Implementation of Singular Value Decomposition Using High Level synthesis on FPGA

Singaravelan N, Lokesh D, Muthumanickam S, Arun C
Department of ECE, RMK College of Engineering and Technology, Puduvoyal

ABSTRACT

Singular value decomposition is a technique that can be used for a variety of operations such as image compression and watermarking. The important feature of the SVD is the invariance of singular values to common image processing operations and geometric transforms like rotation, translation and scaling. Due to this property, SVD has been used and also combined with other techniques for developing watermarking algorithms particularly resistant to geometric attacks. Though the applications are vast the implementation of the same in the FPGA is still a challenge. Recent tools such as Vivado HLS(High Level Synthesis) has enabled an easy solution to this problem, by which one can easily generate the codes in VHDL that can be executed, simulated and implemented successfully in Design suites. The IP core (Intellectual Property) for the generated code can also be created and used for other projects that need SVD.

Keywords: Field Programmable Gate Array(FPGA) , Hardware Descriptive Language (HDL) ,High Level Synthesis (HLS), Singular value decomposition (SVD).

I. INTRODUCTION

The SVD is a meticulously useful decomposition, useful many purposes. Consider some matrix A with rank three hundred; that is, the columns of this matrix span a 300-dimensional space. Encoding this matrix on a system is going to take a lot of memory. It is helpful in approximating this matrix with one of lower rank - how close can we get to this matrix if we only approximate it as a matrix with rank fifty, so that we only have to store a fifty columns.[9]This yields a quick compression algorithm for matrices. Since we only need to store the columns of lesser dimensional matrix actually get used, we greatly reduce the memory usage.

The singular value decomposition (SVD) is an incredibly efficient tool, and it is scattered throughout almost very scientific discipline[8]. For instance, it can be used for efficiently simulating higher order partial differential equations by taking all the data generated from the simulations, reducing the data dimensionality by throwing away some of the singular values, and then simulating the lower-dimensional system. The fact that SVD gives us an optimal low-rank representation guarantees that this sort of simulation preserves most of the detail in a system, as getting rid of the extra modes (singular values) in the system is guaranteed to get rid of the least important modes. in other cases, the SVD is used everywhere for dimensionality reduction; the algorithm commonly known as Principal Component Analysis (PCA)[4]which is explained by Raul V. Ramirez et.al.

High-level synthesis (HLS), referred to as C synthesis, electronic system-level (ESL) synthesis, algorithmic synthesis, or behavioral synthesis, is an automated design process that infers an algorithmic description of a desired behavior and creates digital hardware that implements that behavior[10]. Synthesis starts with a high-level specification of the problem, where performance is generally decoupled from e.g. clock-level timing. Early HLS explored a variety of input specification languages such as Fortran, ANSI etc .,although recent research and commercial applications accept synthesizable subsets of ANSI C/C++/SystemC/Matlab. The code is analyzed, architecturally constrained, and scheduled to create a register-transfer level (RTL) and VHDL or Verilog which are hardware description language (HDL), which is then in turn commonly synthesized to the gate level by the use of a logic synthesis tool. The goal of HLS is to let hardware designers efficiently build and verify hardware, by giving them better control over optimization of their design architecture, and through the nature of allowing the designer to describe the design at a higher level of abstraction while the tool does the RTL implementation. Verification of the RTL is ancrucial part of the process. Hardware design can be created at a different of levels of abstraction. The commonly used levels of abstraction are gate level, register-transfer level (RTL), and algorithmic level.

While logic synthesis employs an RTL description of the design, high-level synthesis works at a higher level of abstraction, starting with an algorithmic description in a high-level language such as SystemC and Ansi C/C++. The designer usually develops the module functionality and the interconnect protocol. The high-level synthesis tools handle the micro-architecture and transform untimed or partially timed functional code into fully timed RTL implementations, automatically creating cycle-by-cycle detail for hardware implementation. The (RTL) implementations are then used directly in a conventional logic synthesis flow to create a gate-level implementation.

II. RELATED WORK

Singular value decomposition have been one of the complex arithmetic computation with hardware implementation. Over a past decade it has been proposed different methodologies for implementing in hardware. Christophe Bobda et.al. [1] proposed a Floating-point implementation and Traditional method with rotational angle methodology to implement SVD.One side Jacobi Iteration has less memory space and simple. Pentium processor is used, but No solution for fixed-point operations to increase the design clock while decreasing the design area. X.Hu, S. Bass et.al. [7] defined Circular CORDIC which is simpler than rotational algorithms and it is presented only for 2x2 matrix. SwanirbharMajumder et.al.[6] gave detail explanation about various methods for computing SVD such as Jacobi SVD Algorithm, QR Decomposition, Hestenes Jacobi Method. They mentioned that hardware implementation is very much rare. P.A. Ramachandran [3] proposed two methodologies such as Boundary collocation and Boundary fitting. There is no hardware details or implementation results found. Shuge Yin et.al.[5] gave time and frequency domain localization features for wavelet ridge that can be calculated by SVD it has been aimed only at intra pulse width modulation but not on the implementation. Unai Martinez-Corral et.al. described Double dataflow Paradigm. Modified hestenes-jacobi algorithm for calculating SVD here the implementation platform has been in FPGA. Insection 3 we have discussed the flow for solving singular value decomposition, in section 4 we have explained about Vivado HLS, In section 5 the SVD module has been synthesized on HLS and in section 6 RTL Co simulation is discussed. In section 7 the experimental results have been presented, Implementation details are in section 8and Conclusion and Future work has been explained in section 9.

III. SINGULAR VALUE DECOMPOSITION

If a matrix A has a matrix of eigenvectors P that is not invertible, then A does not have an Eigen decomposition. However, if A is an m×n real matrix with m>n, then A can be written using a so-called singular value decomposition of the form

$$A = U\sum V^T$$

where U and V are respectively m x m and n x n orthogonal matrices, UTU = I, VTV = I. P is a diagonal matrix, P = diagonal (r1, r2,…,rr) with non negative singular values of A arranged in a descending order with m x n. There are many numerically stable methods for computing the SVD such as QR algorithm, Jacobi method and one sided Hestenes–Jacobi method. Let A be a general real (complex) matrix of order MxN. The singular value decomposition (SVD) of A is the factorization,

$$\begin{pmatrix} \\ \mathbf{A} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \mathbf{U} \\ \\ \end{pmatrix} \begin{pmatrix} s_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & s_n \end{pmatrix} \begin{pmatrix} \\ \mathbf{V} \\ \\ \end{pmatrix}^{\mathrm{T}}$$

where U and V are orthogonal(unitary) and S=diagonal($\lambda 1, \lambda 2, \ldots \lambda r$), where $\lambda i, I = 1{:}r$ are the singular values of matrix A with r=min(m,n) and satisfying

$$\lambda 1 \geq \lambda 2 \geq \lambda 3 \geq \cdots \geq \lambda r$$

The first r columns of V are the right singular vectors and the first r columns of U are the left singular vectors of A. To calculate the SVD of A we need to compute the Eigen values and Eigen vectors of AA^T or A^TA. The eigenvectors of AA^T from the columns of U, while the Eigen vectors of A^TA from the columns of V. The singular values in S are the square roots of the Eigen values of AA^T or A^TA. Each singular value specifies the luminance of the image layer while the corresponding pair of singular value specifies the geometry of the image layer. Another important feature of SVD is the invariance of singular values to common image processing operations and geometric transforms like rotation, translation and scaling. Due to this property, SVD has been used and also combined with other techniques for developing watermarking algorithms particularly resistant to geometric attacks[2].

IV. VIVADO HIGH LEVEL SYNTHESIS

Complicated algorithms used today in wireless, medical, defense, and consumer applications are more sophisticated than ever before. Vivado High-Level Synthesis included as a no cost upgrade in all VivadoHLx Editions, accelerates IP creation by enabling C, C++ and System C specifications to be directly targeted into Xilinx All Programmable devices without the need to manually create RTL. Supporting both the ISE® and Vivado design environments Vivado HLS provides system and design architects alike with a faster path to IP creation and also helps in Accelerated verification using C/C++ test bench simulation, automatic VHDL or Verilog simulation and test bench generation.[11] Vivado HLS supports older architectures specific to ISE Design Suite and installs automatically as part of the VivadoHLx Editions.

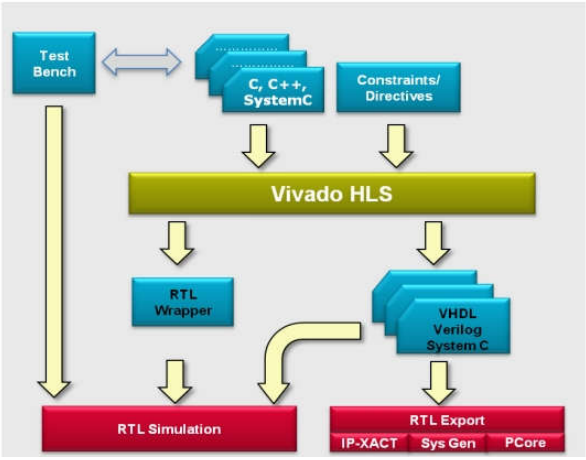

Figure 1. Flow of HLS

It can be seen that with the help of HLS it is possible for us to create an HDL code for any given C or C++ program which are basically a high level language.

V. SYNTHESIS OF SVD IN VIVADO HLS

Linear algebra module is used for implementing the Singular value decomposition which is pre-defined module in HLS and next the top module for the SVD is written where the matrices A S U and V and then again the matrix A is reconstructed are defined in terms of their dimensions. The C++ program with the top module and the function called by linear algebra is first debugged with the help of the inbuilt compiler and debugger. After debugging the synthesis for the high level program is done and also a test bench program also synthesized at the same time. The synthesis report for the Kintex 7 architecture board is taken by us.

VI. RTL CO-SIMULATION IN HLS

After the C synthesis HDL code is obtained by selection of RTL Co-Simulation here the desired language for the implementation is chosen and then the simulation starts. After the completion the implementation details are given by the HLS. Now the RTL analysis is done by the HLS and we can acquire the HDL code for SVD in the implementation folder thus this the desired project folder for our project in this case for Singular Value Decomposition. Next is to export the project, with the help of Export RTL selection it is possible to create an IP (Intellectual Property) for the whole SVD module.

VII. EXPERIMENT RESULTS

1. Test bench Results

In SVD the input is the matrix which has to be defined in the test bench for particular situation that is clock and power. Here in HLS the test bench has been created in C++ where the input matrix is given with rows and columns defined. We have taken a formal 6x6 matrix and got the desired results from the vivado HLS during its synthesis. The below figure gives the detailed summary of the RTL ports for the SVD module.

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	svd_top	return value
ap_rst	in	1	ap_ctrl_hs	svd_top	return value
ap_start	in	1	ap_ctrl_hs	svd_top	return value
ap_done	out	1	ap_ctrl_hs	svd_top	return value
ap_idle	out	1	ap_ctrl_hs	svd_top	return value
ap_ready	out	1	ap_ctrl_hs	svd_top	return value
A_address0	out	6	ap_memory	A	array
A_ce0	out	1	ap_memory	A	array
A_q0	in	32	ap_memory	A	array
S_address0	out	6	ap_memory	S	array
S_ce0	out	1	ap_memory	S	array
S_we0	out	1	ap_memory	S	array
S_d0	out	32	ap_memory	S	array
U_address0	out	6	ap_memory	U	array
U_ce0	out	1	ap_memory	U	array
U_we0	out	1	ap_memory	U	array
U_d0	out	32	ap_memory	U	array
V_address0	out	6	ap_memory	V	array
V_ce0	out	1	ap_memory	V	array
V_we0	out	1	ap_memory	V	array
V_d0	out	32	ap_memory	V	array

Figure 2. Port Configurations in SVD module

2.Synthesis details in HLS

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	96
FIFO	-	-	-	-
Instance	14	32	13396	13632
Memory	8	-	0	0
Multiplexer	-	-	-	80
Register	-	-	507	-
Total	22	32	13903	13808
Available	650	600	202800	101400
Utilization (%)	3	5	6	13

Detail

Figure 3. Utilization Estimates of Synthesized SVD in HLS

The logic synthesis is a procedure by which an abstract form of desired circuit behavior, typically at register transfer level (RTL), is turned into a design implementation in terms of logic gates. Here before implementing in Vivado Design suite the HLS provides information about the utilization measures. With respect to Kintex architecture totally 13 percent of the Look up table are utilized by this module and only 6percent of flip-flops are used. Block RAM is used for only about 3percent. These are only estimation about utilization that are presented in HLS.

3.Synthesis report on vivado

Utilization - Post-Synthesis

Resource	Estimation	Available	Utilization %
LUT	8938	101400	8.81
LUTRAM	594	35000	1.70
FF	11708	202800	5.77
BRAM	9	325	2.77
DSP	32	600	5.33
IO	157	285	55.09
BUFG	1	32	3.12

Figure 4. Utilization Estimates of Synthesized SVD in Vivado Design Suite

Now after synthesizing the SVD module in the Vivado Design suite we got our post synthesis utilization report in the below figure. Here the table gives information about the total components present and the components which have been utilized. From the table we can see that most IO are utilized which constitutes about 55%. And around 6% of flip-flops are been utilized by this module and both LUT and LUTRAM constitutes about total 11% of utilization. Only one Global buffer is used and 9 block RAM are used. Over 6% of Digital signal processing block is used by SVD module. The below figure represents the RTL schematic of the whole SVD module.

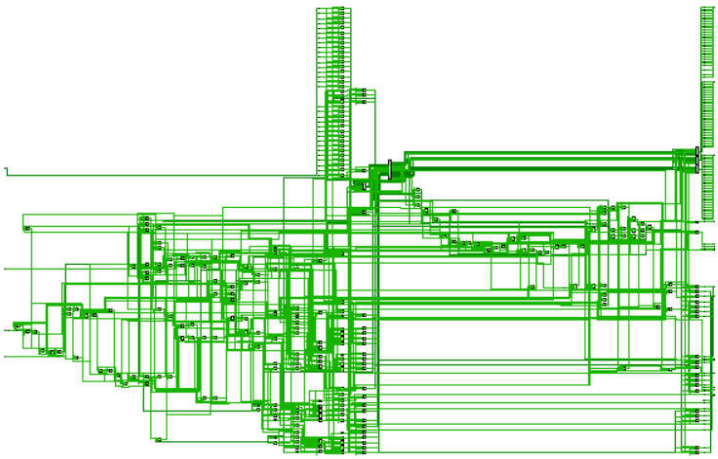

Figure 5. RTL schematic of SVD module

4.Power report

The total on chip power is of 0.751 Watts and the Junction temperature of the kintex 7 chip after implementing the SVD module is near 26.9 degree Celsius. Thermal margin is of 58.1 degrees and of 23 watts. On chip the dynamic power dominates over static power by constituting over 85% of power consumption. In dynamic power the signals have consumed about 39% which is 0.250 watts. And 14% for clock signals, 24% for the logic inside the chip. The static power of the device constitutes about 15% that is of 0.115watts.

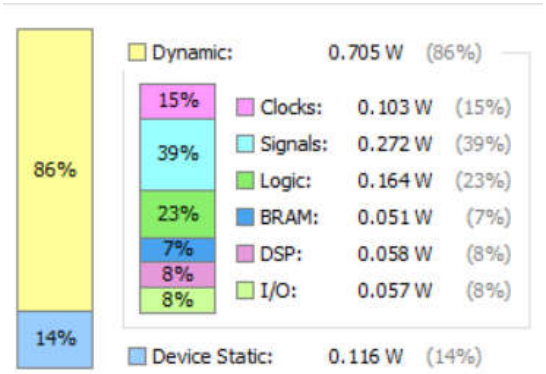

Figure 6. Power Report for Implemented SVD on FPGA

VIII. Implementation and Intellectual property

SVD module has been synthesized and implemented on Vivado design suite for Kintex 7 FPGA. Entire routing has been done. In its simplest form, a net list contains a list of the terminals ("pins or ports") of the electronic components in a circuit and a list of the electrical conductors that interconnect the terminals. A net is a conductor that interconnects two or more component terminals. It has been verified that there are no conflict nets or not routed or partially routed nets. All nets are fully routed. 18529 nets have been fully routed without any conflicts. The below figure represents the fully implemented design on the FPGA Kintex 7 board.

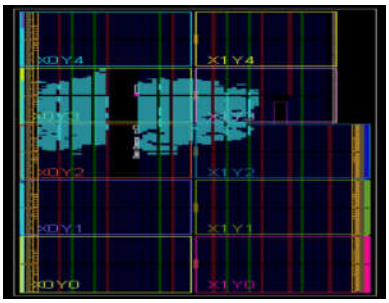

Figure 7. Implemented Design for SVD module on Kintex 7 FPGA

With the help of HLS an Intellectual property is also created, thus it can be coupled with other modules for other work which involves the computation of SVD.

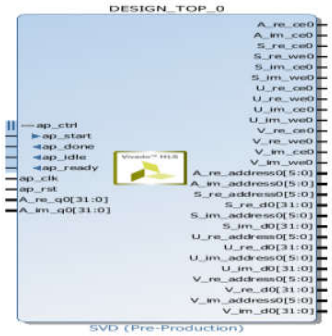

Figure 8. IP core for SVD module.

IX. CONCLUSION AND FUTURE WORK

We have implemented Singular Value Decomposition using High-level synthesis with the help of C++ program and then synthesized it on HLS. After that we have exported the RTL of SVD from HLS to Vivado and implemented the whole module with VHDL language on Kintex 7 architecture FPGA board, the implementation and synthesis reports have been presented.It has been seen that only 5.77% of the total Flip flops are utilized and static power has been minimized that is only 14% of the total power is consumed. An IP core for SVD module also been created which can be added with other modules where SVD is needed to be computed. We will be extending our work to couple SVD module with discrete wavelet transform to embed a watermark for multimedia content. This SVD module can also be used for Data compression also.

REFERENCES

[1] Christophe Bobda, Klaus Danne, and Andr´e Linarth 'Efficient Implementation of the Singular Value Decomposition on a Reconfigurable System' Springer-Verlag Berlin Heidelberg 2010.

[2] ManojKumar.V.R., S.Muthumanikcam , U.Manoranjan, C.Arun 'Non-Blind Watermarking of Color Images in DWTSVD Domain & Robustness to Various Attacks' International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:2682 Vol.35, No.35, March 2015

[3] P.A.Ramachandran,'Method of fundamental solutions: singular value decomposition analysis' COMMUNICATIONS IN NUMERICAL METHODS IN ENGINEERING 2002, (DOI: 10.1002/cnm.537)

[4] Raul V. Ramirez-Velarde, Martin Roderus, Carlos Barba-Jimenez, 'A Parallel Implementation of Singular Value Decomposition for Video-on-Demand Services Design Using Principal Component Analysis', Procedia Computer Science, Volume 29, 2014, Pages 1876-1887

[5] ShugeYina , Wei Liub, Chao Wang, 'Based on the singular value decomposition fast wavelet ridge intra pulse modulation recognition' Applied Mechanics and Materials Vols 556-562 (2014) pp 4933-4940)

[6] SwanirbharMajumder , Anil Kumar Shaw, Subir Kumar Sarkar,'Hardware Implementation of Singular Value Decomposition' 12 November 2014 J. Inst. Eng. India Ser. B DOI 10.1007/s40031-014-0158-0.

[7] Unai Martinez-Corral, KoldoBasterretxea, and Raul Finker, 'Scalable Parallel Architecture for Singular Value Decomposition of Large Matrices' IEEE 2014

[8] digilib.icpac.net/dochelp/StatTutorial/SVD/

[9] http://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition.

[10] https://en.wikipedia.org/wiki/High-level_synthesis

[11] https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html