Chapter 6
Software Implementation of Cryptographic Algorithms

6.1 Introduction: Any encryption algorithm can be implemented in software. While implementing in software, the disadvantages are in speed, cost and ease of modification and the advantages are in flexibility and portability, ease of use and ease of upgrade. The algorithms written in any programming language can be implemented on any computer. They can be inexpensively copied and installed on many machines .They can be incorporated into larger applications. They are popular and are available for all major operating systems.

This chapter describes software implementation of cryptographic algorithms using MATLAB and DSP. AES, KASUMI, f8 and f9 algorithms are implemented in MATLAB. Same algorithms are also implemented in DSP environment . IDEA is implemented in DSP environment alone. Comparisons are made for various parameters like encryption speed and memory usage. MATLAB and DSP profiling is done to investigate the functions at micro level.

6.2 Previous work on Software Implementation of 3G Cryptographic Algorithms: Though there are very few research papers describing MATLAB/DSP implementation of 3G security algorithms, we look at these papers in this section.

6.2.1 MATLAB/SIMULINK Implementation: MATLAB is a matrix oriented programming language which is perfectly suited for data structure of many algorithms like the AES. Moreover, it has a Code Composer Link which can load the CCS project on DSP target for emulator analysis. MATLAB can further lead to building of SIMULINK blocks which are suitable for system level studies. Researchers at Institute of System Level Integration, Livingston, UK have designed a video encryption system [43] which uses SIMULINK block for AES algorithm. It uses Xilinx blockset SIMULINK library and is a powerful tool which links system level simulation to VLSI implementation on FPGA.

There were probably no research papers describing implementation of 3G security algorithms on MATLAB/SIMULINK when we started our work. This had motivated us to use this powerful language to study encryption/decryption algorithms and build SIMULINK blocks for integrating them with system architecture blocks.

6.2.2 DSP Implementation : Digital Signal Processors are a highly attractive option for software implementations of the cryptographic algorithms like Advanced Encryption Standard (AES) .These DSPs perform certain arithmetic operations at high speeds ,they are often smaller and more energy-efficient than general purpose processors. However, the research done on implementation of cryptographic algorithms on a DSP is limited. There are a few papers that deal with public-key cryptography. [20] , [23],[25] .The main conclusion of these papers is that DSPs are a good choice for these algorithms due to integer arithmetic capabilities of DSPs. One research paper by Thomas J.Wollinger et al investigates suitability of high end DSPs for AES algorithms. The paper describes memory usage and CPU cycles as well as speed in Mbps for AES candidates like Twofish, RC6, Rijnadael ,Mars and Serpent. The paper concludes that the AES finalists' encryption and decryption functions reach higher speeds on the C6000 DSPs than the best known Pentium Pro II implementations, at identical clock rates.

Research work done at Nokia Networks on Performance Evaluation of software ciphering in UMTS Radio Network Controller [12] discusses about pros and cons of ASIC ciphering and software ciphering. The software ciphering was done on Texas Instruments' TMS 320C55x family processors. It was concluded that the software ciphering is significantly faster than that on ASIC for speech call simulation test.

To the best of our knowledge, research work on the DSP implementation of 3G security algorithms like KASUMI, f8, f9 and cryptographic functions f1 to f5 was minimal . This has motivated us to take up the task of DSP implementation on Texas Instrument's DSP TMS 320C6713. Our studies have found DSP implementation to be very much promising when wireless communications extensively use it for other tasks like speech processing and smart antennas.

6.3 MATLAB Implementation of 3G Security Algorithms : We have first implemented the Advanced Encryption Standard (AES) algorithm on MATLAB platform. Basically it is software ciphering and our aim is to study the role of individual functions in the entire process of encryption. This analysis is done using profile function available under the category of profiling m-files.

6.3.1 AES program

The Matlab program aes_demo calls the functions cipher, inv _cipher , aes_init to perform various tasks. The first task of the "main" program aes_demo is a call to aes_init . This initialization routine provides the actual encryption and decryption functions (cipher and inv_cipher) with the expanded key schedule w, the substitution tables s_box and inv_s_box, and the polynomial matrices poly_mat and inv_poly_mat. These quantities have to be generated only once and can be used by any subsequent encryption or decryption call. This input block is passed to the encryption function cipher, which returns the corresponding 16 bytes(128 bits) of ciphertext. In order to demonstrate the decryption process too, the ciphertext is then forwarded to the decryption function inv_ cipher , resulting in the reprocessed plaintext block re_ plaintext, which can be compared to and certainly has to be equal the original plaintext.

Before doing any actual encryption or decryption, the initialization function aes_init has to be called once. aes_init generates the two substitution tables s_ box and inv_ s_ box by a call to s_ box_ gen , defines the round constant vector rcon and an exemplary key and computes the expanded key schedule w . Additionally the two polynomial matrices poly_ mat and inv_ poly_ mat are generated .

The substitution tables (S-boxes) s_ box and inv_ s_ box are used by the expanded key schedule function key_ expansion and the encryption and decrypting functions cipher and inv_ cipher to directly substitute a byte by another byte of the same finite field.

The function s_ box_ gen creates the S-boxes by searching for the inverses of all elements of GF(28) (Lines 4 . . . 6) by the use of find_ inverse and by applying transformations to all inverses via aff_ trans . Finally the inverse S-Box inv_ s_ box, to be used in inv_ cipher, is constructed from s_ box by s_ box_ inversion .

The first step in the S-box generation process is to search for the multiplicative inverses of all elements of the finite field GF(28). The algorithm itself simply loops through all possible byte values and test-wise computes the product of the byte to be inverted (b_ in) and the current test candidate . If the product equals 1 , the inverse is found and the loop breaks.

The inverse S-box is used in the decrypting function inv _cipher to revert the substitution carried out via the S-box. The corresponding AES- Matlab function takes the S-box (s_ box) as its input and generates the inverse S-box inv_ s_ box in a single loop . The loop runs through all elements of the S-box, interprets the current S-box element value as an index into the inverse S-box and inserts the values 0 . . . 255 at the appropriate places in the inverse S-box .

The round constant matrix is used in the key expansion scheme (key_ expansion). It is a 10×4 matrix of zeros except for the first column, which contains byte-conform powers of 2 .The key expansion function takes the user supplied 16 bytes long key and utilizes the previously created round constant matrix rcon and the substitution table s_box to generate a 176 byte long key schedule w, which will be used during the encryption and decryption processes.

The function cipher is doing the actual encryption of the 16 byte(128 bits) long input vector of plaintext into the output ciphertext vector .Further input parameters of cipher, that have been created by the initialization function aes_ init, are the substitution table s_ box, the key schedule w, and the polynomial matrix poly_ mat. The decryption function inv_ cipher step-by-step reverses the transformations of the encryption process. The input parameter of inv_ cipher are the ciphertext to be decrypted (back) into plaintext, the inverse S-box (inv_ s_ box), the key schedule w, and the inverse polynomial matrix inv_ poly_ mat.

The encryption and the decryption functions are very much similar.

6.3.2 KASUMI, f8 and f9 Implementation in MATLAB and SIMULINK : The KASUMI function is coded in MATLAB.f8 and f9 functions are coded in MATLAB and their S-functions allow us to use these two cryptographic algorithms as SIMULINK blocks. The purpose of making SIMULINK blocks is to take up system level studies of 3G mobile systems.

The arguments of the function f8 is data, key, count, bearer and direction. Here the input data is in the range of 1 to 53 bits. Data is in the form of array of 8 bits .64 bit data is broken into 8 elements of 8 bits. Key is 128 bits long which is given as row vector of 16 elements with each element of 8 bits. A sample output of the program for algorithm f8 is given below:

Original Message (Plaintext) : 0x04585654575af45d5f52cf56

Key	Encrypted Data
[16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1]	[205 145 159 157 158 147 61 148]
[250 2 1 3 6 5 1 22 21 20 3 23 20 1 8]	[131 223 209 211 208 221 115 218]
[255 0 1 2 3 52 87 9 10 2 5 120 22 2 1 5]	[112 44 34 32 35 46 128 41]
[52 62 6 56 5 5 52 56 54 57 58 1 2 3 8 1]	[65 29 19 17 18 31 177 24]
[10 20 2 2 2 3 6 9 7 5 225 2 62 52 55 3]	[3 95 81 83 80 93 243 90]

Table 6.1 : MATLAB implementation of function f8-sample output

The f9 function arguments are message ,key ,fresh, count and direction. The data is again in the form of array of 8 bits. Again, key is given as row vector of 16 elements with each element being 8-bits.The MAC (Message Authentication Code) is generated which is 32 bits long .A sample output of the program is given below:

Message	MAC
[41 45 58 98 96 63 255 250 1 2 0]	A24CFDB2
[250 2 1 3 6 5 1 22 21 20 3 23 20 1 8]	F8E66DEF
[255 0 1 2 3 52 87 9 10 2 5 120 22 2 1 5]	896AE0AD
[52 62 6 56 5 5 52 56 54 57 58 1 2 3 8 1]	537D6C0B
[10 20 2 2 2 3 6 9 7 5 225 2 62 52 55 3]	8B84E3F1

Table 6.2 MATLAB implementation of function f9-sample output

The Simulink blocks for functions f8 and f9 are shown here. They represent the encryption functions as independent blocks which can be integrated with other communication blocks for simulation of 3G mobile communications.

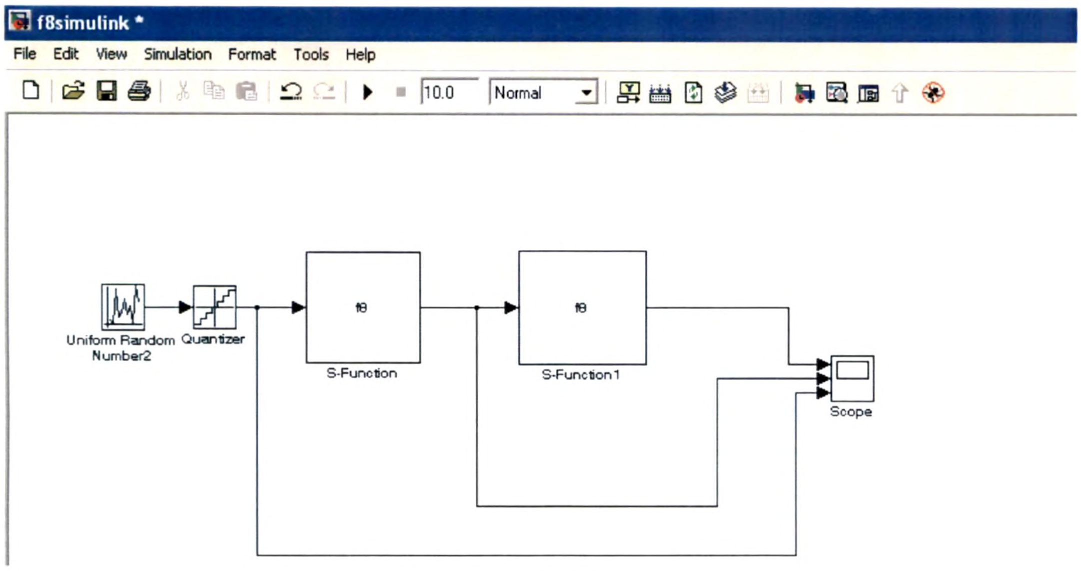


Figure 6.1 Simulink model for f8

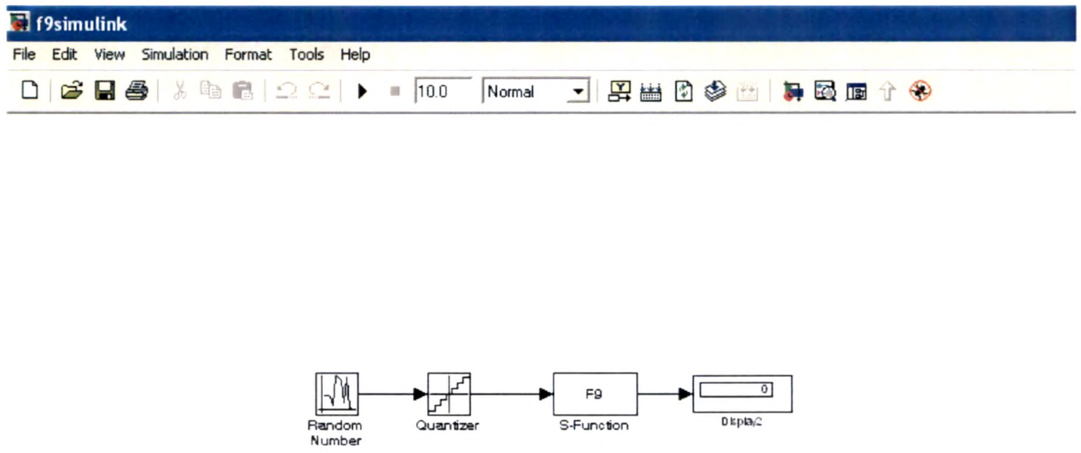


Figure 6.2 Simulink model for f9

6.3.3 Analysis of MATLAB Implementation using Profile function: The profile reports and graphs for AES, f8 and f9 functions are reproduced here:

6.3.3.1 AES Implementation

MATLAB Profile Report: Summary for AES algorithm

Report generated 13-Dec-2005 10:57:42

Total recorded time:	7.98 s
Number of M-functions:	25
Number of M-scripts:	1
Number of MEX-functions:	2
Clock precision:	0.00000005 s
Clock Speed:	2200 Mhz

Function List

Name	Time		Calls	Time/call	Self time	
aes demo	7.98200000	100.0%	1	7.9820000000000	0.03100000	0.4%
aes init	7.63000000	95.6%	1	7.6300000000000	0.00000000	0.0%
s box gen	7.59000000	95.1%	1	7.5900000000000	0.02000000	0.3%
find inverse	7.33000000	91.8%	255	0.0287450980392	1.09200000	13.7%
poly mult	6.37900000	79.9%	34057	0.0001873036380	6.37900000	79.9%
mix columns	0.30100000	3.8%	18	0.0167222222222	0.17000000	2.1%
aff trans	0.23000000	2.9%	256	0.0008984375000	0.01000000	0.1%
bin2dec	0.22000000	2.8%	788	0.0002791878173	0.20000000	2.5%
inv cipher	0.21000000	2.6%	1	0.2100000000000	0.00000000	0.0%
cipher	0.10100000	1.3%	1	0.1010000000000	0.01000000	0.1%
key expansion	0.03000000	0.4%	1	0.0300000000000	0.03000000	0.4%
iscellstr	0.02000000	0.3%	792	0.0000252525253	0.02000000	0.3%
hex2dec	0.02000000	0.3%	4	0.0050000000000	0.01000000	0.1%
fliplr	0.01000000	0.1%	4	0.0025000000000	0.01000000	0.1%
profile	0.00000000	0.0%	1	0.0000000000000	0.00000000	0.0%
inv shift rows	0.00000000	0.0%	10	0.0000000000000	0.00000000	0.0%
etime	0.00000000	0.0%	1	0.0000000000000	0.00000000	0.0%

datenummx	0.00000000	0.0%	4	0.0000000000000	0.00000000	0.0%
shift rows	0.00000000	0.0%	10	0.0000000000000	0.00000000	0.0%
add round key	0.00000000	0.0%	22	0.0000000000000	0.00000000	0.0%
cycle	0.00000000	0.0%	22	0.0000000000000	0.00000000	0.0%
repmat	0.00000000	0.0%	46	0.0000000000000	0.00000000	0.0%
poly mat gen	0.00000000	0.0%	1	0.0000000000000	0.00000000	0.0%
sub bytes	0.00000000	0.0%	30	0.0000000000000	0.00000000	0.0%
rot word	0.00000000	0.0%	10	0.0000000000000	0.00000000	0.0%
cellfun	0.00000000	0.0%	4	0.0000000000000	0.00000000	0.0%
rcon gen	0.00000000	0.0%	1	0.0000000000000	0.00000000	0.0%
s box inversion	0.00000000	0.0%	1	0.0000000000000	0.00000000	0.0%

Table 6.3 MATLAB profile report summary for AES

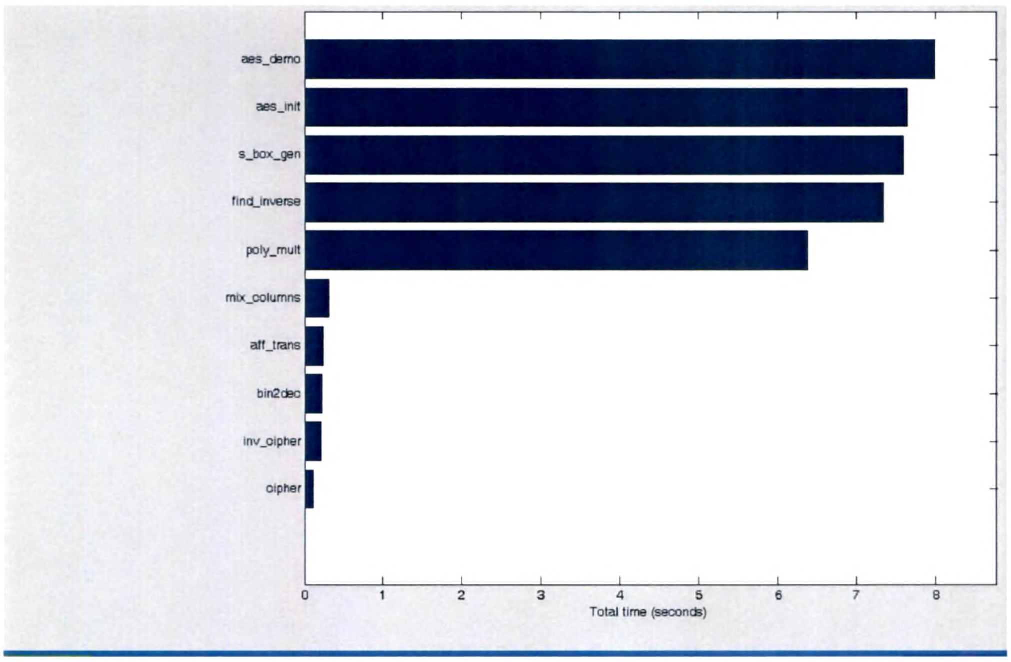


Figure 6.3 MATLAB Profile Graph for AES

6.3.3.2 MATLAB Profiling of function f8:

MATLAB Profile Report: Summary for f8 algorithm

Report generated 13-Dec-2005 16:09:09

Total recorded time:	0.72 s
Number of M-functions:	10
Number of M-scripts:	1
Clock precision:	0.00000005 s
Clock Speed:	2200 Mhz

Function List

Name	Time		Calls	Time/call	Self time	
f8	0.72100000	100.0%	1	0.72100000000	0.18100000	25.1%
hex2dec	0.40000000	55.5%	704	0.00056818182	0.26000000	36.1%
kasumi	0.39000000	54.1%	3	0.13000000000	0.00000000	0.0%
fo	0.29000000	40.2%	24	0.01208333333	0.05000000	6.9%
fi	0.22000000	30.5%	48	0.00458333333	0.03000000	4.2%
fl	0.08000000	11.1%	24	0.00333333333	0.00000000	0.0%
rol16	0.08000000	11.1%	112	0.00071428571	0.03000000	4.2%
keyschedule	0.07000000	9.7%	2	0.03500000000	0.03000000	4.2%
fliplr	0.07000000	9.7%	704	0.00009943182	0.07000000	9.7%
iscellstr	0.07000000	9.7%	704	0.00009943182	0.07000000	9.7%
profile	0.00000000	0.0%	1	0.00000000000	0.00000000	0.0%

Table 6.4 MATLAB profile report summary for f8

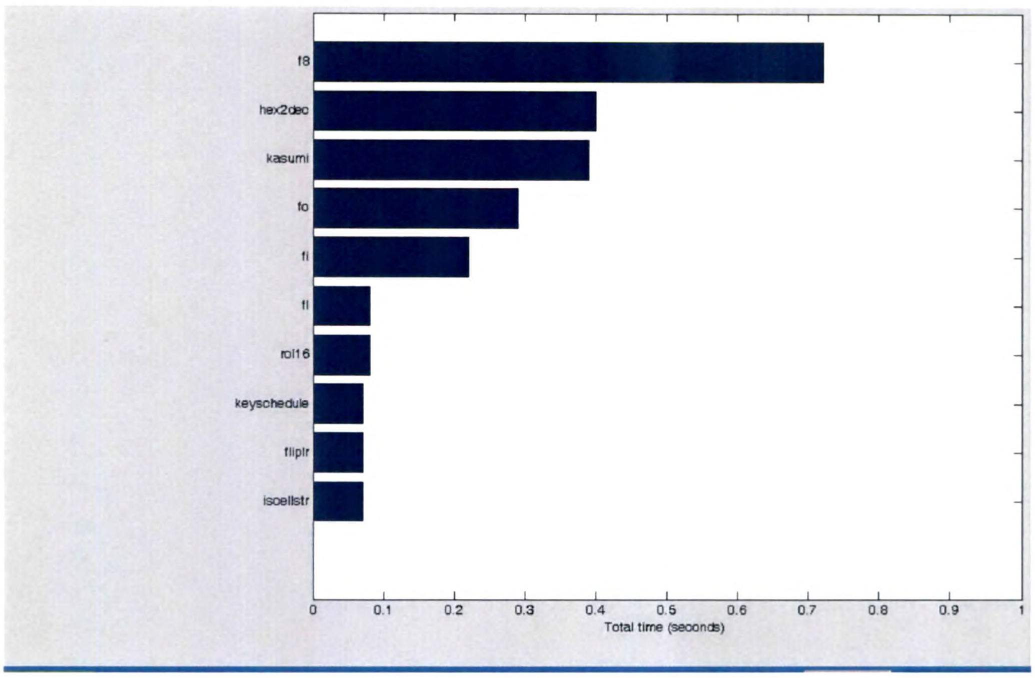


Figure 6. 4 MATLAB Profile Graph for f8

6.3.3.3 MATLAB Profiling of function f9:

Profile Report Summary for f9

Report generated 30-Jan-2006 11:43:56

Total recorded time:	0.14 s
Number of M-functions:	12
Number of M-scripts:	1
Number of M-subfunctions:	1
Clock precision:	0.016 s

Function List

Name	Time		Calls	Time/call	Self time	
f9	0.14	100.0%	1	0.14100	0.02	11.3%
hex2dec	0.11	77.3%	711	0.00015	0.08	55.3%
fi	0.09	66.0%	48	0.00194	0.00	0.0%
fo	0.09	66.0%	24	0.00387	0.00	0.0%
kasumi	0.09	66.0%	3	0.03100	0.00	0.0%
int2str	0.02	11.3%	1	0.01600	0.02	11.3%
dec2hex	0.02	11.3%	1	0.01600	0.00	0.0%
keyschedule	0.02	11.3%	2	0.00800	0.00	0.0%
iscellstr	0.02	11.3%	711	0.00002	0.02	11.3%
fliplr	0.02	10.6%	711	0.00002	0.02	10.6%
profile	0.00	0.0%	1	0.00000	0.00	0.0%
rol16	0.00	0.0%	48	0.00000	0.00	0.0%
fl	0.00	0.0%	24	0.00000	0.00	0.0%
keyschedule/ROL16	0.00	0.0%	64	0.00000	0.00	0.0%

Table 6.5 MATLAB profile report summary for f9

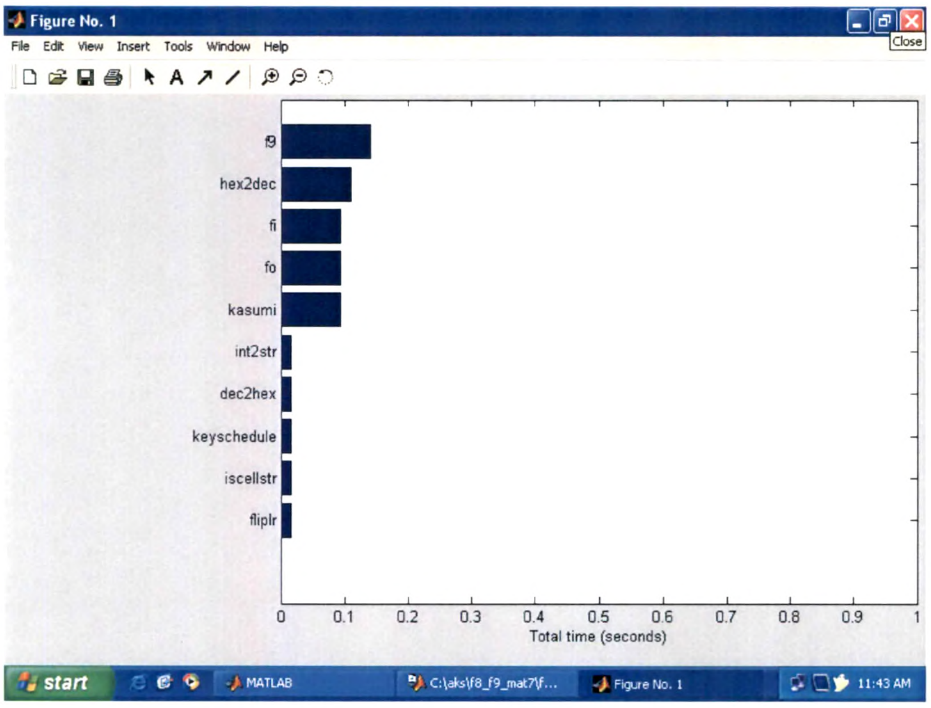


Figure 6.5 MATLAB profile graph for f9

6.4 DSP Implementation of 3G security algorithms: We have implemented the 3G security algorithms in DSP environment and both Simulator and Emulator analysis was taken up for all the algorithms.

6.4.1 Tools for Implementation: We have used Code Composer Studio (v 3.0) development tool for creating a project , building it and loading of program on target DSK TMS 320C6713. The DSK is a standalone development platform that enables users to evaluate and develop applications for TI C67xx DSP family. It is equipped with a full compliment of on-board devices that suit a wide variety of application environments. Key features of this DSK include:

- TMS 320C6713 DSP operating at 225 MHz
- Stereo codec
- 16 Mbytes synchronous DRAM
- 512 Kbytes non-volatile Flash memory
- JTAG emulation through on-board JTAG emulator with USB host interface

6.4.2 The methodology: The source codes for cryptographic algorithms are written in C .The projects are created in Code Composer Studio . The analysis is done in both C6713 functional simulator and on- board JTAG emulator. The simulator and emulator analysis is done for counting CPU clock cycles and memory usage. The C/C++ compiler accepts C/C++ source code and produces assembly language source code. A shell program, an optimizer and an interlist utility are parts of compiler. The optimizer modifies code to improve efficiency of the C programs. We have used optimization of compiler. The optimization levels are selected for most critical speed and size.

We have also used DSP/BIOS Configuration Tool and Real Time Analysis Tool which provides unique visibility into our application by allowing us to probe, trace and monitor a DSP application during its course of execution. The main parameter of concern is CPU Load Graph. This graph displays a graph of the target CPU processing load. The most recent CPU load is shown in the left corner and the highest load is shown in the right corner. CPU load is defined as the amount of time not spent performing the low-priority task that runs when no other thread needs to run. Thus, the CPU load includes any time required to transfer data from the target to the host and to perform additional background tasks. The CPU load is averaged over the polling rate period. The longer the polling period, the more likely it is that short spikes in the CPU load is not shown in the graph.

We have also used Statistics view in DSP/BIOS real time analysis. It displays summary statistics amassed in kernel accumulator objects, reflecting dynamic program elements ranging from simple counters and time-varying data values, to elapsed processing intervals of independent threads. The target program accumulates statistics explicitly through DSP/BIOS API calls or implicitly by Kernel, when scheduling threads for execution or performing I/O operations.

The execution graph displays the execution of threads in real-time. Through the execution graph, we can view the timing and order in which threads are executed. We have taken these graphs for f8,f9, MILENAGE and IDEA algorithms.

Another tool which we have used is CodeSizeTune (CST) that enables us to easily optimize the trade-off between code size and cycle count for our application. Using a variety of profiling configurations, CodeSizeTune profiles our application, collect data on individual functions and determine the best combinations of compiler options. CST then produces a graph of these function-specific option sets. This allows us to graphically choose the configuration that best fits the needs.

The algorithms implemented on DSP platform include KASUMI,f8 f9 , Rijnadael and IDEA. All the cryptographic functions f1 to f5 used for Authentication and Key Generation are also implemented on DSP. This leads to almost entire security mechanism for 3G mobile communication systems built around DSP environment and its performance comparison is simultaneously done with hardware (VLSI) implementation .

6.4.3 DSP Implementation for KASUMI, f8, f9, IDEA, AES (Rijndael) and MILENAGE:

Both simulator and emulator analysis was done with Code Composer Studio v3.0 development tools. Because simulator execute DSP programs on a host computer, the speed is much slower than the program run on the actual DSP processor in real time. The results are as follows:

Algorithm	**CPU clock cycles**	
	Speed most critical	**Size most Critical**
KASUMI	174532	174842
f8 (1024 bit message)	367904	370120
f9	122891	125181
IDEA	133553	135997
AES (Rijndael)	324504	326384
MILENAGE	522115	535313

Table 6. 6 DSP Simulator Analysis Without Optimization

Algorithm	**CPU clock cycles**
KASUMI	169553
f8 (1024 bit message)	295206
f9	47799
IDEA	108188
AES(Rijndael)	298475
MILENAGE	178353

Table 6.7 DSP Simulator Analysis With Optimization Level – O3

Sir No	Length of Message in Bits	Register		Local		Function		File	
		CPU	Total	CPU	Total	CPU	Total	CPU	Total
1	32	55656	76181	53654	74619	51973	72885	51904	72846
2	128	77800	107442	74956	104655	73244	103005	73131	102897
3	1024	312546	436016	298053	418932	296019	419573	295206	416639
4	5120	1392506	1949842	1324765	1872884	1321259	1881053	1317246	1869497

Table 6.8 Different levels of optimization for f8 (simulator analysis)

Sr No	Length of Message in Bits	File level (-03,-pm,no ms) optimization level	
		CPU	Total
1	32	19318	23603
2	128	19911	24304
3	1024	43312	47799
4	5120	150320	155529

Table 6.9 File level optimization for f9 (Simulator analysis) with variable Message lengths.

Algorithm	**CPU clock cycles**	
	Speed most critical	**Size most critical**
KASUMI	239	219
f8	224	219
f9	229	229
IDEA	219	214
AES(Rijndael)	83	64
MILENAGE	104	71

Table 6.10 DSP Emulator Analysis Optimization Level – O3

Algorithm	**Throughput in Mbits/sec**
f8	64.28
f9	62.88
IDEA	67.28
MILENAGE	276.92
KASUMI	60.25
AES(Rijndael)	346.98

Table 6. 11 Encryption throughput calculation based on emulator CPU cycles (speed most critical)

6.4.4 Profile graphs: This section shows the profile graphs for various algorithms using application code tuning.

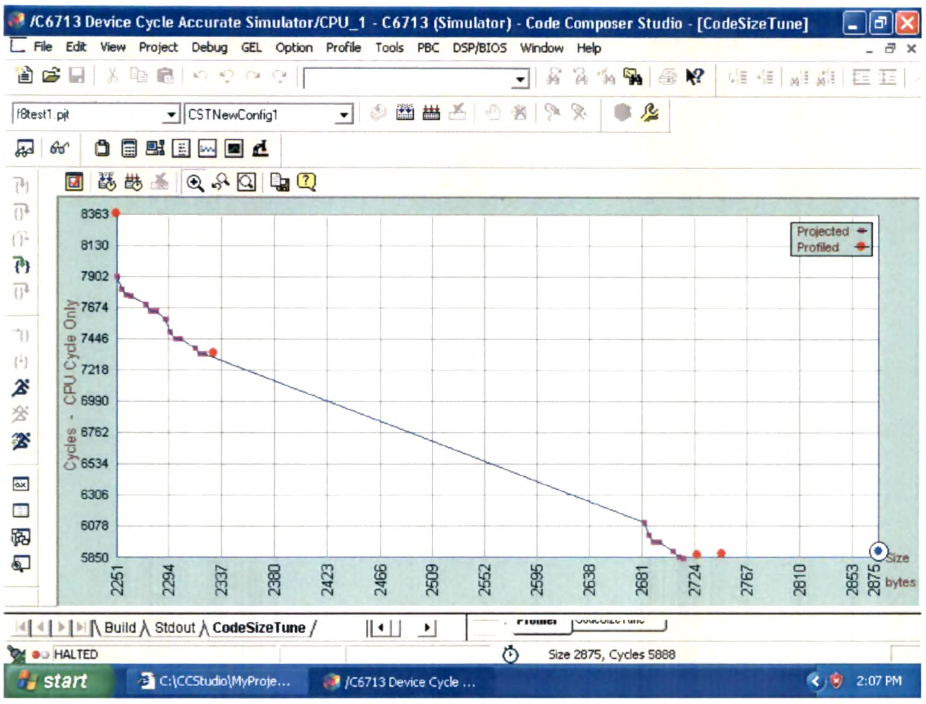


Figure 6.6: DSP Profile Graph for f8 algorithm

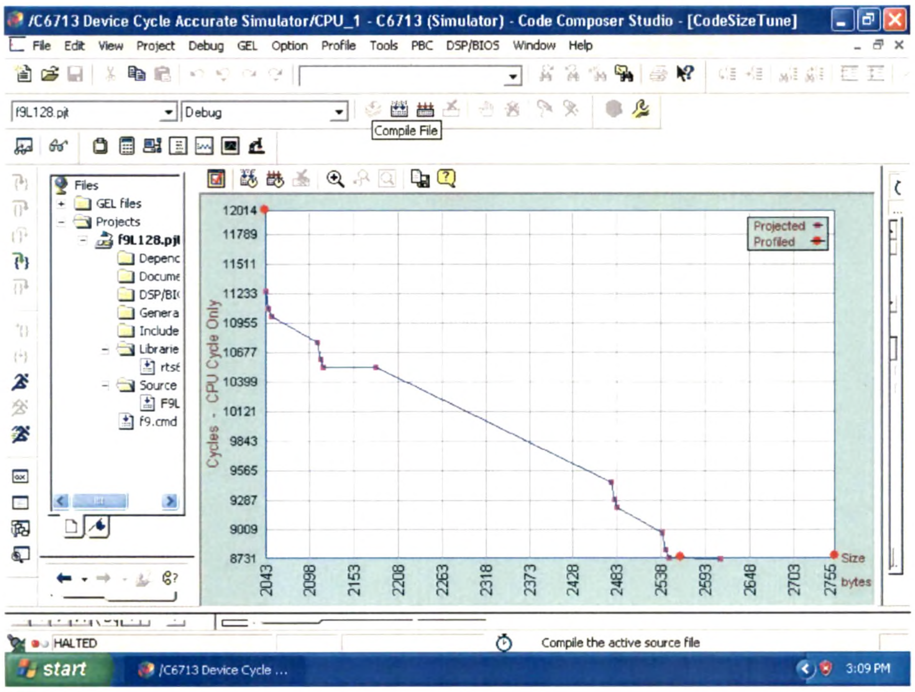
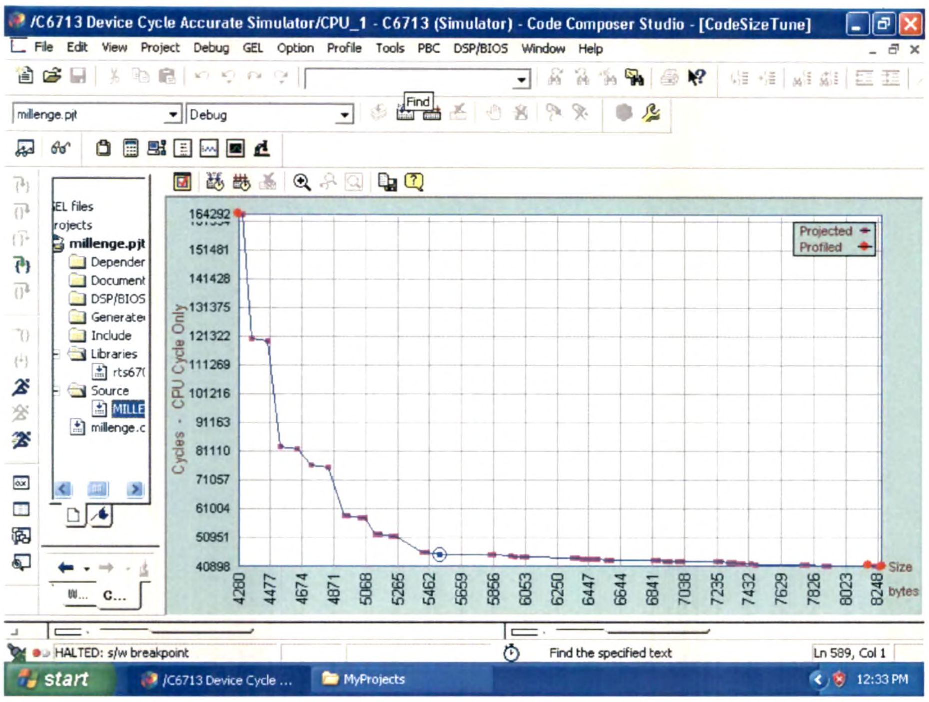


Figure 6.7 : DSP Profile Graph for f9 algorithm



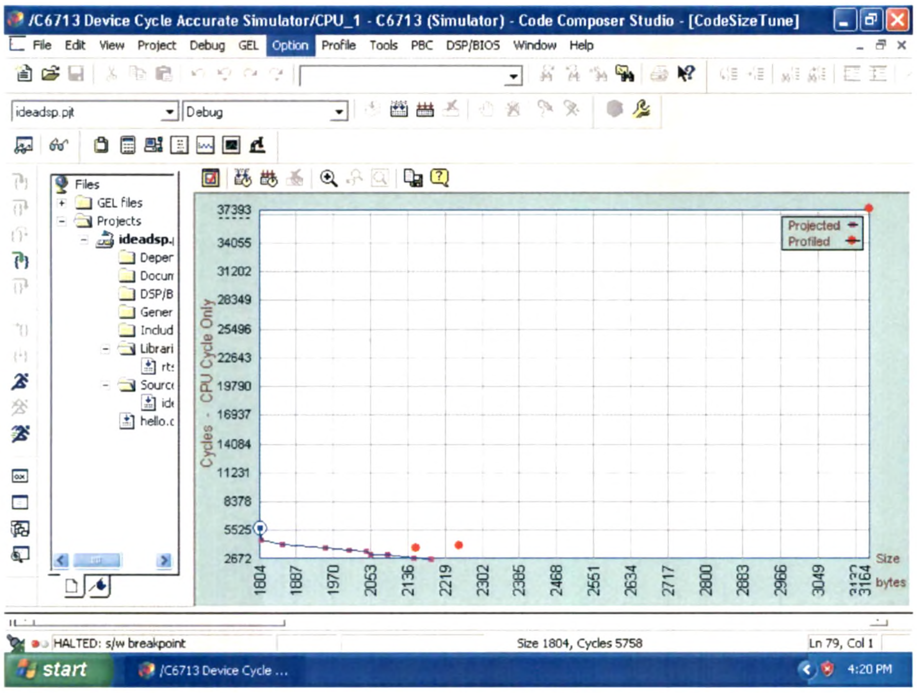
/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio - [CodeSizeTune]

HALTED: s/w breakpoint

Find the specified text

Ln 589, Col 1

Figure 6.8 : DSP Profile Graph for MILENAGE functions



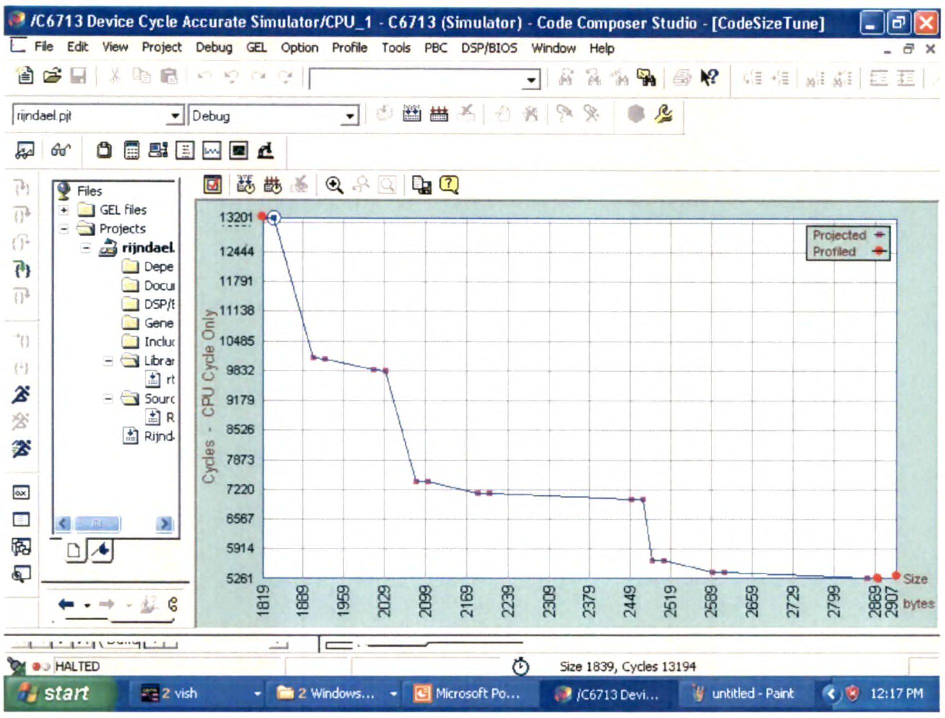
/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio - [CodeSizeTune]

HALTED: s/w breakpoint

Size 1804, Cycles 5758

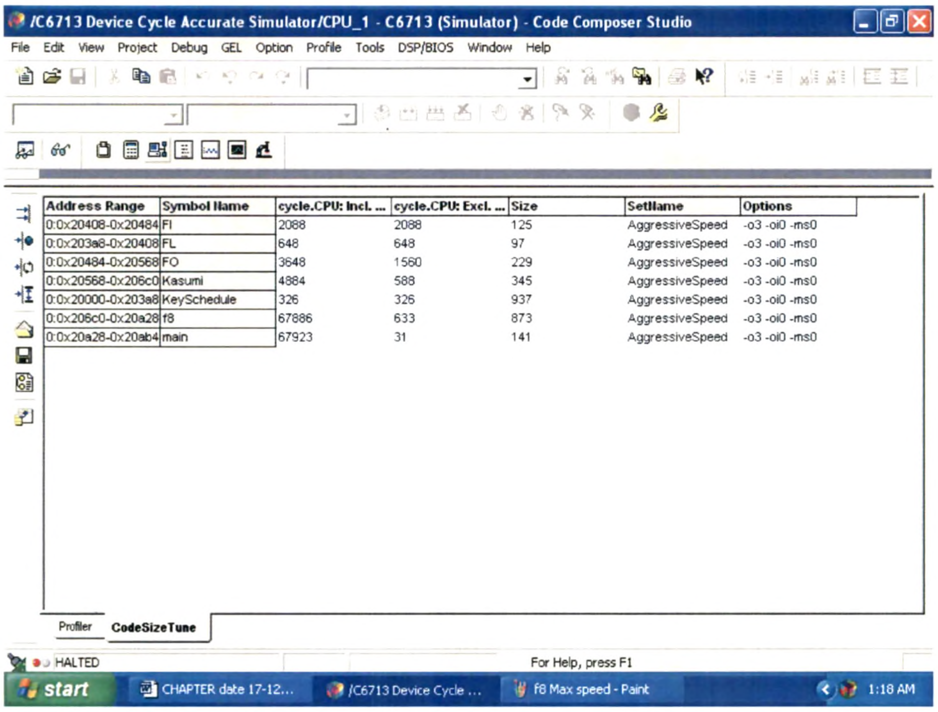
Ln 79, Col 1

Figure 6.9 : DSP Profile Graph for IDEA algorithm



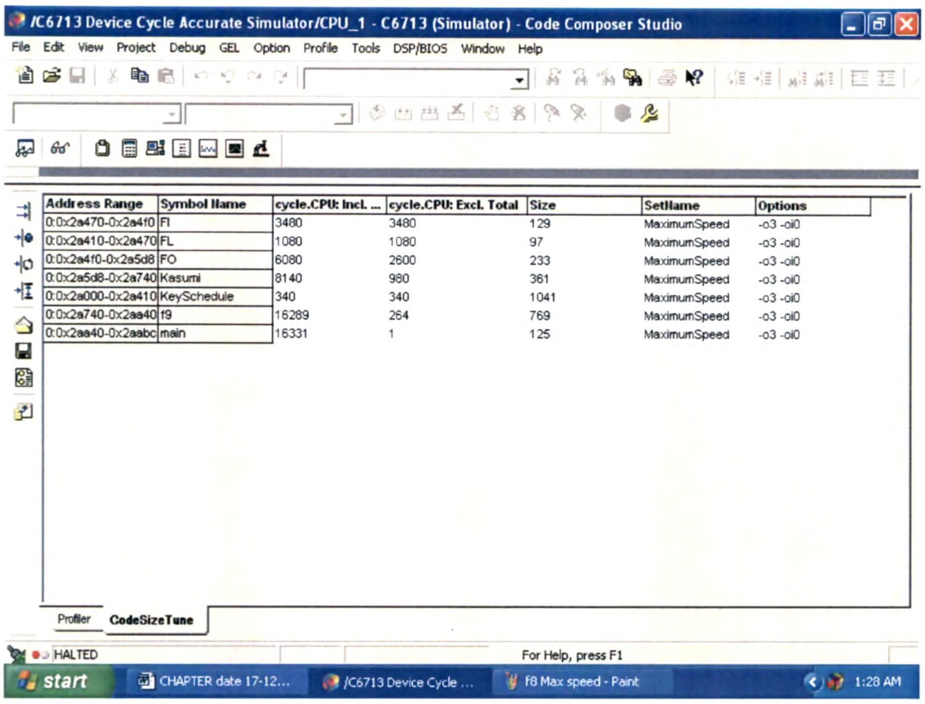
Size 1839, Cycles 13194

Figure 6.10: DSP Profile Graph for Rijndael algorithm


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler
CodeSizeTune
HALTED
For Help, press F1
start
1:18 AM

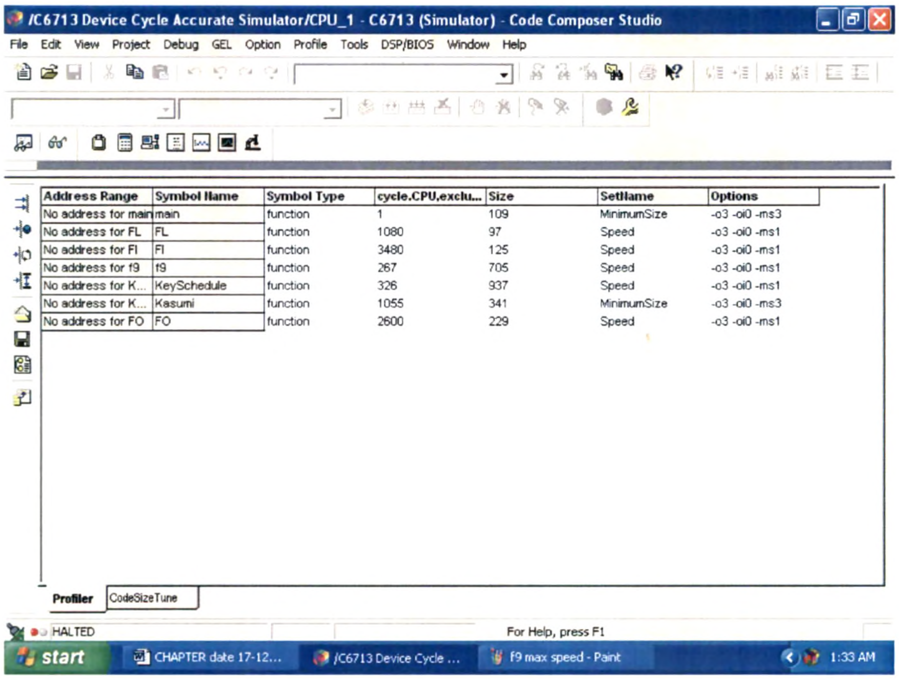
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. ...	Size	SetName	Options
0:0x20408-0x20484	FI	2088	2088	125	AggressiveSpeed	-o3 -oi0 -ms0
0:0x203a8-0x20408	FL	648	648	97	AggressiveSpeed	-o3 -oi0 -ms0
0:0x20484-0x20568	FO	3648	1560	229	AggressiveSpeed	-o3 -oi0 -ms0
0:0x20568-0x206c0	Kasumi	4884	588	345	AggressiveSpeed	-o3 -oi0 -ms0
0:0x20000-0x203a8	KeySchedule	326	326	937	AggressiveSpeed	-o3 -oi0 -ms0
0:0x206c0-0x20a28	f8	67886	633	873	AggressiveSpeed	-o3 -oi0 -ms0
0:0x20a28-0x20ab4	main	67923	31	141	AggressiveSpeed	-o3 -oi0 -ms0

Figure 6.11 : DSP Profile viewer for f8 algorithm (max. speed)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler CodeSizeTune
HALTED
For Help, press F1
start CHAPTER date 17-12... /C6713 Device Cycle ... f8 Max speed - Paint 1:28 AM

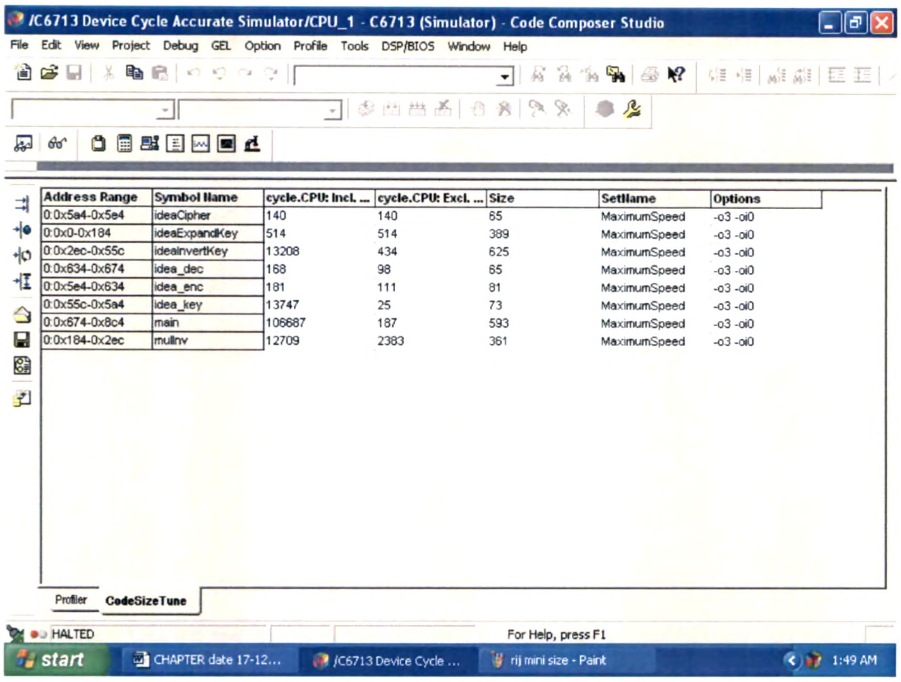
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. Total	Size	SetName	Options
0:0x2a470-0x2a4f0	FI	3480	3480	129	MaximumSpeed	-o3 -oi0
0:0x2a410-0x2a470	FL	1080	1080	97	MaximumSpeed	-o3 -oi0
0:0x2a4f0-0x2a5d8	FO	6080	2600	233	MaximumSpeed	-o3 -oi0
0:0x2a5d8-0x2a740	Kasumi	8140	980	361	MaximumSpeed	-o3 -oi0
0:0x2a000-0x2a410	KeySchedule	340	340	1041	MaximumSpeed	-o3 -oi0
0:0x2a740-0x2aa40	f9	16289	264	769	MaximumSpeed	-o3 -oi0
0:0x2aa40-0x2aabc	main	16331	1	125	MaximumSpeed	-o3 -oi0

Figure 6.12 : DSP Profile Viewer for f9 algorithm (max. speed)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler CodeSizeTune
HALTED
For Help, press F1
start CHAPTER date 17-12... /C6713 Device Cycle ... f9 max speed - Paint 1:33 AM

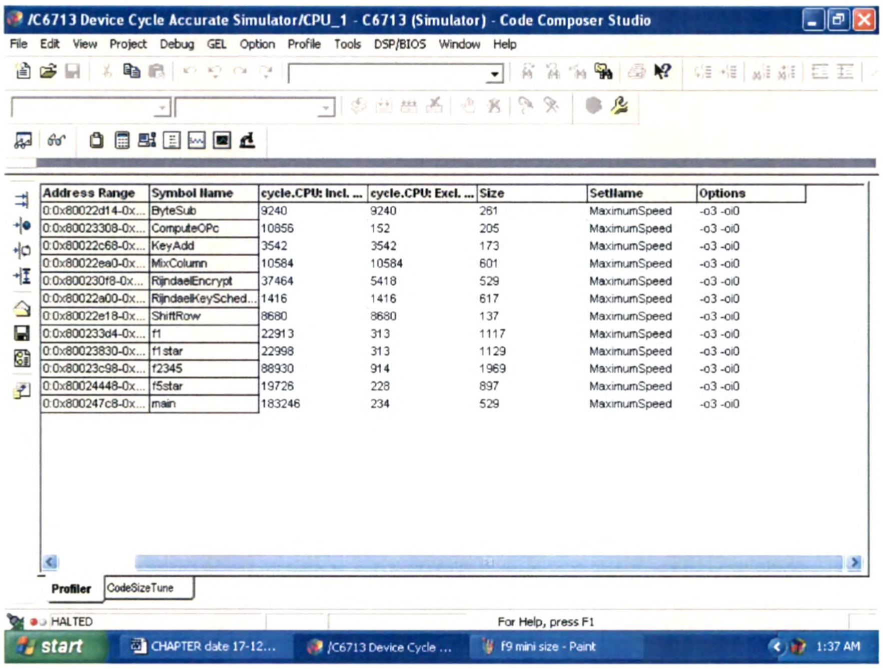
Address Range	Symbol Name	Symbol Type	cycle.CPU,exclu...	Size	SetName	Options
No address for main	main	function	1	109	MinimumSize	-o3 -oi0 -ms3
No address for FL	FL	function	1080	97	Speed	-o3 -oi0 -ms1
No address for FI	FI	function	3480	125	Speed	-o3 -oi0 -ms1
No address for f9	f9	function	267	705	Speed	-o3 -oi0 -ms1
No address for K...	KeySchedule	function	326	937	Speed	-o3 -oi0 -ms1
No address for K...	Kasumi	function	1055	341	MinimumSize	-o3 -oi0 -ms3
No address for FO	FO	function	2600	229	Speed	-o3 -oi0 -ms1

Figure 6.13 :DSP Profile Viewer for f9 algorithm (min. size)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler
CodeSizeTune
HALTED
For Help, press F1
start
CHAPTER date 17-12...
/C6713 Device Cycle ...
rij mini size - Paint
1:49 AM

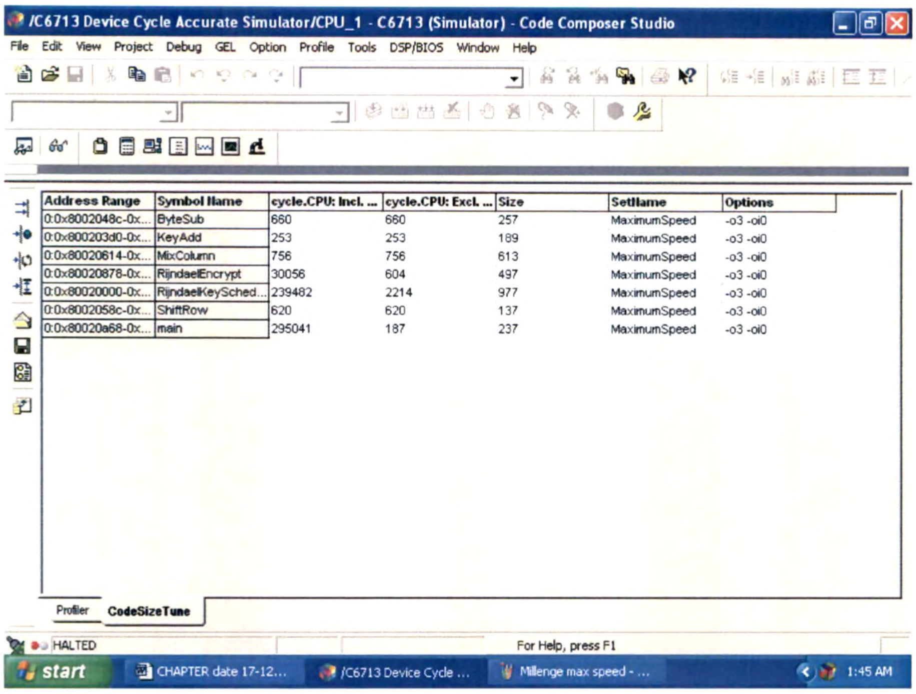
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. ...	Size	SetName	Options
0:0x5a4-0x5e4	ideaCipher	140	140	65	MaximumSpeed	-o3 -oi0
0:0x0-0x184	ideaExpandKey	514	514	389	MaximumSpeed	-o3 -oi0
0:0x2ec-0x55c	ideaInvertKey	13208	434	625	MaximumSpeed	-o3 -oi0
0:0x634-0x674	idea_dec	168	98	65	MaximumSpeed	-o3 -oi0
0:0x5e4-0x634	idea_enc	181	111	81	MaximumSpeed	-o3 -oi0
0:0x55c-0x5a4	idea_key	13747	25	73	MaximumSpeed	-o3 -oi0
0:0x674-0x8c4	main	106687	187	593	MaximumSpeed	-o3 -oi0
0:0x184-0x2ec	mulInv	12709	2383	361	MaximumSpeed	-o3 -oi0

Figure 6.14 : DSP Profile Viewer for IDEA algorithm (max. speed)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler
CodeSizeTune
HALTED
For Help, press F1
start
CHAPTER date 17-12...
/C6713 Device Cycle ...
f9 mini size - Paint
1:37 AM

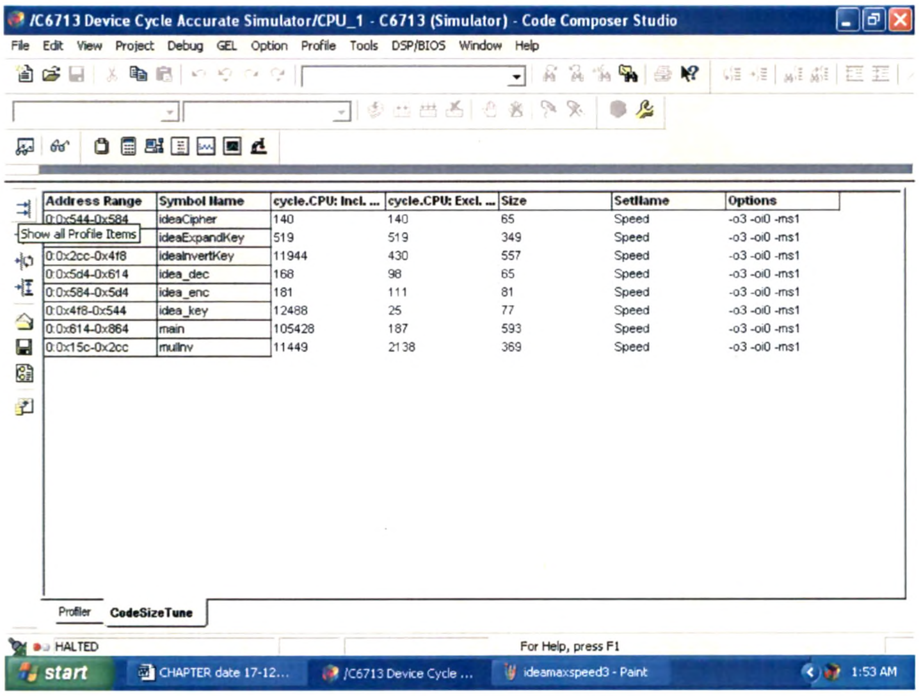
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. ...	Size	SetName	Options
0:0x80022d14-0x...	ByteSub	9240	9240	261	MaximumSpeed	-o3 -oi0
0:0x80023308-0x...	ComputeOPc	10856	152	205	MaximumSpeed	-o3 -oi0
0:0x80022c68-0x...	KeyAdd	3542	3542	173	MaximumSpeed	-o3 -oi0
0:0x80022ea0-0x...	MixColumn	10584	10584	601	MaximumSpeed	-o3 -oi0
0:0x800230f8-0x...	RijndaelEncrypt	37464	5418	529	MaximumSpeed	-o3 -oi0
0:0x80022a00-0x...	RijndaelKeySched...	1416	1416	617	MaximumSpeed	-o3 -oi0
0:0x80022e18-0x...	ShiftRow	8680	8680	137	MaximumSpeed	-o3 -oi0
0:0x800233d4-0x...	f1	22913	313	1117	MaximumSpeed	-o3 -oi0
0:0x80023830-0x...	f1star	22998	313	1129	MaximumSpeed	-o3 -oi0
0:0x80023c98-0x...	f2345	88930	914	1969	MaximumSpeed	-o3 -oi0
0:0x80024448-0x...	f5star	19726	228	897	MaximumSpeed	-o3 -oi0
0:0x800247c8-0x...	main	183246	234	529	MaximumSpeed	-o3 -oi0

Figure 6.15 : DSP Profile Viewer for MILENAGE algorithm (max. speed)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler CodeSizeTune
HALTED
For Help, press F1
start
CHAPTER date 17-12...
/C6713 Device Cycle ...
Milenge max speed - ...
1:45 AM

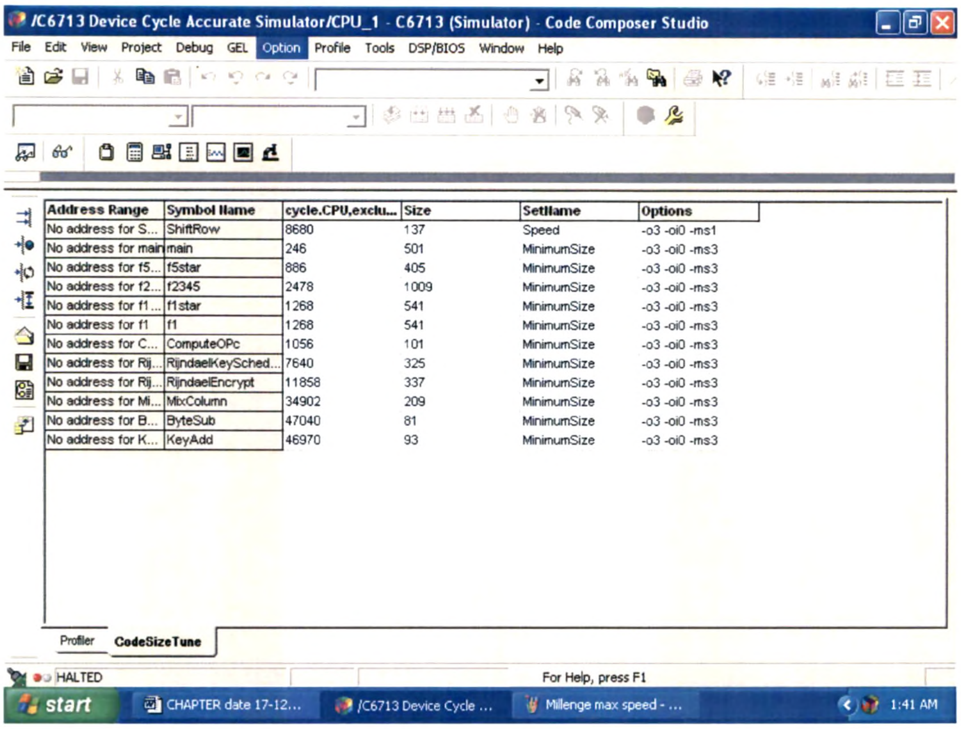
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. ...	Size	SetName	Options
0:0x8002048c-0x...	ByteSub	660	660	257	MaximumSpeed	-o3 -oi0
0:0x800203d0-0x...	KeyAdd	253	253	189	MaximumSpeed	-o3 -oi0
0:0x80020614-0x...	MixColumn	756	756	613	MaximumSpeed	-o3 -oi0
0:0x80020878-0x...	RijndaelEncrypt	30056	604	497	MaximumSpeed	-o3 -oi0
0:0x80020000-0x...	RijndaelKeySched...	239482	2214	977	MaximumSpeed	-o3 -oi0
0:0x8002058c-0x...	ShiftRow	620	620	137	MaximumSpeed	-o3 -oi0
0:0x80020a68-0x...	main	295041	187	237	MaximumSpeed	-o3 -oi0

Figure 6.16 : DSP Profile Viewer for Rijndael algorithm (max. speed)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Show all Profile Items
Profiler CodeSizeTune
HALTED
For Help, press F1
start
CHAPTER date 17-12...
/C6713 Device Cycle ...
ideamaxspeed3 - Paint
1:53 AM

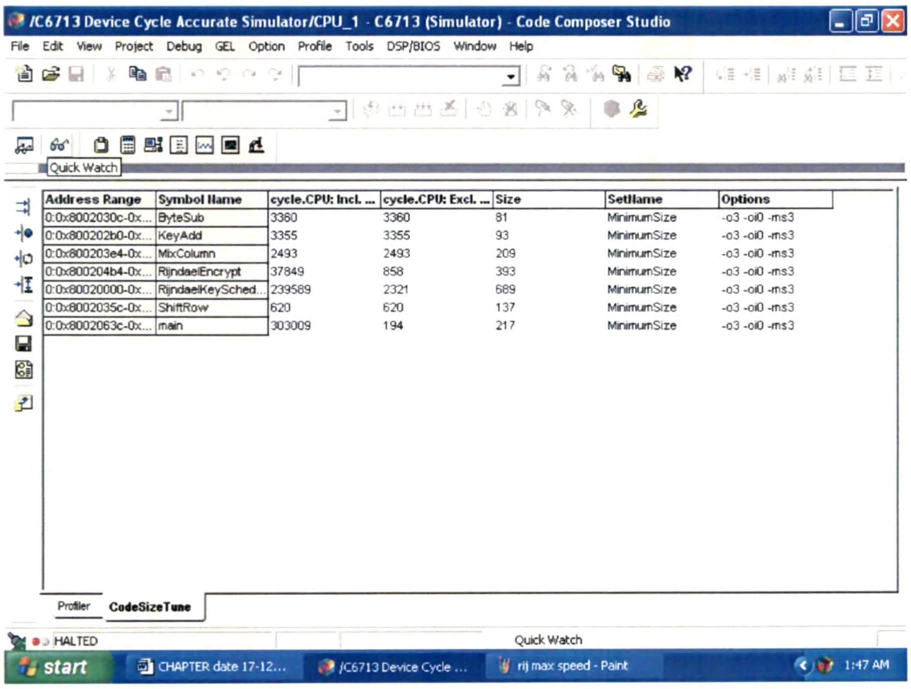
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. ...	Size	SetName	Options
0:0x544-0x584	ideaCipher	140	140	65	Speed	-o3 -oi0 -ms1
[illegible]	ideaExpandKey	519	519	349	Speed	-o3 -oi0 -ms1
0:0x2cc-0x4f8	ideaInvertKey	11944	430	557	Speed	-o3 -oi0 -ms1
0:0x5d4-0x614	idea_dec	168	98	65	Speed	-o3 -oi0 -ms1
0:0x584-0x5d4	idea_enc	181	111	81	Speed	-o3 -oi0 -ms1
0:0x4f8-0x544	idea_key	12488	25	77	Speed	-o3 -oi0 -ms1
0:0x614-0x864	main	105428	187	593	Speed	-o3 -oi0 -ms1
0:0x15c-0x2cc	mulInv	11449	2138	369	Speed	-o3 -oi0 -ms1

Figure 6.17: DSP Profile viewer for IDEA Algorithm (min. size)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Profiler
CodeSizeTune
HALTED
For Help, press F1
start
CHAPTER date 17-12...
/C6713 Device Cycle ...
Milenge max speed - ...
1:41 AM

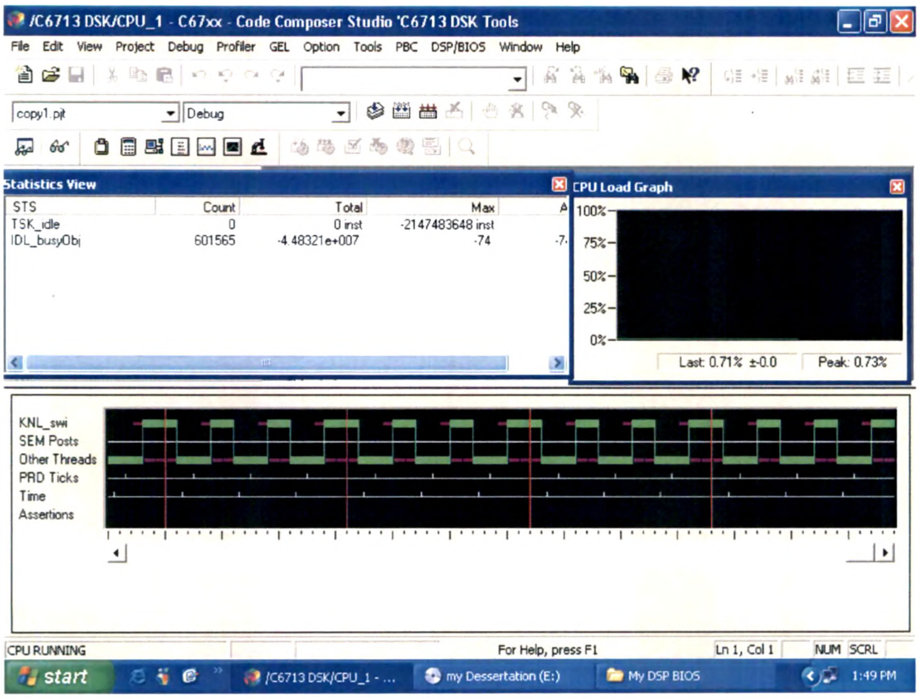
Address Range	Symbol Name	cycle.CPU,exclu...	Size	SetName	Options
No address for S...	ShiftRow	8680	137	Speed	-o3 -oi0 -ms1
No address for main	main	246	501	MinimumSize	-o3 -oi0 -ms3
No address for f5...	f5star	886	405	MinimumSize	-o3 -oi0 -ms3
No address for f2...	f2345	2478	1009	MinimumSize	-o3 -oi0 -ms3
No address for f1 ...	f1star	1268	541	MinimumSize	-o3 -oi0 -ms3
No address for f1	f1	1268	541	MinimumSize	-o3 -oi0 -ms3
No address for C...	ComputeOPc	1056	101	MinimumSize	-o3 -oi0 -ms3
No address for Rij...	RijndaelKeySched...	7640	325	MinimumSize	-o3 -oi0 -ms3
No address for Rij...	RijndaelEncrypt	11858	337	MinimumSize	-o3 -oi0 -ms3
No address for Mi...	MixColumn	34902	209	MinimumSize	-o3 -oi0 -ms3
No address for B...	ByteSub	47040	81	MinimumSize	-o3 -oi0 -ms3
No address for K...	KeyAdd	46970	93	MinimumSize	-o3 -oi0 -ms3

Figure 6.18 : DSP Profile Viewer for MILENAGE (min. size)


/C6713 Device Cycle Accurate Simulator/CPU_1 - C6713 (Simulator) - Code Composer Studio
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
Quick Watch
Profiler
CodeSizeTune
HALTED
Quick Watch
start
CHAPTER date 17-12...
/C6713 Device Cycle ...
rij max speed - Paint
1:47 AM

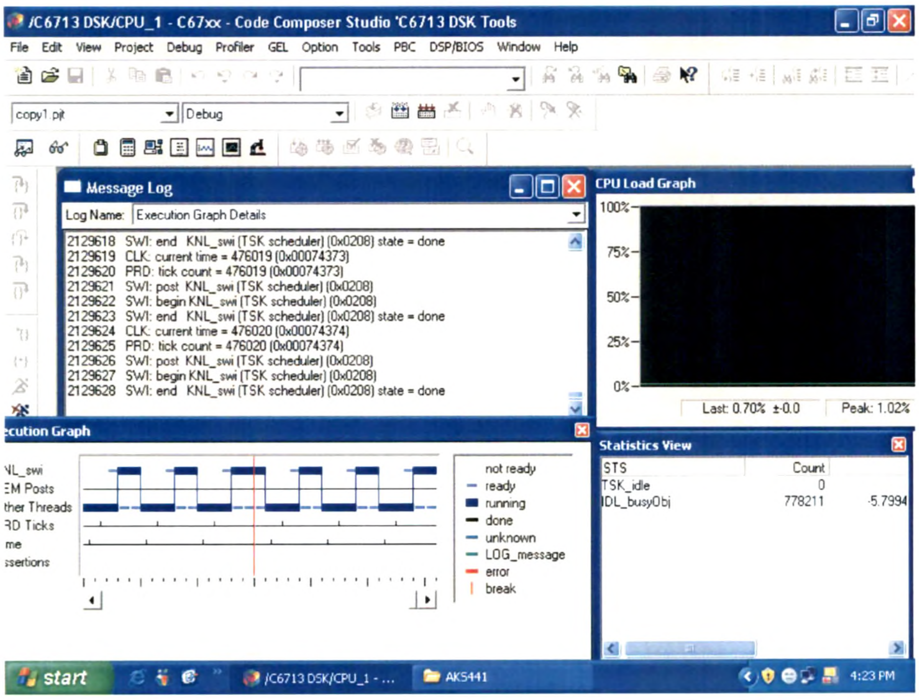
Address Range	Symbol Name	cycle.CPU: Incl. ...	cycle.CPU: Excl. ...	Size	SetName	Options
0:0x8002030c-0x...	ByteSub	3360	3360	81	MinimumSize	-o3 -oi0 -ms3
0:0x800202b0-0x...	KeyAdd	3355	3355	93	MinimumSize	-o3 -oi0 -ms3
0:0x800203e4-0x...	MixColumn	2493	2493	209	MinimumSize	-o3 -oi0 -ms3
0:0x800204b4-0x...	RijndaelEncrypt	37849	858	393	MinimumSize	-o3 -oi0 -ms3
0:0x80020000-0x...	RijndaelKeySched...	239589	2321	689	MinimumSize	-o3 -oi0 -ms3
0:0x8002035c-0x...	ShiftRow	620	620	137	MinimumSize	-o3 -oi0 -ms3
0:0x8002063c-0x...	main	303009	194	217	MinimumSize	-o3 -oi0 -ms3

Figure 6.19 : DSP Profile viewer for AES (Rijndael) (min. size)



STS	Count	Total	Max
TSK_idle	0	0 inst	-2147483648 inst
IDL_busyObj	601565	-4.48321e+007	-74

Figure 6.20 : DSP/BIOS analysis for MILENAGE algorithm

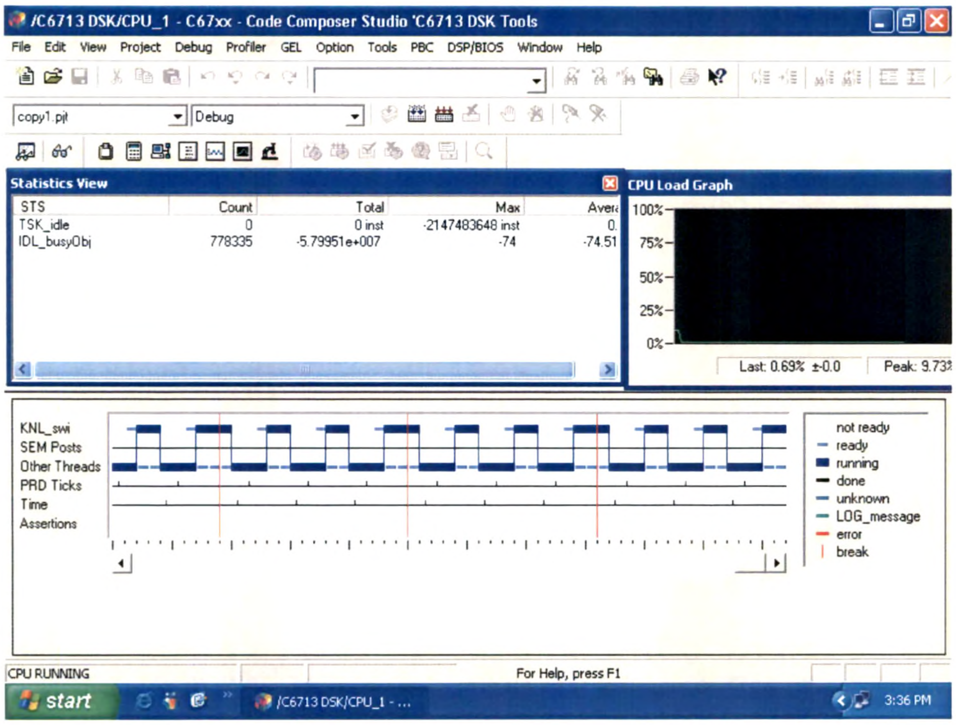


Log Name: Execution Graph Details

```
2129618  SWI: end   KNL_swi (TSK scheduler) (0x0208) state = done
2129619  CLK: current time = 476019 (0x00074373)
2129620  PRD: tick count = 476019 (0x00074373)
2129621  SWI: post  KNL_swi (TSK scheduler) (0x0208)
2129622  SWI: begin KNL_swi (TSK scheduler) (0x0208)
2129623  SWI: end   KNL_swi (TSK scheduler) (0x0208) state = done
2129624  CLK: current time = 476020 (0x00074374)
2129625  PRD: tick count = 476020 (0x00074374)
2129626  SWI: post  KNL_swi (TSK scheduler) (0x0208)
2129627  SWI: begin KNL_swi (TSK scheduler) (0x0208)
2129628  SWI: end   KNL_swi (TSK scheduler) (0x0208) state = done
```

STS	Count	
TSK_idle	0	
IDL_busyObj	778211	-5.7994

Figure 6.21 : DSP/BIOS analysis for f8 algorithm

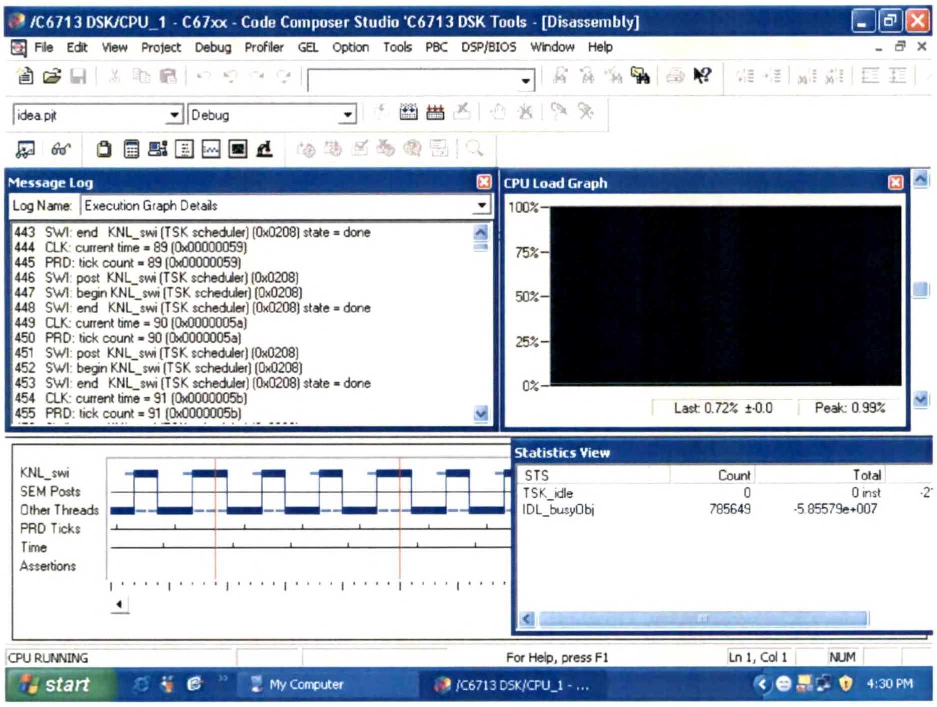


STS	Count	Total	Max	Avera
TSK_idle	0	0 inst	-2147483648 inst	0.
IDL_busyObj	778335	-5.79951e+007	-74	-74.51

Last: 0.69% ±0.0 Peak: 9.73

	not ready
—	ready
■	running
—	done
—	unknown
—	LOG_message
—	error
|	break

Figure 6.22 : DSP/BIOS analysis for f9 algorithm



Log Name: Execution Graph Details

```
443  SWI: end   KNL_swi (TSK scheduler) (0x0208) state = done
444  CLK: current time = 89 (0x00000059)
445  PRD: tick count = 89 (0x00000059)
446  SWI: post  KNL_swi (TSK scheduler) (0x0208)
447  SWI: begin KNL_swi (TSK scheduler) (0x0208)
448  SWI: end   KNL_swi (TSK scheduler) (0x0208) state = done
449  CLK: current time = 90 (0x0000005a)
450  PRD: tick count = 90 (0x0000005a)
451  SWI: post  KNL_swi (TSK scheduler) (0x0208)
452  SWI: begin KNL_swi (TSK scheduler) (0x0208)
453  SWI: end   KNL_swi (TSK scheduler) (0x0208) state = done
454  CLK: current time = 91 (0x0000005b)
455  PRD: tick count = 91 (0x0000005b)
```

Last: 0.72% ±0.0 Peak: 0.99%

STS	Count	Total
TSK_idle	0	0 inst
IDL_busyObj	785649	-5.85579e+007

Figure 6.23 : DSP/BIOS analysis for IDEA

Following table summarizes profile details.

Algorithm	Ref. Fig.	CPU cycles	Code size(Bytes)	Remarks
f8	6.6	8363	2251	Min. size
f8	6.6	5850	2724	Max. speed
f9	6.7	12014	2043	Min. size
f9	6.7	8731	2550	Max. speed
MILENAGE	6.8	164292	4280	Min. size
MILENAGE	6.8	40898	8248	Max. speed
IDEA	6.9	5758	1804	Min. size
IDEA	6.9	2672	2180	Max. speed
Rijndael	6.10	13201	1819	Min. size
Rijndael	6.10	5261	2869	Max. speed

Table 6.12 : Profile summary (CPU cycles v/s code size) for Code Size Tune tool

Following table summarizes DSP/BIOS CPU peak load.

Sr.No	Algorithm	Peak CPU load
1	f8	1.02%
2	f9	9.73%
3	MILENAGE	0.73%
4	IDEA	0.99%

Table 6.13 : DSP/BIOS CPU peak load

6.5 Scholarly comments: Software Implementation - Cryptographic Algorithms:

Our implementation includes analysis and synthesis using MATLAB/SIMULINK , DSP simulator and emulator on TMS 320C6713, Modelsim , Xilinx and altera tools

6.5.1 SIMULINK/MATLAB and CCS with Simulator

The study and analysis of MATLAB based software implementation gives an idea of general purpose processor speed for executing the code.

- The functions are profiled for timing analysis, which is critical for measuring encryption rate of the algorithm. It is found that the core algorithm KASUMI consumes 66% time in f9 execution, whereas it is 56.1% for f8 execution. The difference is clearly indicating slower execution of f9 on general purpose processor.
- Since f8 is used for confidentiality of data, it needs to run faster compared to integrity algorithm f9. Since f9 authenticates the control messages rather than actual information, it can tolerate delay.
- MATLAB is matrix oriented language and is ideally suitable for data structure of Advanced Encryption Standard (AES). The block cipher Rijndael is the 128-bit block version of AES and it was found to be consistently good performer in both hardware and software across a wide range of computing environments. Our implementation of AES in MATLAB proves that it is more complex compared to f8 and f9 in terms of number of functions, their repeated execution and total execution time.
- System level studies for end-to-end communications is done using SIMULINK. The bit error rate (BER) is studied for the system with and without encryption. The embedded target for DSP and RTW enabled us to develop SIMULINK blocks f8 and f9 for carrying out link by link or end to end encryption mechanism. Using SIMULINK support of Xilinx blocksets, the encryption functionality can be directly designed on VLSI platform. This is to be explored further and we see a promising research work based on this modeling .

Performance evaluation of software ciphering in UMTS radio network controller was investigated by Prof. Timo Korhonen [12]. The purpose of study was to find out whether the software implementation of UMTS radio access network encryption is feasible. Feasibility was evaluated primarily from the performance and capacity point of view.

A ciphering software module was implemented for the tests based on reference implementation in 3GPP TS 35.202 [70] . C programs were used and existing hardware-based ciphering was used as a reference. Ciphering mask generation was done in a separate ASIC circuit. Performance was measured in terms of execution time. Average, maximum and minimum execution times were measured. Texas Instruments TMS 320C55x family of DSPs was used and the ciphering ASIC was connected to DSPs via serial interface. Major conclusions drawn were :

1. Software ciphering is significantly faster.
2. Software ciphering consumes about half of the time used by ASIC.
3. Difference behaves linearly throughout the tested range.
4. Performance is almost similar for speech and non-real time data.
5. Average difference in execution time varies from 0.06% to 21.74%

Digital Signal Processors provide faster arithmetic instructions and capability for real time signal processing .The main areas of DSP applications are embedded systems, wireless devices ,DSL modems and various consumer electronic devices.

6.5.2 Implementation and Analysis on Hardware Platform :

For assuring security over wireless channels, encryption features are required by DSPs. Whether high-end DSPs are suitable for implementation of cryptographic algorithms was a matter of investigation and there were very few research papers available in this regard [20] . The research work carried out by Thomas J. Wollinger indicated that Rijndael encryption require 16384 bytes and gives speed of 112.3 Mbps.

⇨ On the same line, we have implemented f8, f9, and IDEA and MILENAGE algorithms on DSP TMS 320C6713. This research work clearly favors DSP as an implementation platform and we have investigated this in much more details using Code Composer Studio and DSK 6713 hardware.

⇨ Both Simulator and Emulator analysis are done and they show consistency in execution speed with optimized codes. Simulator analysis is considered to be more reliable than emulator analysis because of communication between the target and host in latter case.

⇨ Profile based compilation is an important tool available with TMS 320C6000 devices. It is identified as Code Size Tune (CST) in the new version of CCS.CST uses several profile collection option sets to build and profile the application. The best profile collection data sets using various options: maximum speed , minimum size and speed were studied and plotted on a two –dimensional graph of code size vs. performance.

⇨ There is a trade-off between speed and size. This means that the speed can be compromised if the code size has to be minimum. Most of the User Equipments (UE) in 3G mobile communications will have limited memory and hence the speed can be compromised with the condition that it does not fall below 2 Mbps. In RNC, enough memory is available but speed will-be many times higher due to multiplexed data in time domain. Detailed investigations of profiled functions have enabled us to arrive at some important conclusions .

The following conclusions are drawn:

1. Source codes written in C language for encryption algorithms are tuned for optimization of speed/size. The profile viewer displays all collected information during the tuning process, which can be used to pinpoint sections of the code that require the most tuning. Data in the profile viewer can be sorted from selection.

   For example, FI function requires maximum time (3480 CPU cycles) for f8 and f9 algorithms. This function can be coded with utmost care for fastest possible execution.

2. AES gives highest throughput of 346.98 Mbps on DSP platform. The AES algorithm is the Kernel in MILENAGE constructions because of its strength. Because AES is the fastest running algorithm on DSP platform , MILENAGE is also faster and its throughput is 276.92 Mbps. This indicates that the key

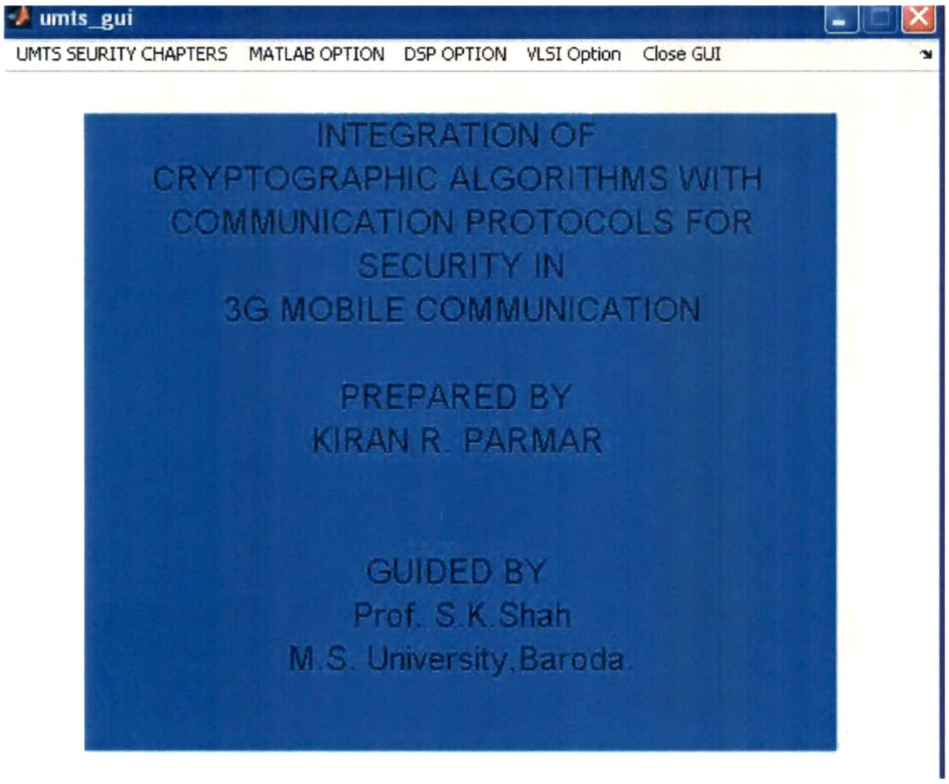
generation process is much faster and Cipher Key (CK) and Integrity Key(IK) are quickly generated for fresh rounds of communication session.

3. The trade-off between speed and size is very important for DSP implementation since it gives us idea about data rate requirement and available data rate for 3G applications.
4. Throughput available for all algorithms namely f8, f9, IDEA, MILENAGE, KASUMI and AES (RIJNDAEL) exceed 2 Mbps requirement of 3G on DSP platform. We strongly recommend DSP for encryption functionality since it is already used for many applications.
5. Profiling of all algorithms on DSP platform may be used to modify the algorithm for speed or memory optimization without seriously affecting their cryptographic strength.
6. The length of message is important in packet data communication. We have selected four different data sizes viz. 32, 128, 1024 and 5120 bits. The relation between no. of bits encrypted and time required to encrypt is exactly linear.
7. CPU peak load indicates that other programs can run concurrently while these encryption algorithms run on DSP. f9 requires highest CPU load indicating that the authentication process requires more time to transfer data from target to the host . The host is part of RNC and UE in this case and this is permissible because authentication takes place prior to actual communication.

6.6 GUI development for 3G security algorithms' implementation: MATLAB has an integrated environment for developing Graphical User Interface called Graphical User Integrated Development (GUIDE), which supports number of graphics functions and features to demonstrate various functionalities of the software. It helps in design, laying out, setting properties for GUI components, programming and running the GUI. CCS link within MATLAB provides interface between MATLAB and CCS so as to run and test MATLAB program directly and interactively in real time on the embedded targets such as TMS 320C6713. The GUI enables user to select algorithm, implementation method and purpose/his interest using normal windows commands and view the desired results on the screen. The software and hardware support tools required for demonstration of software includes:

1. MATLAB version 7.0.1.24704 (R14) Service Pack 1 or higher
2. MATLAB link for embedded Targets- ccslink for TI67xx
3. Real Time Workshop (RTW)
4. C6713 DSK CCS 3.0 or higher(Simulator version)
5. C6713 DSK CCS3.0 or higher(Emulator version)
6. DSK 6713 kit based on Texas DSP TMS 320C6713.

The projects containing source files for various algorithms are built, loaded and executed on DSP target CPU. It is possible to open the projects by invoking GUI menu and work with options available within CCS environment. On command prompt in the command window enter the command **>> umts gui** .The screen shown in figure 6.24 is displayed.



INTEGRATION OF CRYPTOGRAPHIC ALGORITHMS WITH COMMUNICATION PROTOCOLS FOR SECURITY IN 3G MOBILE COMMUNICATION

PREPARED BY
KIRAN R. PARMAR

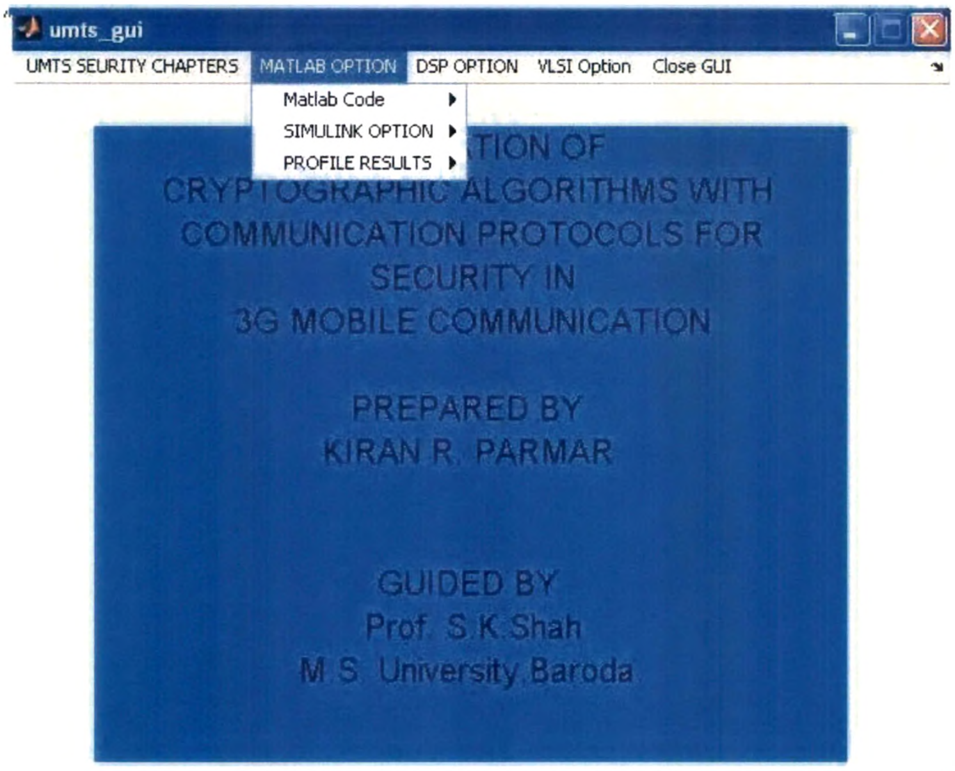
GUIDED BY
Prof. S.K.Shah
M.S. University,Baroda.

Fig.6.24: Graphical User Interface: MAIN- Menu Screen

As shown in the fig 6.24 the MAIN menu screen provides...

- chapters: to help user go through the theoretical background and information regarding algorithm
- MATLAB option: Selection of algorithm → f8, f9 and KASUMI
  - Profile Option: Results of selected algorithm
- DSP option: Selection of Hardware platform or profile plot of selected algorithm. (f8, f9, Rijndael, MILENAGE and IDEA).
- VLSI option: Selection of Simulation graphs of f8,f9 and KASUMI

User can select implementation option MATLAB , DSP or VLSI and accordingly figure 6.25(a), 6.25(b) or 6.25(c) will appear. Click the mouse to run the simulation/emulation program and view results.

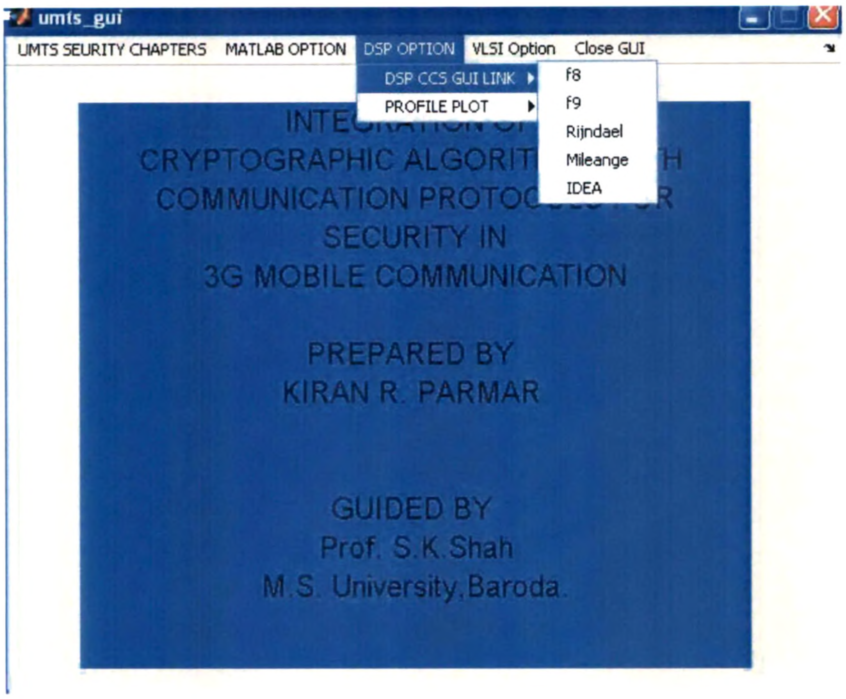


...TION OF CRYPTOGRAPHIC ALGORITHMS WITH COMMUNICATION PROTOCOLS FOR SECURITY IN 3G MOBILE COMMUNICATION

PREPARED BY
KIRAN R. PARMAR

GUIDED BY
Prof. S.K.Shah
M.S. University Baroda

Fig.6.25(a) : Graphical User Interface: MATLAB Option

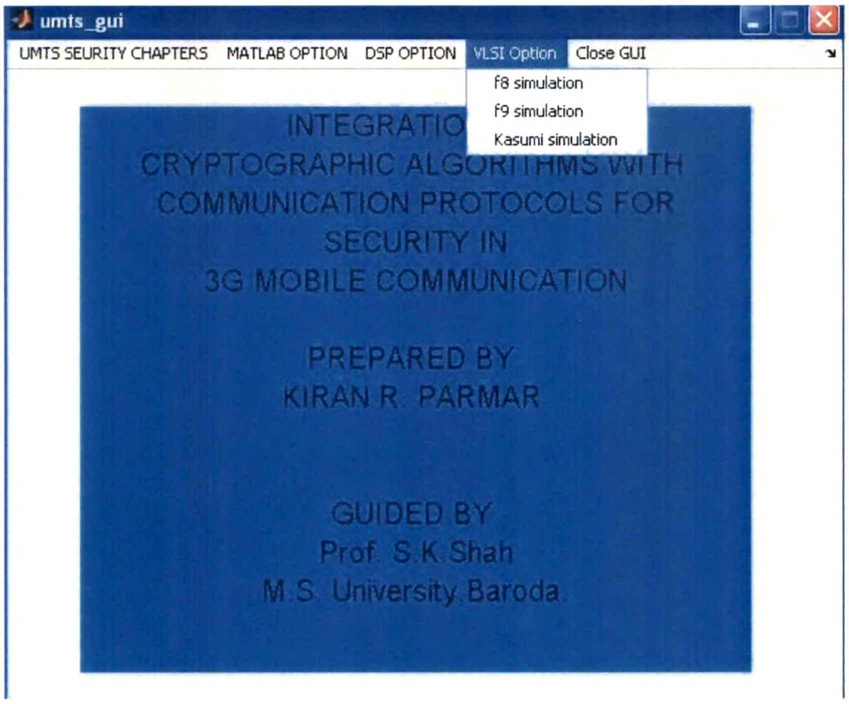


INTE
CRYPTOGRAPHIC ALGORIT
COMMUNICATION PROTOC
SECURITY IN
3G MOBILE COMMUNICATION

PREPARED BY
KIRAN R. PARMAR

GUIDED BY
Prof. S.K.Shah
M.S. University,Baroda.

Fig 6.25(b): Graphical User Interface: DSP Option



INTEGRATIO
CRYPTOGRAPHIC ALGORITHMS WITH
COMMUNICATION PROTOCOLS FOR
SECURITY IN
3G MOBILE COMMUNICATION

PREPARED BY
KIRAN R. PARMAR

GUIDED BY
Prof. S.K.Shah
M.S. University,Baroda.

Fig 6.25(c) : Graphical User Interface: VLSI Option

Operating Procedure : The steps required to use the GUI software are as follows:

- Run the MATLAB program **umts gui.**
- Select the menu
  1. **"UMTS SECURITY CHAPTERS"** - study thesis chapters.
  2. **"MATLAB OPTION"** to view source codes on f8, f9 and Kasumi.
  3. "**DSP option**" to run CCS projects .
     CCS link has to be established and the project will be loaded into memory. The project can run from debug menu within CCS The profiling and analyzing the code for optimization is possible.
  4. **"VLSI Option"** to view Simulation waveforms for f8, f9 or KASUMI
  5. **"CLOSE GUI"** to exit from GUI.

6.7 Summary : In this chapter, we have discussed software implementation of AES (Rijndael), IDEA, KASUMI, f8 , f9 and MILENAGE algorithms. Profiling of the algorithms is also done in MATLAB and DSP. Simulator and Emulator analysis as well as DSP-BIOS based real time analysis is discussed in order to have in-depth study of implementation aspects on software platform. Scholarly remarks on software implementation are mentioned at the end of the chapter. The GUI developed is supplementary to the research work undertaken on various platforms. It can serve as an academic product to teach basics of UMTS security, Details of implementations of cryptographic algorithms on MATLAB, SIMULINK and DSP environments and analysis of the same using profile graphs.