Speed optimization of Cryptographic Algorithm Using Hardware-Software Co-design

Indrayani I. Patle (M.Tech. 4th sem) , Prof. Jitendra Zalke (Assistant Professor)

1(M.Tech. 4th sem) Department of Electronics Engineering, RCOEM, Nagpur, India ; 2(Assistant Professor) Department Electronic Design Technology, RCOEM, Nagpur, India
E-mail : indpatle@gmail.com

ABSTRACT

Cryptography is the science of information security which converts data into a secret code for transmission over a public network. In cryptography the original text called "plaintext" is turned into a coded equivalent called "ciphertext" via an encryption algorithm. The ciphertext is decrypted at the receiving end and turned back into plaintext.

This encryption/decryption algorithms are commonly implemented using an embedded processor combined with specific hardware. There are a several embedded soft core processors like NIOS3, LEON3, MicroBlaze etc that are commonly used to implement whole encryption/decryption algorithm.

Implementation of whole cryptographic algorithm in C programming language results in slower the speed. While FPGAs with soft processor cores allows designer to make proper choice based on his own design requirements to build his own hardware platform.

So this Paper presents the work on speed optimization of cryptographic algorithm using hardware-software co-design. Its main purpose is to increase the computational speed of the SMS4 algorithm. Custom hardware of time consuming block is interface with microblaze soft core processor. Hardware of time consuming block and software function of remaining algorithm increases computational speed of cryptographic algorithm.

Keywords : VHDL, FPGA, MICROBLAZE, SCP

1 INTRODUCTION

Data exchanges in communication systems demand more and more security. Consequently, it is typical to embed a specific cryptographic subsystem in all kinds of network equipment.

This encryption subsystem is commonly implemented using an embedded processor [1] combined with specific hardware.

Field programmable gate arrays (FPGAs) provide designers with the ability to quickly create hardware of circuits. Increases in FPGA configurable logic capacity and decreasing FPGA costs have enabled designers to more readily incorporate FPGAs in their designs. While FPGAs with soft processor cores provide designers with increased flexibility.

Reconfigurable logic devices, such as field programmable gate arrays (FPGAs), have been very effective to implement dedicated encryption architectures [2]. Over the last few years, the huge increase in FPGA features made possible the implementation of a whole system in a single device: processor, peripherals, memories and so on. Nowadays, it is feasible to implement on an FPGA an entire cryptographic system based on a soft-core processor (SCP), which also includes ciphering cores for hardware acceleration [3, 4].

There are a several soft core processors that are commonly used in SOC applications like PowerPC [5], NIOS3 [6], MicroBlaze [7], and free or open cores that may be used without the need to acquire a license, like LEON3 [8]. The main advantage of these processors are that they are usually well tested and optimized for a specific target hardware and provide a complete set of CAD tools to make the SOC design an easier process. For example, MicroBlaze from Xilinx is well integrated with the development platform from the same foundry, which leads to highly optimized designs at the cost of being bound to a particular technology (Xilinx Spartan and Virtex FPGA families[9] ) and a concrete set of tools (Xilinx ISE and EDK [10]).

1.1 Microblaze Soft-Core Processor

MicroBlaze soft core is highly simplified embedded processor soft core with relatively high performance developed by XILINX Company. This soft core enjoys high configuration and allows designer to make proper choice based on his own design requirements to build his own hardware platform. The processor architecture includes thirty-two 32-bit general-purpose registers and an orthogonal instruction set. It features a three-stage instruction pipeline, with delayed branch capability for improved instruction throughput. As it is a SCP, the functional units incorporated into the processor architecture can be customized in order to fit as much as possible the target application. This soft core adopts RISC instruction set and Harvard architecture and has the following performance char-
acteristics:

1) 32-bit general-purpose registers and 2 special registers
2) 32-bit instruction word length, 3 operands and 2 kinds of addressing modes.
3) Separated 32-bit instruction and data bus.
4) Complying with IBM OPB specification;
5) Local Memory Bus (LMB) enables direct access to on-chip block memory (BRAM), it provides high-speed instructions and data caching and features three-stage pipeline architecture;
6) Hardware debugging module (MDM) and eight in put/output fast link interfaces (FSL) are available.

Figure 1 shows MicroBlaze's internal structure.

1.2 Cryptography

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography enables you to store sensitive information or transmit it across insecure networks (like the Internet) so that it cannot be read by anyone except the intended recipient. While cryptography is the science of securing data, cryptanalysis is the science of analyzing and breaking secure communication.

1.3 Types of Cryptographic Algorithm

There are several ways of classifying cryptographic algorithms. They can be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms that will be discussed are (Figure 2):

a) **Secret Key Cryptography (SKC):** Uses a single key for both encryption and decryption

b) **Public Key Cryptography (PKC):** Uses one key for encryption and another for decryption

Secret key cryptography algorithms are:

**Data Encryption Standard (DES):** The most common SKC scheme used today, DES was designed by IBM in the 1970s and adopted by the National Bureau of Standards (NBS) [now the National Institute for Standards and Technology (NIST)] in 1977 for commercial and unclassified government applications.

**Advanced Encryption Standard (AES):** In 1997, NIST initiated a very public, 4-1/2 year process to develop a new secure cryptosystem for U.S. government applications.

**SMS4 algorithm** is used in the Chinese National Standard for Wireless LAN WAPI.

1.4 SMS4 algorithm

The Chinese national standard for Wireless Local Area Networks (WLANs), WLAN Authentication and Privacy Infrastructure (WAPI), has been the subject of extensive international debate, especially between China and USA, since over the last four years it has been a rival for IEEE802.11i for adoption as an ISO (International Organization for Standardization) international standard.[11,20] WAPI and IEEE 802.11i have both been proposed as security amendments to the ISO/IEC.
802-11WLAN standard. The two schemes use two different block ciphers for encryption of data: IEEE 802.11i uses the AES cipher, while WAPI uses the SMS4 cipher. Many international corporations, such as SONY, support WAPI in relevant products.[21] SMS4 is a block cipher that produces a 128 bit output from a 128 bit input under the control of a 128 bit key. Currently, SMS4 algorithm can be realized under various environments, Gao Xianwei [12,13] adopts FPGA to realize SMS4 algorithm, the maximum operating clock is 139MHz and the corresponding data throughput is about 539 Mbit/s. Weiwei Yan implement the SMS4 based on SMIC 0.13 um CMOS technology can achieve 200MHz working frequency at very low cost of 22k gates.[15,16]

1.5 SMS4 Encryption Algorithm for Wireless Networks

SMS4 is the first commercial cipher algorithm published in China.[17,18] Both its block length and key length are 128 bit; both encryption algorithm and key expansion algorithm feature 32-round nonlinear iteration structure; S box used in nonlinear transformation deals with 8bit input and 8bit output replacement.[19]

Terminology:

a) Zi and ZiJie (Word and Byte):

\( Z_2^3 \) is the set of e-bit vectors. Specifically, the elements of \( Z_2^3 \) are called \( Zi \) (32 bit words), and the elements of \( Z_2^8 \) are called \( ZiJie \) (8-bit characters, or bytes).

b) S box: The S (substitution) box takes in 8 bits and outputs 8 bits. It is written Sbox (·).

All Sbox numbers are in hexadecimal notation.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d6</td>
<td>90</td>
<td>e9</td>
<td>fe</td>
<td>cc</td>
<td>e1</td>
<td>3d</td>
<td>b7</td>
<td>16</td>
<td>b6</td>
<td>14</td>
<td>c2</td>
<td>28</td>
<td>fb</td>
<td>2c</td>
</tr>
<tr>
<td>1</td>
<td>2b</td>
<td>57</td>
<td>9a</td>
<td>76</td>
<td>2a</td>
<td>be</td>
<td>04</td>
<td>c3</td>
<td>aa</td>
<td>44</td>
<td>13</td>
<td>26</td>
<td>49</td>
<td>86</td>
<td>06</td>
</tr>
<tr>
<td>2</td>
<td>9c</td>
<td>42</td>
<td>50</td>
<td>f4</td>
<td>91</td>
<td>ef</td>
<td>98</td>
<td>7a</td>
<td>53</td>
<td>54</td>
<td>0b</td>
<td>43</td>
<td>ed</td>
<td>cf</td>
<td>ac</td>
</tr>
<tr>
<td>3</td>
<td>e4</td>
<td>b3</td>
<td>1c</td>
<td>a5</td>
<td>c9</td>
<td>88</td>
<td>a8</td>
<td>95</td>
<td>80</td>
<td>df</td>
<td>94</td>
<td>fa</td>
<td>75</td>
<td>8f</td>
<td>3f</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>07</td>
<td>a7</td>
<td>fc</td>
<td>13</td>
<td>73</td>
<td>17</td>
<td>ba</td>
<td>84</td>
<td>59</td>
<td>3c</td>
<td>19</td>
<td>e6</td>
<td>85</td>
<td>4f</td>
</tr>
<tr>
<td>5</td>
<td>68</td>
<td>5b</td>
<td>81</td>
<td>b2</td>
<td>71</td>
<td>84</td>
<td>da</td>
<td>9b</td>
<td>f8</td>
<td>cb</td>
<td>6f</td>
<td>4b</td>
<td>70</td>
<td>56</td>
<td>9d</td>
</tr>
<tr>
<td>6</td>
<td>1e</td>
<td>24</td>
<td>0e</td>
<td>5e</td>
<td>63</td>
<td>58</td>
<td>d1</td>
<td>a2</td>
<td>2b</td>
<td>22</td>
<td>7c</td>
<td>3b</td>
<td>01</td>
<td>21</td>
<td>78</td>
</tr>
<tr>
<td>7</td>
<td>5d</td>
<td>0d</td>
<td>46</td>
<td>57</td>
<td>9f</td>
<td>d3</td>
<td>27</td>
<td>52</td>
<td>4c</td>
<td>36</td>
<td>62</td>
<td>e7</td>
<td>a0</td>
<td>c6</td>
<td>c8</td>
</tr>
<tr>
<td>8</td>
<td>ea</td>
<td>bf</td>
<td>8a</td>
<td>d2</td>
<td>40</td>
<td>c7</td>
<td>38</td>
<td>b5</td>
<td>a3</td>
<td>f7</td>
<td>f2</td>
<td>ce</td>
<td>f9</td>
<td>61</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>e0</td>
<td>ae</td>
<td>5d</td>
<td>a6</td>
<td>9b</td>
<td>34</td>
<td>1a</td>
<td>55</td>
<td>ad</td>
<td>08</td>
<td>32</td>
<td>30</td>
<td>55</td>
<td>58</td>
<td>e1</td>
</tr>
<tr>
<td>a</td>
<td>1d</td>
<td>f5</td>
<td>e2</td>
<td>8e</td>
<td>62</td>
<td>66</td>
<td>ca</td>
<td>60</td>
<td>c9</td>
<td>29</td>
<td>23</td>
<td>ab</td>
<td>0d</td>
<td>53</td>
<td>4e</td>
</tr>
<tr>
<td>b</td>
<td>d5</td>
<td>2b</td>
<td>37</td>
<td>45</td>
<td>df</td>
<td>8e</td>
<td>2f</td>
<td>03</td>
<td>ff</td>
<td>6a</td>
<td>72</td>
<td>6d</td>
<td>6c</td>
<td>5b</td>
<td>51</td>
</tr>
<tr>
<td>c</td>
<td>bd</td>
<td>1b</td>
<td>af</td>
<td>92</td>
<td>bb</td>
<td>dd</td>
<td>4c</td>
<td>7f</td>
<td>11</td>
<td>9c</td>
<td>5c</td>
<td>41</td>
<td>1f</td>
<td>10</td>
<td>5a</td>
</tr>
<tr>
<td>d</td>
<td>ca</td>
<td>c1</td>
<td>31</td>
<td>08</td>
<td>a5</td>
<td>cd</td>
<td>7b</td>
<td>bd</td>
<td>2d</td>
<td>74</td>
<td>d0</td>
<td>12</td>
<td>b3</td>
<td>e6</td>
<td>b4</td>
</tr>
<tr>
<td>e</td>
<td>89</td>
<td>69</td>
<td>97</td>
<td>4a</td>
<td>0c</td>
<td>96</td>
<td>77</td>
<td>7e</td>
<td>85</td>
<td>b9</td>
<td>11</td>
<td>09</td>
<td>c5</td>
<td>6e</td>
<td>c6</td>
</tr>
<tr>
<td>f</td>
<td>18</td>
<td>f0</td>
<td>7d</td>
<td>ec</td>
<td>3a</td>
<td>dc</td>
<td>4d</td>
<td>20</td>
<td>79</td>
<td>ee</td>
<td>5f</td>
<td>3e</td>
<td>d7</td>
<td>cb</td>
<td>39</td>
</tr>
</tbody>
</table>

For example, if the input to the Sbox is ‘ef’, then go to e-th row and f-th column, to find Sbox (‘ef’) = ‘84’.

c) Fundamental Operations:

The two fundamental operations used by this algorithm are: The bitwise XOR of two 32-bit vectors and circular shift of a 32-bit word.

2. DESIGN OF SMS4 MODULE

The SMS4 module architecture in microblaze is shown in Fig.3, which uses SMS4 algorithm implemented in c programming language, is running on microblaze processor at 50MHz. RS232 is connected to microblaze peripheral local bus.
As shown in fig.4, SMS4 algorithms consist of encryption and key expansion algorithm. The key expansion circuit performs the key expansion function and the round key registers stores the round keys. Encryption round function circuits using the round keys and the last round result, calculate the input of the next round. Because decryption uses the encryption round keys in the reverse order, the key expansion function must be calculated only once. Hence, the stored round keys are used for both encryption and decryption SMS4 module architecture approach occupies smaller area.

2.1 RESULT OF SMS4 ALGORITHM

Encryption and decryption operation of SMS4 algorithm is implemented in c programming language and result is tested using RS232. Result is shown in figure 5.

3. SOFTWARE PROFILING OF SMS4 ALGORITHM

Software profiling allows to learn where program spent its time and which functions called which other functions while it was executing. This information can show which pieces of program are slower than expected, and might be candidates for rewriting to make program execute faster. It can also tell which functions are being called more or less often than expected. This may help spot bugs that had otherwise been unnoticed. [22]

SMS4 software will run for profiling by enabling ‘-pg’ flag. Program will write the profile data into a file called gmon.out. Profiling result of SMS4 algorithm is shown in fig.6, states time consumption of different function of SMS4 module.

3.1 SOFTWARE PROFILING RESULT

SMS4 software will run for profiling by enabling ‘-pg’ flag. Program will write the profile data into a file called gmon.out. Profiling result of SMS4 algorithm is shown in fig.6, states time consumption of different function of SMS4 module. As shown in fig. 7, SMS4 cryptography function takes 849.999us per call and SMS4 Key expansion function takes 899.99us per call and remaining functions takes 263.917us per call. Total execution time is 2013.915 us. Also percentage utilization of c code is 8.64% time. Custom hardware of key expansion function as time consuming block can be interface with microblaze processor.

4. PROPOSED SMS4 ARCHITECTURE BASED ON PROFILING RESULT

Based on profiling result, the proposed architecture of SMS4 algorithm in microblaze is shown in Fig. 7(a), which uses C code of encryption/decryption block is running on microblaze, and hardware of time consuming part (key expa-
5. PROFILING RESULT OF PROPOSED ARCHITECTURE

It shows that SMS4 cryptography function takes 833.33us per call and key expansion function takes 208.33 us per call and remaining function takes 285.652 us per call. Total execution time is 1327.31 us. Also percentage time utilization of c code is 5.47% of total execution time.

6. CONCLUSION

This paper has described the process of speed optimization of SMS4 algorithm based on hardware software co-design. Since only software implementation results in slower speed, so to increase the computational speed custom hardware of time consuming block is designed and interface with customized Microblaze processor.

Results shows that only software implementation(implementation of whole cryptographic algorithm in c programming language) takes 2013.915 us time and proposed hardware software co-design architecture takes 1327.31 us time. That is, proposed architecture saves 686.605 us. Therefore, proposed architecture of hardware software co-design of cryptographic algorithm, increases computational speed.
REFERENCES

[19] SMS4 Encryption Algorithm for Wireless Networks Translated and typed by Whit_eld D i_e of Sun Microsystems and George Ledin of Sonoma State University 15 May 2008
[20] The Institute of Electrical and Electronics Engineers (IEEE), http://grouper.ieee.org/groups/802/11
[22] EDK PROFILING USER GUIDE, www.xilinx.com