International Journal of MC Square Scientific Research Vol.1, No.1,2009

High Performance Design Of Linear Algebra Operations For Image Processing In Fpga Implementation

Dr .M. Manikandan[1] and Mary Josephine Caroline[2]

Anna University, Guindy, Chennai-25

[2]CEG, Anna university,Ch-25

[1]maniiz@yahoo.com [2]marycaroline63@gmail.com

Abstract

Numerical linear algebra operations are key primitives in scientific computing. In this paper, performance optimizations of linear algebraic operations have been extensively investigated. FPGA-based high-performance designs are proposed for dot product, matrix-vector multiplication and matrix multiplication by identifying the parameters for each operation and analyzing the trade-offs. It is proposed to implement dot product in convolution filter, which is useful in noise removal, and Matrix multiplication in boundary tracing, which is useful for shape analysis and calculating geometric features. These high performance designs of linear algebra applications are proposed to be implemented on Xilinx FPGAs.

Keywords: FPGA, Convolution filter, Matrix multiplication.

1. Introduction

Field Programmable Gate Array (FPGA) work is characterized by a client's program as opposed to by the producer of the gadget. An average coordinated circuit plays out a specific capacity characterized at the season of fabricate. Conversely, a program composed by somebody other than the gadget maker characterizes the FPGA's capacity. Contingent upon the specific gadget, the program is either "singed" in for all time or semi-for all time as a major aspect of a load up get together process, or is stacked from an outside memory each time the gadget is

controlled up. This client programmability that is reconfigurable property gives the client access to complex incorporated plans without the high building costs related with application particular coordinated circuits.

A FPGA is like a PLD, while PLDs are for the most part constrained to several doors, FPGAs bolster a great many entryways. They are particularly prominent for prototyping incorporated circuit plans. Once the outline is set, hardwired chips are delivered for speedier execution. FPGAs are turning into a basic piece of each framework outline. Numerous merchants offer a wide range of structures and procedures.

In this venture configuration exchange offs and propose superior FPGA-based plans for a few direct variable based math operations, including speck item, lattice vector duplication, grid augmentation, are broke down. These operations fill in as essential building obstructs for some numerical direct variable based math applications, including the arrangement of straight frameworks of conditions, direct slightest square issues, and eigen esteem issues. For every operation, its intrinsic attributes, for example, the quantity of drifting point operations and I/O operations required, and recognize different plan parameters are broke down. By investigating the outline space, the plan tradeoffs among zone, dormancy, and capacity measure are computed., Then outline is proposed for every operation that successfully uses the accessible memory and accomplishes the ideal inactivity under the given equipment assets.

As the proposed plans are non specific, it can be connected to different FPGA gadgets. The outline for every operation is described through different parameters, for example, the quantity of coasting point units, the required stockpiling size, and the square size. The parameters can be tuned by the equipment asset limitations, including the accessible chip range, the span of accessible memory, and the quantity of I/O pins. For both spot item and grid vector increase, tree-based engineering and for framework duplication direct cluster design with various Processing Elements (PEs) are executed. Piece calculations are utilized to lessen the necessities on the capacity measure or the memory transfer speed.

The superior outline is actualized in the convolution channel and protest following and limit following framework to enhance the execution of picture handling application.

2. Related Works

The arrangement of Basic Linear Algebra Subprograms, which is normally alluded to as BLAS , has been utilized as a part of an extensive variety of programming given by the free programming Vendors. BLAS give building square schedules to performing essential vector and grid operations. Enhancements for the BLAS library on broadly useful processors incorporate circle unrolling, enroll blocking, and store blocking. A considerable lot of the advancements are stage particular . Map book was proposed which naturally creates and enhances numerical programming for processors with profound memory chains of command and pipelined useful units. FPGA-based outlines for a few BLAS operations, including speck item, grid vector augmentation, and framework increase are considered. These plans are gadget free and can adjust to different equipment asset requirements.

Another vital straight polynomial math library is the Linear Algebra PACKage (LAPACK)]. This library gives schedules to fathoming frameworks of synchronous straight conditions, slightest squares arrangements of direct frameworks of conditions, eigenvalue issues, and particular esteem issues. The outlines which were utilized already in BLAS uses the parallelism of various PC processors. Interestingly, this plan misuses the spatial and fleeting parallelism on one reconfigurable equipment gadget. Rather than utilizing message passing, numerous Processing Elements (PE) convey through directing wires on the gadget. In addition, not at all like the universally useful processors, FPGAs have coordinate control of their on-chip memory and locally available memory.Subsequently, it is expected to address another outline tradeoffs that does not exist in the present work on parallel and appropriated frameworks.

3. Linear Algebra On Fpgas

Numerous scientists have considered FPGA-based executions of straight variable based math operations on settled point number juggling as it were. Because of the usage intricacy of the drifting point units, the work of these analysts is not reasonable for coasting point-based operations. A plan for drifting point thick framework duplication is proposed for issue estimate n, the powerful inactivity of the outline O (n2), utilizing a capacity of size O (n2) , a square grid increase calculation was talked about for vast n and a gliding point Multiplier and Accumulator (MAC) was executed. Tree-based engineering was proposed for inadequate lattice vector increase (SMVM). The design will accomplish substantially higher execution than usage on the universally useful processors.

The outlines which were utilized already in BLAS uses the parallelism of various PC processors. Interestingly, this plan misuses the spatial and fleeting parallelism on one reconfigurable equipment gadget. Rather than utilizing message passing, numerous Processing Elements (PE) convey through directing wires on the gadget. In addition, not at all like the universally useful processors, FPGAs have coordinate control of their on-chip memory and locally available memory.Subsequently, it is expected to address another outline tradeoffs that does not exist in the present work on parallel and appropriated frameworks

The outline exchange offs for a few BLAS operations on reconfigurable equipment is broke down. In any case, just on-chip memory of the FPGA was utilized. In the outlines proposed in this venture likewise use the locally available memory appended to the FPGA. The potential limit of FPGAs in performing drifting point BLAS operations and contrasted the figuring limit of FPGAs and that of the broadly useful processors. The quantity of I/O pins was considered as an imperative. Investigation in view of different equipment asset imperatives are additionally given in the venture.

4. Proposed Design

A The FPGA gadget contains on-chip memory (BRAM) and approaches locally available memory (SRAM).The add up to size of the BRAM is considerably littler than that of the outlines which were utilized already in BLAS uses the parallelism of various PC processors. Interestingly, this plan misuses the spatial and fleeting parallelism on one reconfigurable equipment gadget. Rather than utilizing message passing, numerous Processing Elements (PE) convey through directing wires on the gadget. In addition, not at all like the universally useful processors, FPGAs have coordinate control of their on-chip memory and locally available memory.Subsequently, it is expected to address another outline tradeoffs that does not exist in the present work on parallel and appropriated frameworks

SRAM. Notwithstanding, the BRAM data transmission is significantly bigger than the SRAM transfer speed. The on-chip memory and the installed memory are utilized for inside capacity of the proposed designs. A parameterized and adaptable plan approach work is received. This approach comprises of two stages. In the primary stage, the portrayal of the operation, including the quantity of gliding point operations required and the measure of information to be traded with the outer memory are inspected. The plan parameters are then identified.and few of the parameters concern the coasting point units utilized as a part of the outline, for example, their pipeline delays; some are for the whole engineering, for example, the capacity measure and the quantity of drifting point units utilized. In light of the investigation of the outline space shaped by the plan parameters, an enhanced plan is proposed. In the second stage, a gadget is picked and the equipment limitations are resolved. The limitations determine the accessible range, the I/O stick tally, the extent of accessible memory, and the memory data transmission. A fitting esteems for the parameters in the outline are chosen . Consequently an ideal outline which is autonomous of the gadget utilized and after that adjust the plan to a particular equipment gadget is created.

OPERATION ANALYSIS

For dot product, the minimum number of external I/O operations is 2n+1. There are n fixed point multiplications and n fixed point additions. k and l are identified as the design

parameters because they affect Tcomp. Since there is no data reuse in this operation, no on-chip memory or onboard memory is needed by the design. To reduce the latency, we overlap Tcomp and TI/O. Furthermore, the floating-point multiplications and additions are overlapped. Therefore, the lower bound on the latency of the operation(in cycles) is derived as

$$T \geq \max(Tcomp, TI/O) \geq \max (\max (n/k, n/l), 2n+1/bw)$$

ARCHITECTURE

To accomplish the lower bound, k=l, k=b/m . An engineering which comprises of an indistinguishable number of settled point adders and multipliers is proposed. The settled point adders and multipliers are pipelined with the goal that increases, augmentations, and outside I/O operations are covered. Also, pipelining for the viper and multiplier brings about high clock speed.Amid a clock cycle, every multiplier understands one component from each of the vectors and duplicates these two drifting point numbers. A snake tree at that point entireties up the yields of the multipliers.

When k < n, an additional adder to accumulate the outputs of the adder tree is needed. Thus, k adders, including the k - 1 adders in the adder tree and the additional adder are used. If the clock cycles used to fill the pipelines of the multipliers and the adders are ignored, the effective latency of the architecture

$$T= n/k.$$

The adder tree in the architecture yields one output each clock cycle. Thus, the task of the additional adder is to reduce sets of sequentially delivered floating-point values. However, the pipelining in the floating-point adder can cause read-after-write hazards during the reduction. Therefore, the additional adder outside the adder tree using a reduction circuit is replaced . Tred(n/k) to used to denote the time required for reduction circuit.The design performs 2k+1

external I/O operations During each clock cycle, that is bw = 2k + 1. The number of I/O pins used is (2k+1)w.

MATRIX-VECTOR MULTIPLICATION (MVM)

`A floating-point matrix-vector multiplication system is designed and implemented in this project. The basic operation to be perform is AX=Y, where A is an arbitrary size of matrix, X and Y are vectors with size of matrix A's column number. An example of matrix-vector multiplication is shown in Equation 4.1. The matrix- vector multiplication involves both floating-point multiplication and addition, which is implemented onto the Pilchard platform in this project. The obtain the result vector Y, taking one row from matrix A at a time, multiply it with vector X element-by-element, hence accumulate the partial results to get the final result, Equation 4.2 shown the process of calculating Y.

OPERATION ANALYSIS

In matrix-vector multiplication, the total number of floating- point operations is $2n^2$. As in the case of dot product, we identify k and l as the design parameters.In this operation, each element in x can be reused n times. If any element of x in the architecture is not stored , the total number of external I/O operations is $2n^2$. To partially reduce the I/O costs, block matrix vector multiplication is used. Thus, a new parameter, b, needs to be introduced. Without loss of generality, n is taken as a multiple of b In the blocked version, A is partitioned into n/b blocks of columns, with each block of size n x b.Similarly, x is partitioned into n/b blocks of size b x 1.

The blocks are denoted using Ag and xg, respectively (g= 0; 1; . . . n/ b-1). For each block matrix-vector multiplication Ag x xg, Ag and xg need to be read from and the results need to be written to the external memory. Therefore, the total number of external I/O operations with block size b equalsSet k =l to reduce Tcomp. The above equation shows a design trade-off between the block size and the I/O time. The smaller the block size, the more external I/O operations are required and vice versa. Another trade-off exists between the storage size and the

data access time. If one copy of xg in the stored in the architecture only, reading k distinct values from the internal storage takes multiple clock cycles. On the other hand, if xg is duplicated at each multiplier, each multiplier can access the needed data in one clock cycle. In this case, the storage size is increased to k*b.

ARCHITECTURE

The matrix-vector multiplication is performed, incorporate with both software and hardware design. The hardware design on FPGA for this application is to perform one row multiply with one column at a time, the complete matrix-vector multiplication can be performed by iteratively apply one row and one column to the FPGA.

This architecture is almost the same as the architecture for dot product except that each multiplier is attached to a local storage.

In matrix-vector multiplication, the total number of floating- point operations is $2n^2$. As in the case of dot product, we identify k and l as the design parameters.In this operation, each element in x can be reused n times. If any element of x in the architecture is not stored , the total number of external I/O operations is $2n^2$. To partially reduce the I/O costs, block matrix vector multiplication is used. Thus, a new parameter, b, needs to be introduced. Without loss of generality, n is taken as a multiple of b In the blocked version, A is partitioned into n/b blocks of columns, with each block of size n x b.Similarly, x is partitioned into n/b blocks of size b x 1.

During each clock cycle, the k multipliers multiply k distinct elements of A with one element of x. The adders then accumulate the results of the multiplications with the intermediate results of y stored in the local storage. In this case, the intermediate result of yi (0, 1; . . . n-1) is updated every n clock cycles.

MATRIX MULTIPLICATION(MM)

OPERATION ANALYSIS

The total number of floating-point operations is $2n^3$. As each element of A and B is used n times, the total number of external I/O operations is $2n^2$. Because of the data reuse, the size of internal storage for intermediate results needed by the design becomes an important design parameter. I/O complexity to refer to the total number of external I/O operations performed by an algorithm is used. It has been proven that the I/O complexity of any implementations of the usual matrix multiplication algorithm is when , where m is the size of the internal storage.Again, set k=l to optimize Tcomp.

ARCHITECTURE

An architecture that employs both the on-chip memory and the onboard memory is proposed. The matrices A and B are partitioned into b2 x b2 blocks. These blocks are denoted as Aiq and Bqj. The local storage units of the PEs in MM are implemented using the on-chip memory of the FPGA. Thus, we have mb = 2b12. The architecture uses one storage unit of size b1 * b2 to store elements in Bqj. It also contains one storage unit of size to store the intermediate results Cij. These storage units are implemented using SRAM. The output of MM is accumulated with the stored intermediate results by the adder in the architecture.If the adder generates a final element of C, the element is output to the external memory; otherwise, the element is written back to the storage unit.

5. Simulation Results

In our design, the control logic occupies less than 5 percent of the total area. The clock speed of the design is limited by that of the fixed-point units, which is 170 MHz. When k increases from 1 to 6, the area of the design increases linearly. The latency of the design decreases proportionally as k increases. When the clock speed remains fixed, the required external memory bandwidth also increases linearly with k

MATLAB Object is normally placed on some background ,so to track the object foreground and background are separated as matrix using matrix algorithm .Except the object other pixels are converted to zero. So matrix multiplication of the foreground and background detects only the object in the two dimensional .Pre-blend imitations is the first of two plan confirmations in the outline procedure. In this stage, the useful reenactment of the outline is being tried to check the rationale in the plan carry on accurately. Since the outline has not been actualized on any gadget, timing data is inaccessible at this stage. To play out the practical reproduction, a test seat is connected to the outline to acquire the recreation waveform for signals in the plan.. This test seat is just utilized by the test system and furthermore blended. The test seat utilized as a part of this venture has an arrangement of 16 bit - speck item that spoken to in above.

Table 1 Device utilization summary

S.No	Parameters	Performance Without optimized		Performance With Optimized Settings	
1.	Number of Slices:	496 out of 3584	13%	334 out of 3584	9%
2.	Number of Slice Flip Flops:	690 out of 7168	9%	528 out of 7168	7%
3.	Number of 4 input LUTs:	536 out of 7168	7%	291 out of 7168	4%
4.	Number of IOs:	19	-	29	-

5.	Number of bonded IOBs:	19 out of 141	13%	-	-
6.	Number of GCLKs:	1 out of 8	12%	-	-
7.	Minimum period:	11.070ns	-	9.339ns	-

Table 2 Synthesis Report

Device Utilization for 2V40cs144			
Resource	**Used**	**Avail**	**Utilization**
IOs	40	88	45.45%
Function Generators	16	512	3.13%
CLB Slices	8	256	3.13%
Dffs or Latches	0	776	0.00%
Block RAMs	0	4	0.00%
Block Multipliers	4	4	100.00%

Clock Speed	80Mhz		

6. Conclusion

Improved FPGA-based usage for dab item, network vector increase and Matrix Multiplication are proposed and Various outline parameters were distinguished for each straight polynomial math operation and the subsequent plan exchange offs were broke down. The parameters incorporate the quantity of settled point units, the extent of the inside capacity of the plan, and the piece estimate when blocking calculations are utilized. Both the on-chip memory and the installed memory of the FPGA gadget are used. The execution comes about demonstrate that utilizing all the more skimming point units in a plan prompts littler idleness however requires more zone and higher memory transmission capacity. The piece estimate causes exchange offs between the span of inside capacity and the required outer memory transmission capacity for network vector increase. Be that as it may, the piece estimate has little impact on the inactivity of these two operations. In the majority of the proposed plans, the execution increments relatively with the accessible equipment assets. The execution of our plan contrasts positively and that of outlines in light of broadly useful processors. The superior outlines of DOT Product are executed in convolution channel and Matrix Multiplication in protest and limit following circuit and their exhibitions are broke down. The net yield comes about demonstrates that this technique is viable and execution is great. In future, it can likewise be connected to other picture handling applications.

Improved FPGA-based usage for dab item, network vector increase and Matrix Multiplication are proposed and Various outline parameters were distinguished for each straight polynomial math operation and the subsequent plan exchange offs were broke down. The parameters incorporate the quantity of settled point units, the extent of the inside capacity of the

plan, and the piece estimate when blocking calculations are utilized. Both the on-chip memory and the installed memory of the FPGA gadget are used.

Be that as it may, the piece estimate has little impact on the inactivity of these two operations. In the majority of the proposed plans, the execution increments relatively with the accessible equipment assets. The execution of our plan contrasts positively and that of outlines in light of broadly useful processors. The superior outlines of DOT Product are executed in convolution channel and Matrix Multiplication in protest and limit following circuit and their exhibitions are broke down. The net yield comes about demonstrates that this technique is viable and execution is great. In future, it can likewise be connected to other picture handling applications.

References

1. Storaasli O., Singleterry R.C. and Brown S. (2002), 'Scientific Computations on a NASA Reconfigurable Hypercomputer', Proc. Fifth Ann. Int'l Conf. Military and Aerospace Programmable Logic Devices, Sept. 2002.
2. Underwood K.D. and Hemmert K.S. (2004), 'Closing the Gap: CPU and FPGA Trends in Sustainable Floating-Point BLAS Performance', Proc. 12th Ann. IEEE Symp. Field-Programmable Custom Computing Machines, Apr. 2004.
3. Smith M., Vetter J. and. Liang X (2005), 'Accelerating Scientific Applications with the SRC-6 Reconfigurable Computer: Methodologies and Analysis', Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp., Apr. 2005.
4. Guo Z., Najjar W., Vahid F. and Vissers K. (2004), 'A Quantitative Analysis of the Speedup Factors of FPGAs over Processors', Proc.12th ACM/SIGDA Int'l Symp. Field Programmable Gate Arrays, Feb. 2004, pp. 162-170.

5. Aggarwal V., George A. and Slatton K. (2006), 'Reconfigurable Computing with Multiscale Data Fusion for Remote Sensing', Proc. 14th ACM/SIGDA Int'l Symp. Field Programmable Gate Arrays, Feb. 2006, p. 235.
6. Bajracharya S., Shu C., Gaj K. and El-Ghazawi T. (2004), 'Implementation of Elliptic Curve Cryptosystems over gfð2nÞ in Optimal Normal Basis on a Reconfigurable Computer', Proc. 12th ACM/SIGDA Int'l Symp. Field Programmable Gate Arrays, Feb. 2004.
7. Buell D.A. and Davis J.P. (2002), 'Reconfigurable Computing Applied to Problems in Communications Security', Proc. Fifth Ann. Int'l Conf. Military and Aerospace Programmable Logic Devices, Sept. 2002.
8. Koohi A., Bagherzadeh N. and Pan C. (2003), 'A Fast Parallel Reed-Solomon Decoder on a Reconfigurable Architecture', Proc. First IEEE/ACM/IFIP Int'l Conf. Hardware/Software Codesign and System Synthesis, Oct. 2003.
9. Bader D., Moret B. and Sanders P. (2002), 'High-Performance Algorithm Engineering for Parallel Computation', Lecture Notes in Computer Science, Vol. 2547, pp. 1-23..
10. Lawson C., Hanson R., Kincaid D. and Krogh F. (1979), 'Basic Linear Algebra Subprograms for FORTRAN Usage', ACM Trans. Math. Software, Vol. 5, No. 3, pp. 308-323.
11. Zhuo L. and Prasanna V.K. (2004), 'Scalable and Modular Algorithms for Floating- Point Matrix Multiplication on FPGAs', Proc. 18th Int'l Parallel and Distributed Processing Symp., Apr. 2004.
12. Smith M., Vetter J. and Alam S. (2005), 'Scientific Computing Beyond CPUs: FPGA Implementations of Common Scientific Kernels', Proc. Eighth Ann. Int'l Conf. Military and Aerospace Programmable Logic Devices, Sept. 2005.
13. Barrett R., Berry M., Chan T.F., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo R., Romine C. and der Vorst H.V. (1994), Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, second ed. SIAM, 1994.

14. Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T. (1992), Numerical Recipes in C: The Art of Scientific Computing. Cambridge Univ. Press, 1992. IEEE 754 Standard for Binary Floating-Point Arithmetic, IEEE, 1984. Whaley R.C., Petitet A. and Dongarra J.J. (2001), 'Automated Empirical Optimization of Software and the ATLAS Project', Parallel Computing, also available as Univ. of Tennessee LAPACK Working Note #147, UT-CS-00-448, 2000.