Cryptographic Functions Using GPU with CUDA

Miloš Radmanovic, Radomir S Stankovic
Faculty of Electronic Engineering, Aleksandra Medvedeva 14
18000 Niš, Serbia
{milos.radmanovic@elfak.ni.ac.rs}
{radomir.stankovic@gmail.com}

ABSTRACT: *Cryptographic Boolean functions are required for GPU. They represent a small part of the total cryptographic functions. While we search all functions and identify a few bent functions, we can use them effectively. The search of these variables take more time and processing time. We in this work proposed a parallel algorithm for listing of bent functions using GPU with CUDA. When the fast spectral transform calculation is used the few limitations can be satisfied by bent functions. We did testing and found that the proposed GPU-based algorithm can efficientlyenumerate the bent functions of 6 to 10 variables.*

Keywords: Cryptography, Boolean Function, Bent Function, Enumeration, GPU, CUDA

Received: 18 November 2020, Revised 3 March 2021, Accepted 12 March 2021

DOI: 10.6025/jism/2021/11/2/33-39

1. Introduction

Bent Boolean functions are functions with maximum nonlinearity. They ensure the cryptographic effectiveness and they can resist to various cryptanalysis attacks. Furthermore, they are also applied in many other areas such as coding theory, logic synthesis, and digital telecommunications [1]. Bent functions have specific properties and characterizations. The most common characterization for Boolean bent functions is the equal absolute values of all coefficients of their Walsh spectra [1]. They exist only for the even number of variables. There is no precise definition of the structure of bent functions. In general, it is unknown the complete enumeration and classification of them [2].

Finding the complete set of bent functions for a given number of inputs is an open problem and known are the lower and upper bounds in terms of the number of inputs [1]. There is no a formal method for enumeration, generalization, construction, or classification of all bent functions for the given number of inputs. Thus, during recent years, it has been developed a large number of methods for enumeration, construction, and etc., of particular bent functions that have specific characterizations [2]. However, specific bent functions are very rare and they make very small subset of the total number of bent functions, especially for large number of variables.

Therefore, the only possible method for the complete enumeration of bent function is obtained by using the exhaustive search of all possible functions. Testing of bentness across all possible functions, even for small numbers of variables, requires a lot of processing time. Consequently, the number of n-variable Boolean bent functions is known only for $n \le 8$.

The general number of bent functions is an open problem. Note that, the number of Bent functions increases rapidly with increasing n. It is known that, there are 8 bent functions in two variables, 896 bent functions in four variables, 5.425.430.528 bent functions in 6 variables [3], and 99.270.589.265.934.370.305.785.861.242.880 bent functions in 8 variables [4].

There are two types of methods for enumerations of Boolean bent functions, primary and secondary. The primary methods are based on the direct enumeration in Boolean domain [5]. The secondary methods are based on algebraic normal form or enumeration in Reed-Muller domain [4], [6], [7]. Almost all secondary methods used property that all bent functions of n variables have algebraic degree at most $n/2$. The amount of computations for enumerations of bent functions extremely increases, especially in the case of functions with large number of variables. For example, secondary method described in [4], for complete enumeration of bent functions on 8 variables has been used approximately 50 PCs running for 3 months.

The secondary methods for enumerations of bent functions in Reed-Muller domain can be very CPU time consuming and processing time is often the limiting parameter for practical applications. In [8], it is shown that there is significant benefit by using a reconfigurable SRC-6 computer to enumerate bent Boolean functions for cryptographic applications. Further, in [9], it is shown that the speed at which bent functions can be enumerated was improved using the circular pipeline implemented on FPGA.

The efficiency of using parallel multi-core CPU technique for random generation of bent function in Reed-Muller domain is analyzed in [10]. The GPUs are also an attractive target for parallel computations because of its high performance and low cost. Recent generations of GPUs have become programmable, enabling the use of GPUs for general purpose computations. The efficiency of using parallel GPU technique for random generation of bent function in Reed-Muller domain is analyzed in [11], where it is reported significant execution speedup on a GPU with 8 multiprocessors and 384 cores. Therefore, in this paper we proposed a parallel algorithm for enumeration of bent functions using GPU platform. The method is based on the usage of certain restrictions that should be satisfied by bent functions in the spectral Reed-Muller domain. The proposed method uses computation of the fast Reed-Muller transform and the computation of the fast Walsh transform.

For experiments, we developed two independent implementations, a single-core implementation using C++ and a GPU-based implementation using CUDA framework. Experimental results show that the proposed parallel algorithm using GPU platform offers computational speedups for hundreds times over performing the same algorithms on CPUs. As the Boolean function size increases, the number of bentness tests extremely increases. For this reason it is experimented with Boolean functions of the small sizes with computations restriction.

2. Preliminaries

The Reed-Muller (RM) spectral transform represents an important operator for obtaining AND-EXOR expressions of Boolean functions. The RM transform matrix of order n, denoted by $R(n)$, is defined recursively as [12]:

$$R(n) = \bigotimes_{i=1}^{n} R(1), \quad R(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \qquad (1)$$

For the bentness test, it is needed the inverse RM spectral transform. Since the RM transform matrix $R(n)$ is a selfinverse matrix over $GF(2)$, the forward and inverse RM transform are equal.

The Walsh spectral transform matrix of order n in the Hadamard ordering, denoted by $W(n)$, is defined as [12]:

$$W(n) = \bigotimes_{i=1}^{n} W(1), \quad W(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \qquad (2)$$

The spectrum of a Boolean function f given by truth vector $F = [f(0), f(1), ..., f(2^n - 1)]^T$ is computed as [12]:

$$S_f = T(n)F \;, \tag{3}$$

where $T(n)$ is any of the matrices $R(n)$, and $W(n)$, with computations performed in $GF(2)$ for the RM transform, and in the set of integer numbers for the Walsh transform. The recursive definition of $R(n)$, and $W(n)$ is the fundamental for the definition of the fast RM and fast Walsh spectral transform algorithm [12], similar to the fast Fourier transform (FFT) algorithm. Figure 1 shows the "butterfly" operations for $R(1)$, and $W(1)$ [12]. The fast spectral transform algorithm reduces computational complexity of a spectral transform from $O(2^{2n})$ to $O(2^{2n}log_2 2^n)$[12].

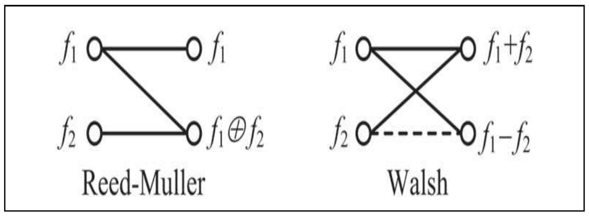

Figure 1. The Elementary "butterfly" operations for $R(1)$, and $W(1)$

The computation of the fast spectral transform algorithm consists of the repeated application of the same "butterfly" operations.

A Boolean function $f(x_1, x_2, ..., x_n)$ in (1, -1) encoding is called bent if all elements of the Walsh spectrum vector $S_{f,\,W}$ have the same absolute value $2^{n/2}$ [1].

For example, for a bent function of four variables $f(x_1, x_2, x_3, x_4)$, given by $F= [1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1]^T$, the Walsh spectrum with (1,-1) encoding can be computed as $S_{f,\,W}= W(4)F$, where result is: $S_{f,\,W}= [4, -4, -4, -4, -4, -4, 4, 4, -4, 4, -4, 4, -4, 4, 4, -4]^T$ and all elements have the same absolute value 4.

The Reed-Muller form of a Boolean function f, also called AND-EXOR expression, can be computed from the RM spectrum vector $S_{f,\,RM}$ [12]:

$$f(x_1, x_2, \ldots, x_n) = X(n)S_{f,RM} \tag{4}$$

where

$$X(n) = \bigotimes_{i=1}^{n} \begin{bmatrix} 1 & x_i \end{bmatrix} \tag{5}$$

and where addition and multiplication are modulo 2.

For example, for a bent function of four variables $f(x_1, x_2, x_3, x_4)$, from previous example, the RM spectrum with (1,-1) encoding can be computed as $S_{f,RM} = RM(4)F$, where result is: $S_{f,RM} = [1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0]^T$. The elements of vector $X(n)$ are $X(4) = [1, x_4, x_3, x_3x_4, x_2, x_2x_4, x_2x_3, x_2x_3x_4, x_1, x_1x_4, x_1x_3, x_1x_3x_4, x_1x_2, x_1x_2x_4, x_1x_2x_3, x_1x_2x_3x_4]^T$.

The corresponding RM form is: $f = 1 \oplus x_2x_3 \oplus x_1x_4$.

The algebraic degree of a product term in a RM form is the number of variables in that term. The algebraic degree of a RM form $deg(f)$ is the number of variables in the product term with the maximum algebraic degree.

Algebraic degree $\deg(f)$ of a bent function $f(x_1, x_2, \ldots x_n)$ is at most $n / 2$ for $n > 2$ [2].

Note that the bent function from the previous example can be characterized by the upper bound of the algebraic degree of the RM form: $\deg(f(x_1, x_2, x_3, x_4) = 1 \oplus x_2x_3 \oplus x_1x_4) = 2 \leq 4/2$.

3. Enumeration of Bent Functions in Reedmuller Domain

The algorithm for enumeration of bent functions in the Reed-Muller domain takes as its input only the number of function variables. Since the algebraic degree of an n-variable bent function is less or equal to $n/2$, the number of non-zero elements of RM spectrum vector is limited and their positions in the RM spectrum vector are restricted.

For example, for a bent function of four variables $f(x_1, x_2, x_3, x_4)$, elements of the vector $X(n)$ are $X(4) = [1, x_4, x_3, x_3x_4, x_2, x_2x_4, x_2x_3, x_2x_3x_4, x_1, x_1x_4, x_1x_3, x_1x_3x_4, x_1 x_2, x_1 x_2 x_4, x_1x_2x_3]^T$ where the number of variables in a product term is less or equal to 2. For this reason the following positions in the RM spectrum vector are restricted for non-zero elements and other positions are always zero: $S_{f,RM}$ = [-,-,-,-,-, -,-,0,-,-,-,0,-,0,0,0]^T There are 11 of 16 positions in the RM spectrum vector of bent functions available for non-zero elements.

Therefore, we define the restricted RM spectrum of a bent function $f(x_1, x_2, x_3, x_4)$ denoted by vector $SR_{f,RM} = [S_{f,RM}(0), S_{f,RM}(1), S_{f,RM}(2), S_{f,RM}(3), S_{f,RM}(4), S_{f,RM}(5), S_{f,RM}(6), S_{f,RM}(8), S_{f,RM}(9), S_{f,RM}(10), S_{f,RM}(12)]^T$. It should be noticed that elements of the restricted RM spectrum have the number of ones in the binary representation of the RM spectrum vector index less or equal to 2. Thus, the number of elements of the restricted RM spectrum is 11.

The size of the restricted RM spectrum of n-variable bent function is

$$\sum_{i=0}^{n/2} \binom{n}{i}$$

For example, the size of the restricted RM spectrum in relation to the size of RM spectrum for bent functions with 8 variables is 163 out of 256, and with 10 variables is 638 out of 1024.

Starting from this observation, the search space is defined by the size of the restricted RM spectrum, where elements of spectrum takes values from $[0, 0,...,0, 0]^T$ to $[1, 1,...,1, 1]^T$. For example, the size of search space for bent functions with 8 variables is 2^{163}, and with 10 variables is 2^{638}.

An outline of the algorithm for the enumeration of bent functions in Reed-Muller domain is given as Algorithm 1.

Algorithm 1

```
1: Set the number of function variables $n$ and set the counter of bent functions to 0.
2: Calculate the size of the restricted RM spectrum.
3: Generate vector IND of the RM spectrum vector indices for translations from the restricted RM spectrum to the RM spectrum.
4: Execute for loop where elements of the restricted RM spectrum take values from 0,0,...,0,0 to 1,1,...,1,1.
4: Translate the restricted RM spectrum to the RM spectrum by using vector IND.
5: Compute the truth-vector from the RM spectrum by using the Fast RM transform.
6: Do (1, -1) encoding of the truth-vector.
7: Compute the first, second, and last element of the Walsh spectrum from truth-vector using rows of the Walsh spectral transform matrix [11], and test if the element has the absolute value $2^{n/2}$. If the bentness test is sucessful, increase the counter of bent functions and go to the step 4.
```

8: Compute all elements of the Walsh spectrum from truthvector by using the Fast Walsh transform, and test if all elements have the absolute value $2^{n/2}$. If the bentness test is sucessful, increase the counter of bent functions and go to the step 4.

9: Obtain the number of bent functions by reading the counter value.

---

4. GPU-based Algorithm for Enumeration of Bent Functions

For GPU architectures, the model of parallel processing is based on a large number of processor cores with the ability todirectly address into a shared GPU RAM memory. This organization of computations allows to have a large number of threads performing the same operations on different data simultaneously [13]. Enumeration of bent function in the Reed-Muller domain uses computations of the fast Reed- Muller transform and the fast Walsh transform. Running FFTlike algorithms on a GPU platform can give a better performance compared to a singlecore CPU or a multicore CPU platform. FFT-like algorithms applied on small vectors can be more efficient performed on the CPU compared to GPU [13]. If on the other hand, there is the need to perform large amount of FFT-like algorithms applied on small vectors with minimal moves to/from the GPU, then the GPU-based algorithm is the optimal choice.

Thus, in this paper we proposed GPU-based algorithm for enumeration of bent functions in the RM domain. Details of algorithm are given as Algorithm 2.

Algorithm 2

1: Set the number of function variables n and set the counter of bent functions to 0.

2: Calculate the size of restricted RM spectrum.

3: Generate vector IND of RM spectrum vector indices for translations from the restricted RM spectrum to the RM spectrum.

4: Alloc GPU memmory and copy data stored in CPU memory to the GPU memory.

5: Split the loop into multiple loops according to maximal number of GPU threads (determined by the GPU hardware) where elements of the restricted RM spectrum take values from 0,0,...,0,0 to 1,1,...,1,1.

6: Use threads id and the number of splits to work out which loop iterations of the restricted RM spectrum to perform.

7: Translate the restricted RM spectrum to the RM spectrum using vector IND.

8: Computae of the truth-vector from the RM spectrum by using the Fast RM transform.

9: Do (1,-1) encoding of the truth-vector.

10: Compute the first, second, and last element of the Wlash spectrum from the truth-vector using rows of the Walsh spectral transform matrix [11], and test if the element has the absolute value $2^{n/2}$. If the bentness test is sucessful, increase the counter of bent functions and go to the step 6.

11: Compute of the all elements of the Walsh spectrum from the truth-vector using fast Walsh transform, and test if all elements have the absolute value $2^{n/2}$. If the bentness test is sucessful, increase the counter of bent functions and go to the step 6.

12: Copy data stored in the GPU memory back to the CPU memory.

```
13: Obtain the number of bent functions by reading the counter value.
```

Note that steps 2 to 8 of the Algorithm 2 are executed on GPU and the number of actual threads that are simultaneously active in hardware depends on the used GPU. For example, with NVidia GTX560Ti GPU, the maximal number of threads is 65535*1024 = 67107840, and the maximal number of simultaneously active threads is 384 (determined by number of cores).

5. Experimental Results

CUDA framework gives to program developers a direct access to the virtual instruction set and memory of the parallel computational elements in NVidia GPUs. For this reason, it is developed a referent single-core CPU C++ and CUDA implementation on GPU platform of the algorithm for enumeration of bent function. The single-core CPU implementation is performed on an Intel i7 CPU at 3.66 GHz. The CUDA implementations on GPU platform is performed on NVidia GeForce GTX-560Ti on 900 MHz with 1 GB DDR5 4 GHz RAM (8 multiprocessors and 384 cores).

Table 1 shows computation performance of the algorithm for enumeration of bent function using referent single-core CPU C++ and the GPU-based implementations. The presented values are computational times for 1 million bentness tests of functions. From data in Table 1, it can be seen that on GPU platform, for all the computations, CUDA implementation of the algorithm significantly reduces computation times when compared to the CPU implementation. Note that for the largest Boolean functions, experiments were not performed, due to very long CPU computation time.

Num. of function variables	**Computation time for 1 million bentness tests [s]**	
	CPU i7-3.66	**GPU GTX-560Ti**
6	0.498	0.009
8	1.778	0.017
10	7.566	0.041

Table 1. Computation Performance of Algorithm For Enumeration of Bent Functions Using C++ and Cuda Implementations

4. Conclusion

This paper proposes an algorithm for enumeration of bent functions based on the exhaustive search of all possible functions in the Reed-Muller domain by using GPU and CUDA. The ultimate goal is to exceed the computation performance for finding the total number of bent functions, since the computation time for the enumeration is exponential in the number of variables in the function. The proposed implementation exploits the parallel mapping of exhaustive search algorithm for the enumeration of bent functions in Reed-Muller domain by performing the same operations over different data simultaneously which is a good match to GPU hardware.

The experimental results confirm that the application of the proposed GPU-based algorithm leads to significant computational speedups. It is also confirmed that this implementation could be especially efficient for functions with large number of variables. From the results, it is also evident that CUDA framework can be efficiently used in implementation of the proposed GPU-based algorithm.

Future work will be directed towards the extension of the proposed technique to the enumerations various other classes of Boolean functions.

Acknowledgements

The research reported in this paper is partly supported by the Ministry of Education and Science of Serbia, projects ON174026 (2011-2017) and III44006 (2011-2017).

References

[1] Mesnager, S. (2016). *Bent Functions: Fundamentals and Results*, Springer International Publishing, 2016.

[2] Tokareva. N. (2016). *Bent Functions, Results and Applications to Cryptography*, Academic Press, 2015.

[3] Sasao, T., Butler, J., Thornton, M. (2010). *Progress in Applications of Boolean Functions*, Morgan and Claypool Publishers, 2010.

[4] Langevin, P., Leander, G. (2011). Counting All Bent Functions in Dimension Eight 99270589265934370305785861242880, *Designs, Codes and Cryptography*, vol. 59, *p* 193-201, 2011.

[5] Tokareva, N. (2011). On the Number of Bent Functions From Iterative Constructions: Lower Bounds and Hypotheses, *Advances in Math. of Communications*, vol. 5, no. 4, p 609-621, 2011.

[6] Meng, Q., Yang, M., Zhang, H., Cui, J. (2008). A Novel Algorithm Enumerating Bent Functions, *Discrete Math.*, vol. 308, no. 23, p 5576-5584, 2008.

[7] Kolomeets, N. (2012). Enumeration of the Bent Functions of Least Deviation From a Quadratic Bent Function, *Journal of Applied and Industrial Mathematics*, vol. 6, no. 3, pp. 1990-4797, 2012.

[8] Shafer, J., Schneider, S., Butler, J., Stanica, P. (2010). Enumeration of Bent Boolean Functions by Reconfigurable Computer, *18th* IEEE Annual International Symposium on Field-Programmable *Custom Computing Machines*, p 265-272, Charlotte, NC, USA, 2010.

[9] Schneider, S., Butler, J. (2016). Bent Function Enumeration by a Circular Pipeline Implemented on an FPGA, *Proc.* 12*th* *Int. Workshop on Boolean Problems (IWSBP2016)*, p 159-166, Freiberg, Germany, 2016.

[10] Radmanovi, M., Stankovi, R. (2016). Random Generation of Bent Functions on Multicore CPU Platform, *Int. Sci. Conf. on* Inf. Communication and Energy Systems and Technologies (*ICEST 2016*), p 239-242, Ohrid, Macedonia, 2016.

[11] Radmanovi, M. (2016). Efficient Random Generation of Bent Functions using GPU Platform, *12th Int. Workshop on Boolean Problems* (*IWSBP2016*), p 167-173, Freiberg, Germany, 2016.

[12] Thornton, M. A., Drechsler, R., Miller, D. M., *Spectral Techniques in VLSI CAD*, Springer, 2001.

[13] Yuen, D.A., Wang, L., Chi, X., Johnsson, L.,Ge, W., Yaolin, S. (2013). *GPU Solutions to Multi-scale Problems in Science and Engineering*, Springer Science & Business Media, 2013.