GPU Acceleration of Cube Calculus Operations

Vinay Bhardwaj *Vamsi Parasa*
Department of Electrical Engineering, Portland State University, USA
{vinayb, parasa}@cecs.pdx.edu

***Abstract*—In this paper, we present the first massively parallel, GPU accelerated implementation of the Cube Calculus operations for multivalued and binary logic, also called Cube Calculus machine (CCM). Substantial speedups upto the order of 80x are achieved using the CUDA based nVIDIA Tesla GPU compared to the CPU implementation on a sequential processor. It must be noted that an earlier effort by the research group to develop a highly optimised FPGA implementation resulted in speedups only of the order of at most 2.5x. Cube Calculus is a very efficient and convenient mathematical formalism for representation, processing and synthesis of binary and multivalued logic which has significant applications in logic synthesis and machine learning. Thus, massive speedups achieved using GPUs are very encouraging to build future parallel VLSI electronic design automation (EDA) systems.**

I. Introduction

Logic synthesis is a very fundamental and computationally intensive phase of VLSI design. A lot of different parallel programming paradigms were considered before for the task of Electronic Design Automation (EDA) and were not successfully integrated into the modern VLSI EDA tools. A lot of scientific groups have successfully leveraged the massive parallelism and speedups offered by GPUs but so far very few research groups have explored the area of GPU based VLSI design automation.

Cube Calculus (CC) has been demonstrated to be a very elegant mathematical formalism for multi-valued and binary logic synthesis [1]. Our research group being a very active participant in this field of research for the last 15 years [1],[2],[3] and having realised the importance of a fast Cube Calculus machine (CCM) have also worked on a highly optimised FPGA implementation [2][3] but have not been able to achieve significant speedups beyond the order of 2x. This naturally motivated us to explore the GPU computing paradigm and we were able to achieve significant speedups of the order 45x to 80x (depending on the type of Cube Calculus operation implemented).

The organization of the paper is as follows: first, we briefly introduce the basics of Cube Calculus operations and next present the results of our GPU implementation of the cube calculus machine (CCM) on an nVIDIA Tesla GPU using CUDA.

II. Cube Calculus

The theory of CC is presented in great detail in [1],[2],[3] and due to space restrictions we present a very concise introduction to CC. Intuitively, given an n variable boolean function, it can be mapped on to an n dimensional boolean cube (B^n) as shown in Fig. 1a. Thus a cube can be considered as a set of vertices on such a Boolean cube. A cube has a more general definition in multivalued and binary logic and can represent not only a product of literals as in the standard binary logic but can also be a sum or XOR of literals. In Fig. 1b, we show a set of cubes denoted by the product of literals *ab, bc, ca* (shown in circles) representing the cover for the ON set of the function $f=ab+bc+ca$ .The above mentioned idea can be easily extended to the case of multi-valued (MV) logic where a cube represents a subspace in an n dimensional MV space and we now define Cube calculus formally. A cube calculus is a system of a) set of cubes b) set of operations on the cubes.

A. Definition of a Cube

Consider n discrete variables $X_1, X_2, \ldots, X_n$ such that each variable Xi can take values from a certain finite discrete set Vi (Vi can be any finite set of symbols say {0,1} in the case of binary logic). Let S_i be subset of V_i, then a literal $X_i^{S_i}$ represents a characteristic function of variable X_i , i.e. the literal's value is 1 for symbols from this subset. For example, for binary logic, $X^1 = X$, and $X^0 = X'$ are two literals; and for four-valued logic $V_i = \{0, 1, 2, 3\}$: $Y^{\{0,2\}}$ is a literal and equals 1 if $Y \in \{0, 2\}$ or 0 if $Y \in \{1, 3\}$. A cube on $X_1, X_2, \ldots, X_n$ is an ordered set of literals on $X_1, X_2, \ldots, X_n$, that is, $X_1^{S_1}, X_2^{S_2}, \ldots\ldots, X_{n-1}^{S_{n-1}}, X_n^{S_n}$.

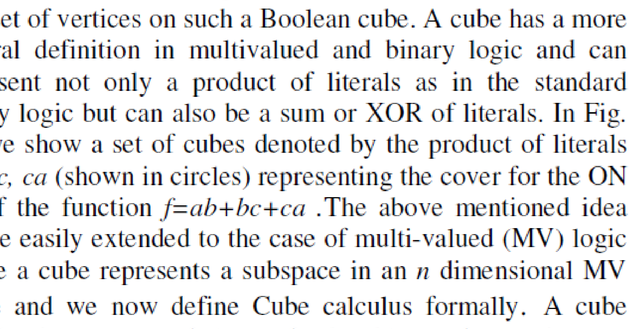

Fig.1. (a) A boolean cube B^n for n = 0,1,2,3. (b) A function $f=ab+bc+ca$ represented on a 3 cube and the cubes *ab,bc,ca* represented by circles.

B. Cube Calculus Operations

A lot of CC operations [2],[3] can be defined or invented but the commonly used ones can be classified into three types and some examples of the operations are illustrated in Table I.

Every CC operation, in general, has two steps *i*) checking for a relation R *ii*) Applying an output operation, which can be different for each of the variables X_i based on the outcome of the relation R. We first fix the notation by considering two cubes A and B defined on n discrete variables $X_1, X_2, \ldots, X_n$ Let $A = X_1^{SA_1} X_2^{SA_2} \ldots X_{n-1}^{SA_{n-1}} X_n^{SA_n}$ *and* $B = X_1^{SB_1} X_2^{SB_2} \ldots\ldots X_{n-1}^{SB_{n-1}} X_n^{SB_n}$.

TABLE I

Types of Cube Calculus Operations

Type of Operation	Example
Simple combinational	Supercube, Intersection
Complex combinational	Prime, Cofactor, Consensus
Sequential Operation	Sharp, Crosslink

In a *Simple combinational operation*, there is no relation R to be checked and the same operation is applied to the individual variables and produce only one output cube. For example consider the Supercube operator defined by $C = A \cup B$ gives $C = X_1^{(SA_1 \cup SB_1)} X_2^{(SA_2 \cup SB_2)} \ldots X_{n-1}^{(SA_{n-1} \cup SB_{n-1})} X_n^{(SA_n \cup SB_n)}$ Similarly, the Intersection operation denoted by $\cap$ is defined as follows:

$$C = A \cap B = \begin{Bmatrix} \varnothing, & \text{if } A \cap B = \varnothing \\ else, & X_1^{(SA_1 \cap SB_1)} X_2^{(SA_2 \cap SB_2)} \ldots X_{n-1}^{(SA_{n-1} \cap SB_{n-1})} X_n^{(SA_n \cap SB_n)} \end{Bmatrix}$$

In *Complex combinational operations*, we apply a relation R on all the variables X_i and for those variables which satisfy the relation, also called as "Active positions", we apply an operator called "Active" and for the rest we apply "Non-active" operator. Table II lists the other complex combinational operations. For example for the Prime operation denoted by A'B the relation is $X_i^{SA_i} \cap X_i^{SB_i} \neq \varnothing$, the active operator is $X_i^{SA_i} \cup X_i^{SB_i}$ and the Non-active operator is $X_i^{SA_i}$ .For example, consider two ternary logic cubes denoted by, $A = X_1^{\{0,1,2\}} X_2^{\{0,1\}} X_3^{\{0,2\}}$, $B = X_1^{\{0,1\}} X_2^{\{2\}} X_3^{\{0,1\}}$. Then we notice that the relation $X_i^{SA_i} \cap X_i^{SB_i} \neq \varnothing$ is satisfied only for the variables X_1, X_3 but not for X_2 . He we apply $X_i^{SA_i} \cup X_i^{SB_i}$ for X_1, X_3 and $X_i^{SA_i}$ for X_2 (i.e. $X_2^{SC_2} = X_2^{SA_2}$). So, $C = X_1^{(\{0,1,2\} \cup \{0,1\})} X_2^{(\{0,1\})} X_3^{(\{0,2\} \cup \{0,1\})} = X_1^{(\{0,1,2\})} X_2^{(\{0,1\})} X_3^{(\{0,1,2\})}$.
Table II shows various Complex CC operations.

TABLE II

Complex combinaitonal operations

Operation	Definition		
Cofactor A| B	A $\|_{basic}$ B (defined below) A $\cap$ B $\neq \varnothing$ $\varnothing$ otherwise		
Consensus A* B	A $\cap$ B when distance (A,B) = 0 A * B = $\varnothing$ when distance (A,B) > 1 A $*_{basic}$ B when distance (A,B) = 1		
Name, Notation	**Relation**	**Active**	**Non-active**
Prime A'B	$X_i^{SA_i} \cap X_i^{SB_i} \neq \varnothing$	$X_i^{SA_i} \cup X_i^{SB_i}$	$X_i^{SA_i}$
Consensus A $*_{basic}$ B	1	$X_i^{SA_i} \cup X_i^{SB_i}$	$X_i^{SA_i} \cap X_i^{SB_i}$
Cofactor A $\|_{basic}$ B	$X_i^{SA_i} \supseteq X_i^{SB_i}$	U (universal set)	$X_i^{SA_i} \cap X_i^{SB_i}$

In *Sequential operations*, we get more than one resultant cube. We apply a relation R on each variable X_i and we get as many resultant cubes as the number of Active positions. For each Active position, we apply an "Active" operator on X_i and all variables preceding it i.e. X_k $(1 \leq k < i)$, we apply an "After" operator and variables succeeding it i.e. X_k $(i < k \leq n)$, we apply a "Before" operator. Table III gives the details of various sequential operators.
We provide one brief example of Disjoint Sharp operator denoted by

$$A \# B = \begin{Bmatrix} A, & \text{if } A \cap B = \varnothing \\ \varnothing, & \text{if } A \subseteq B \\ A \#_{Dbasic} B, & \text{otherwise} \end{Bmatrix}$$

For the $\#_{Dbasic}$ (defined in Table III), the relation R is given by $\Gamma\left(X_i^{SA_i} \subseteq X_i^{SB_i}\right)$, where Γ indicates the compliment operator. Let $A = X_1^{\{0,1,2\}} X_2^{\{0,1\}} X_3^{\{0,2\}}$ and $B = X_1^{\{0,1\}} X_2^{\{2\}} X_3^{\{0\}}$ .We see that X_1, X_3 but not X_2 satisfy the relation, so we get two result cubes C_1, C_2 as there are two active positions. Following Table III, we obtain the following resultant cubes.

$$C_1 = X_1^{(\{0,1,2\} \cup \Gamma\{0,1\})} X_2^{(\{0,1\})} X_3^{(\{0,2\})} = X_1^{(\{0,1,2\})} X_2^{(\{0,1\})} X_3^{(\{0,2\})}$$
$$C_2 = X_1^{(\{0,1,2\} \cap \{0,1\})} X_2^{(\{0,1\} \cup \Gamma\{2\})} X_3^{(\{0,2\})} = X_1^{(\{2\})} X_2^{(\{0,1\})} X_3^{(\{0,2\})}$$

TABLE III

Sequential operations

Name, Notation, Relation	After	Active	Before
Crosslink (A c B) $X_i^{SA_i} \cap X_i^{SB_i} = \varnothing$	$X_i^{SB_i}$	$X_i^{SA_i} \cup X_i^{SB_i}$	$X_i^{SA_i}$
Sharp (A # $_{basic}$ B) $\Gamma\left(X_i^{SA_i} \subseteq X_i^{SB_i}\right)$	$X_i^{SA_i}$	$X_i^{SA_i} \cup \Gamma X_i^{SB_i}$	$X_i^{SA_i}$
Disjoint Sharp (A # $_{Dbasic}$ B) $\Gamma\left(X_i^{SA_i} \subseteq X_i^{SB_i}\right)$	$X_i^{SA_i} \cap X_i^{SB_i}$	$X_i^{SA_i} \cup \Gamma X_i^{SB_i}$	$X_i^{SA_i}$

III. GPU Implementation using CUDA

In order to implement the CC operations, the kernel is designed such that, each thread parallely works on the implementation of CC operations on one variable X_i .

The results of the GPU implementation are as follows: CUDA kernels were written to implement the Intersection, Supercube, Prime, Consensus and Cofactor CC operations. The kernels have been optimized for efficient memory coalescing and asynchronous memory transfers. Let the dimensions of cubes be denoted by (*N, m*), where *N* indicates the number of variables and *m* indicates the number of logic values. The kernels were tested on cubes of dimensions (N = {1024, 2048}, m = {2, 10, 20, 30, 40}). A significant speed-up in the range 45x to 80x is achieved, depending on the operation, when compared to a C program implementation on a CPU. The speedups were calculated using the 'user time' thereby taking into account the memory transfer time taken for moving data between the host and GPU. The results for (N = 2048, m = {2, 10, 20, 30, 40}) is shown in the Fig 2.

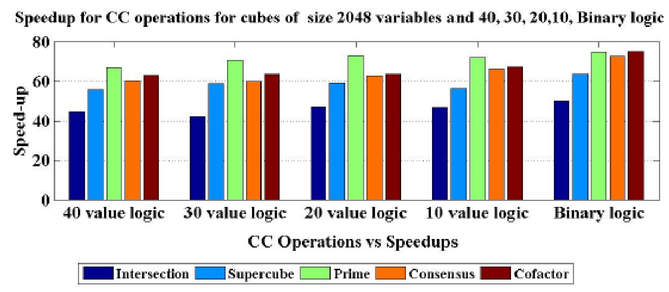

Fig. 2. Speedups for different CC operations for cubes of size 2048 variables

By interpolating the values of speedups obtained for (N=1024), in Fig. 3, we show as to how the speedups decrease linearly as m increases while N is constant. On an average, the speedup decreases by up to 10-15% depending on the operation, as the value of m is increased from 2 to 40.

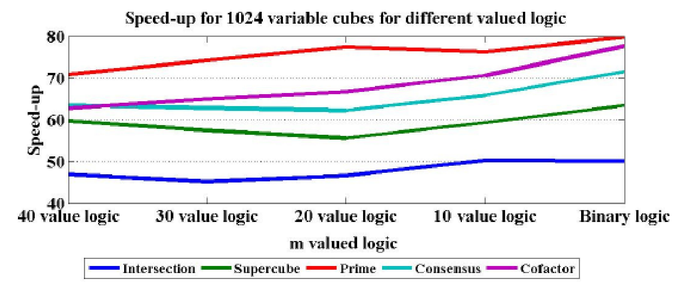

Fig. 3. Interpolated Speed-up on cubes of 1024 variables for different sizes of multivalued logic (m) .We see a linear decrease in speedup as m increases.

IV. Future Work and Conclusion

After implementing a very fast Cube Calculus Machine (CCM) achieving speedups up to 85x, the next step is to apply it for logic synthesis [1],[5] and machine learning [4],[2].All the operations of CCM are fundamental operations to perform logic synthesis [5] and in particular, from [5], we note that, the Consensus operator can be used for proving Tautology and finding prime implicants, the Crosslink operator can be used for ESOP minimization, the Sharp operator can be used for finding the minimum number of prime-implicants. Thus, we believe that a superfast CCM can really speed-up logic synthesis. We are currently working on implementing GPU based logic synthesis algorithms. CC operations can be easily extended to Image processing.

References

[1] Marek A. Perkowski, "A Universal Logic Machine," invited address, *Proc. of the 22nd IEEE International Symposium on Multiple Valued Logic, ISMVL'92,* pp. 262 - 271, Sendai, Japan, May 27-29, 1992. Available: http://web.cecs.pdx.edu/~mperkows/=PUBLICATIONS/=publications-1992.html

[2] Marek A. Perkowski, David Foote, Qihong Chen, Anas Al-Rabadi, Lech Józwiak: Learning Hardware Using Multiple-Valued Logic, Part 2: Cube Calculus and Architecture. IEEE Micro 22(3): 52-61 (2002)

[3] Q. Chen, "Realization of a Universal Cube Calculus Machine in DECPeRLe-1 FPGA Emulator," master's thesis, Dept. of Electrical and Computer Engineering, Portland State Univ., Ore., 1998. Available : http://web.cecs.pdx.edu/~mperkows/CLASS_VHDL/==projectCCM.html

[4] Marek A. Perkowski, David Foote, Qihong Chen, Anas Al-Rabadi, Lech Józwiak: Learning Hardware Using Multiple-Valued Logic, Part 1 : Introduction and Approach. IEEE Micro 22(3): 41-51 (2002)

[5] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.pp. 269-341(chap. 7)