High Speed Systolic Array Structure for Variable Block Size Motion Estimation

Vinod Reddy {nalamalapu@wisc.edu}

Overview

In this project , I aim to explore different systolic array organizations for computational intensive variable block size motion estimation of H264 Video standard. My aim is to come up with a efficient systolic array implementation which is *fully pipelined, highly utilized, highly parallel, reduced data I/O bandwidth* for the VBSME.I plan to do a VLSI implementation with the modules coded in verilog and will try to synthesize to a clock frequency which targets real time video encoding of qcif(176x144) frames with 15 frames per second. I intend to optimize the modules interms of speed and area.

Motivation

H.264/AVC[1] latest Variable block size motion estimation is definitely highly computation intensive and requires hardware acceleration. Motion estimation on different block sizes 4x4,8x4,4x8,8x8,16x8,8x16,16x16 results in better video compression but at the same time increases the number of computations. Software profiling shows integer pel motion estimation requires 95k MIPS and is 78% of computing in the overall H264 Encoding [2]. Hence the software implementation falls behind the real time performance requirements and also techniques using specialized instructions like MMX/SSE are impractical for real time performance. In literature many VLSI architectures are presented for accelerating the H264 Variable block size estimation. One such successful architecture is the partial propogate SAD Tree architecture[3,4]. I will use this as a baseline architecture for my implementation.

High- Level Optimizations

The sample matlab code for VBSME is as follows

```
// This loop outputs a refblk from the search window for each loop iteration
for m = 1:Sw
        for n = 1:Sw
            sads_41(m,n)= compute_sads(CurrMB,SW(m:m+15,n:n+15));
        end
   end

// This loop cal 16 sad4x4 for a given Ref Blk and Current Blk(16x16)
for i = 1:4:13
  for j = 1:4:13      33
        sads4x4(k) = ∑∑ |CurrBlk(i,j)-RefBlk(i,j)|;
                     ij
        k = k + 1;
  end
 end
```

Loop Level Parallelism

- As we can see both the nested loops can be highly parallelized for higher throughput as the next iteration is independent of the previous iteration.
- We can process all the 256 ref blocks (in a 32x32 search window) in parallel.
- We can calculate all the sixteen 4x4 sads in parallel for a ref and curr blk.
- *But the limitations of this heavily parallel architecture is hugh area overhead and Buswidth (I/O) required to read the 256 reference block pixels.*

  Hence I intend to design a systolic array which has

  - moderate I/O Bandwidth (Bus Widths present FPGA can support) can be achieved by exploiting pixel sharing among different ref blocks.
  - Fully Pipelined systolic array which generates "41 SADS" every cycle for a ref blk and curr blk every cycle using sad propogate method [3].
  - Sad4x4 reuse to calculate the sads of 4x8,8x4,8x8,8x16,16x8,16x16 SAD's.

Low Level Optimizations

- optimizing structures for calculating difference of pixels and absolute function.
- Efficient adder trees for calculating the sum of all the differences.
- To combine the adder trees or retime them to reduce the critical path delay to achieve the desired clock frequency for a 4x4 SAD block.

Implementation Methodology

Part I

- Nested Loop analysis for parallelism
- Dependence graph analysis for a 4x4 sad block.
- Dependence graph for the analysis of partial sad usage between different reference blocks.
- Mapping the dependence graphs to a fully pipelined systolic array.
- Enquire for utilization efficiency of the systolic array.
- Analyze the BW(I/O) requirement for the modeled systolic array.

Part II

- Code the modules in verilog and check for functional correctness with the matlab soft implementation.
- Synthesize the design for real time performance requirements.
- Do the critical path analysis and balance the design by retiming if required.
- If time permits I intend to do the performance comparision of software and hardware implementation.

Tools

Language: Verilog,C,Matlab

Tools: Modelsim for verilog simulation.
Matlab,C for software implementation of Motion Estimation.
Design Compiler from synopsys for clock frequency synthesis and area estimation.

Deliverables

Final Project Report contains

- Detailed architecture implementation of SAD calculation, Absolute function etc.
- Timing and Area results of the implemented design
- Verilog code for the modules.
- Software Implementation matlab code.
- Performance results on software platform, If any.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, *Overview of the H.264/AVC Video Coding Standard*, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 560-576, July 2003.

[2] T.-C. Chen, S.-Y. Chien, Y.-W. Huang, C.-H. Tsai, C.-Y. Chen, T.-W. Chen, and L.-G. Chen, "Analysis and architecture design of an HDTV720p 30 frames/s H.264/AVC encoder," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 6, pp. 673–688, Jun. 2006.

[3] Chen Ching-Yeh ; Chien Shao-Yi ; Huang Yu-Wen ; Chen Tung-Chien ; Wang Tu-Chih ; Chen Liang-Gee, "Analysis and Architecture Design of Variable Block Size Motion Estimation for H.264/AVC," IEEE Transactions on Circuits and Systems I, Volume PP, Issue 99, 2005

[4] Zhenyu Liu, Yiqing Huang, Yang Song, Satoshi Goto, Takeshi Ikenaga, "Hardware-Efficient Propagate Partial SAD Architecture for Variable Block Size Motion Estimation in H.264/AVC," *Proceedings of the 17th Great Lakes Symposium on VLSI*, pp. 160-163, 2007