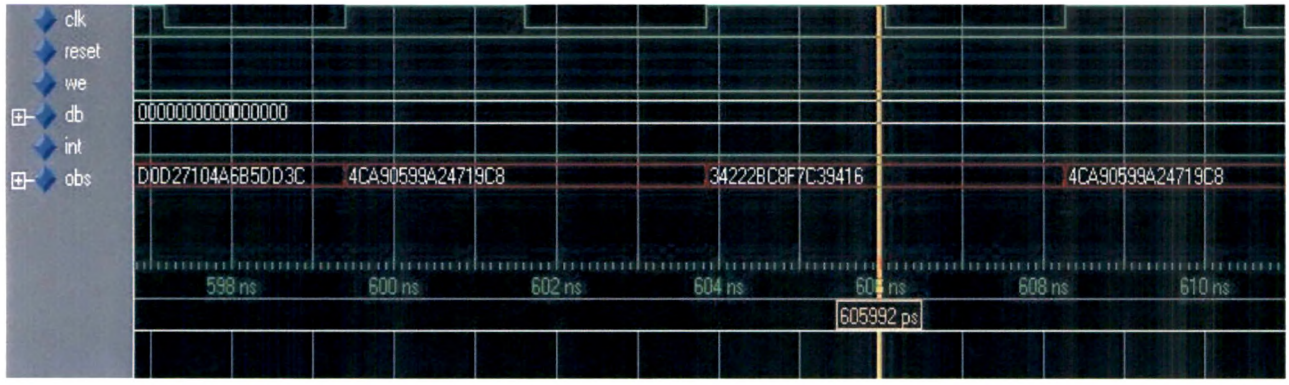

Figure 7.1(b) Simulation waveforms for f8 (Modified A)

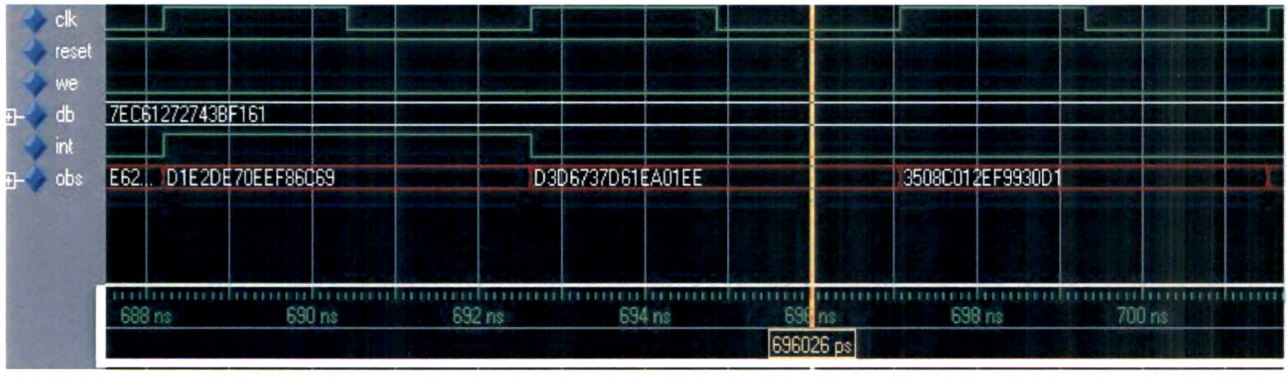

Figure 7.1(c) Simulation waveforms for f8 (Plain text stream)

Figure 7. 1(d) Simulation waveforms for f8 (Cipher text stream)

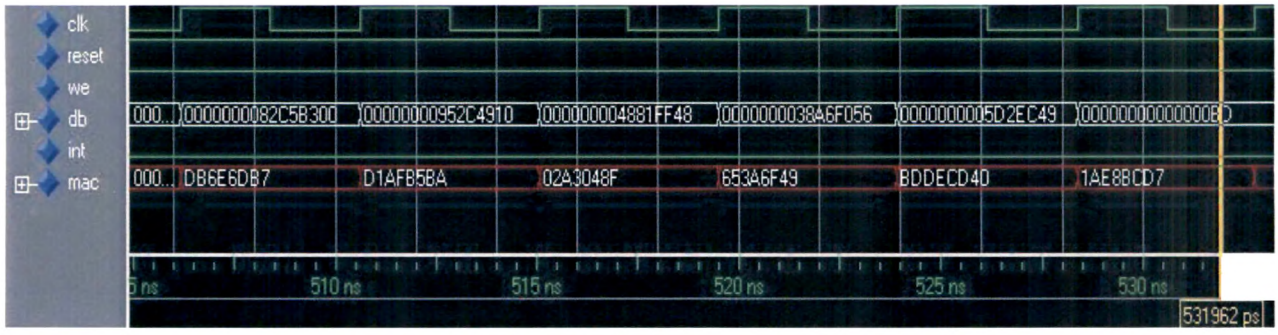

Figure 7. 2(a) Simulation waveforms for f9 (Count Fresh)

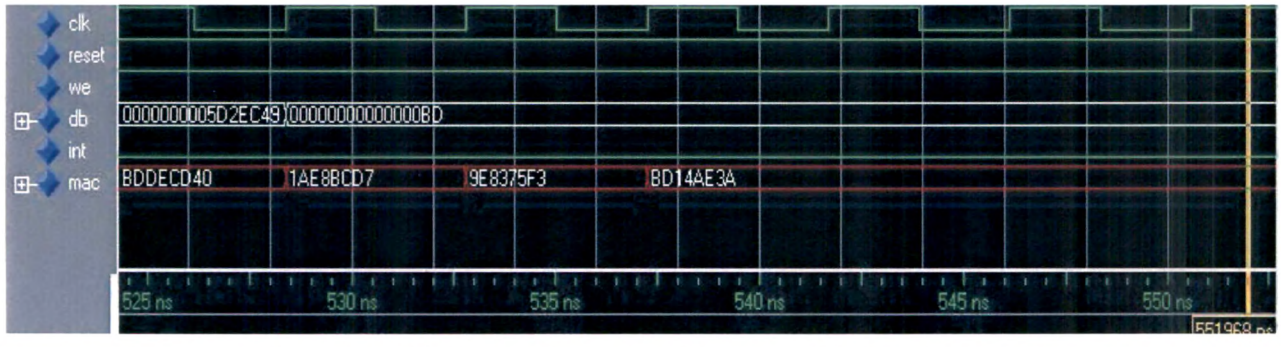

Figure 7.2(b) Simulation waveforms for f 9 (Message)

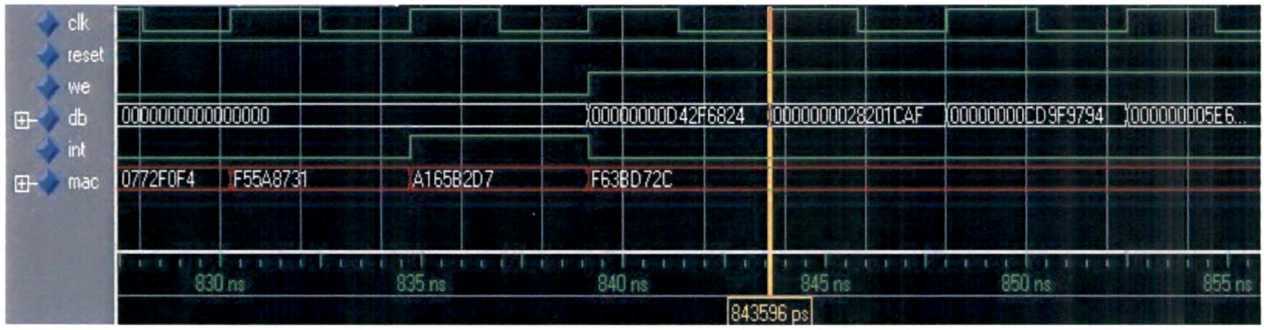

Figure 7. 2 (c) Simulation waveforms for f 9 (MAC – I)

Figure 7. 2(d) Simulation waveforms for f9 (MAC – I)

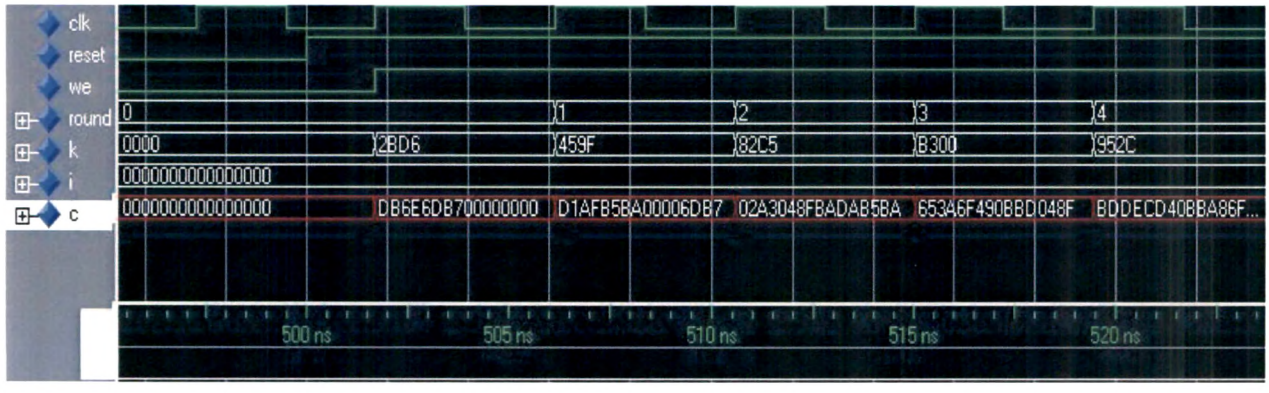

Figure 7. 3(a) Simulation waveforms for KASUMI (Subkeys)

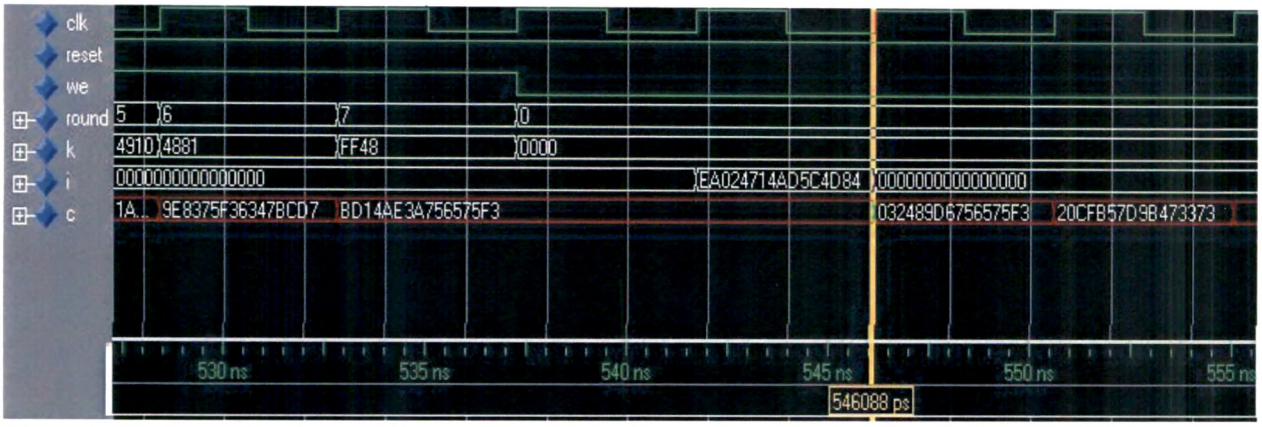

Figure 7. 3(b) Simulation waveforms for KASUMI (Plain text)

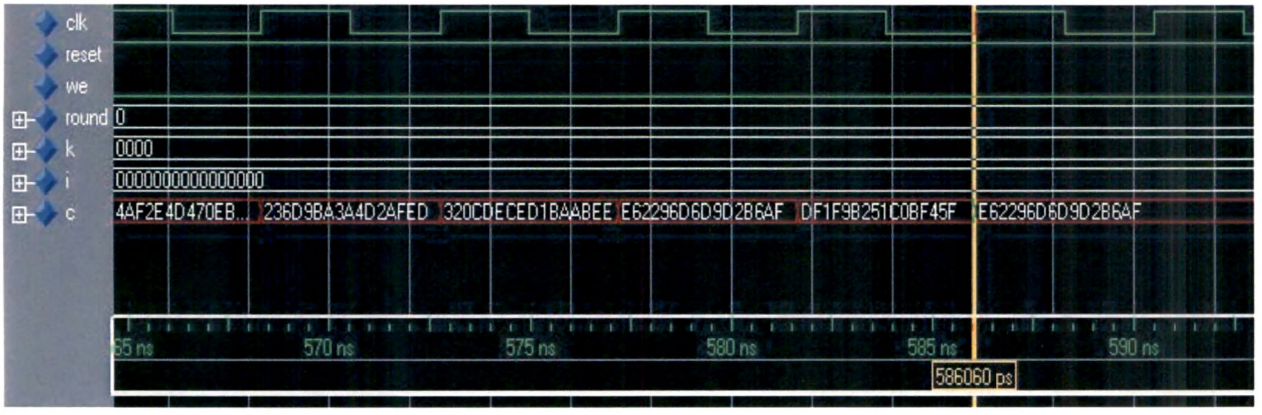

Figure 7.3 (c) Simulation waveforms for KASUMI (Cipher text)

7.8 Throughput :The throughput availability for KASUMI ,f8 and f9 algorithms are summarized in the following table:

Algorithm	**Throughput in Mbps**	
	Xilinx	Altera
KASUMI	3200 (50 MHz)	2048 (32 MHz)
f8	340 (50 MHz)	218 (32 MHz)
f9	339 (50 MHz)	217 (32 MHz)
IDEA	-----	2900 (40 MHz)

Table 7.4 : Throughput for KASUMI, f8 and f9 algorithms

7.9 Estimation of ASIC area: The relative merits and demerits of ASIC, FPGA and DSP implementations were explored. Before taking up the research work, a research paper based on software V/s ASIC implementation was studied by us. This research work was taken up by NOKIA. It was found that ASIC solution performance suffers from relatively large overhead in inter-process communication and operating system context switches. The ASIC solution involves a lot of signaling between the application process and the ASIC driver process. The software ciphering performed on DSP does not have any of these overheads because all the processing is done inside the application process. Software ciphering improves performance for speech traffic ciphering. It also simplifies the architecture because there is no need for HW-SW interface. It has also got faster design cycle. If we talk about specific results, software ciphering on DSP consumes about half of the time used by ASIC for speech traffic. There is no significant difference for data traffic.

Comparison of ASIC and FPGA design leads to following facts:

➔ ASIC requires about 108 mm^2 for IDEA encryption as against 3200 CLBs of XC4000 XL FPGA[14].

➔ A rough estimation of area calculation for ASIC design can be done based on the library attributes and a previously constructed standard library dimensions. Total combinational area is 7390 units and total (sequential) non combinational area is 32340 units for IDEA implementation on ASIC. A unit area is equivalent to physical area covered by 2-input NAND gate.

➔ Depending on the technology that is chosen for fabrication of the device, the actual value of the area can be calculated by multiplying the unit area values with the area covered by a 2-input NAND gate.

7.10 Summary: In this chapter, we have discussed VLSI Implementation of f8, f9, KASUMI and IDEA algorithms on Xilinx and Altera synthesis tools. The throughput availability is very good and is much larger than UMTS chip rate of 3.84 Mcps.