Guiding Power/Quality Exploration for Communication-Intense Stream Processing

Hamed Tabkhi, Majid Sabbagh and Gunar Schirner
Department of Electrical and Computer Engineering
Northeastern University, Boston (MA), USA
{tabkhi,msabbagh,schirner}@ece.neu.edu

ABSTRACT

In this paper, we explore the power/quality trade-off for streaming applications with a shift from the computation to the communication aspects of the design. The paper proposes a systematic exploration methodology to formulate and traverse power/quality trade-off for the class of adaptive streaming applications. The formalization enables to procedurally transition from a set of design requirements to architecture goals. The architecture goals can then be realized through design choices yielding system designs that meet the initial requirements. The reported results are based on an actual implementation of Mixture of Gaussian (MoG) background subtraction on Xilinx Zynq platform.

Keywords

Streaming Applications; Operational Data; Power/Quality Trade-off

1. INTRODUCTION

Streaming applications make up a large portion of the embedded high-performance market. Examples include multimedia and vision computing, software defined radio, radars and cryptography [3][7]. For the high-performance power-efficient execution of streaming applications, the general trend is to move the computation-intense kernel into hardware accelerators (either on FPGAs, or as an ASIC), and more intelligent/control processing into software. While the computation aspect of accelerators has been fairly optimized, the communication aspect has been left as a challenge; particularly the memory wall is an increasing hurtle. This shifts attention toward communication-centric design principles.

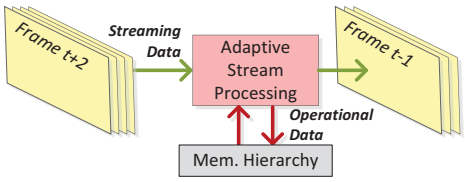


Figure 1: Streaming and operational data in adaptive vision kernels

The communication/memory access overhead is more pronounced in streaming applications; in particular, adaptive stream processing [4]. A good class of applications are vision algorithms. Fig. 1 outlines a typical data access flow for adaptive vision algorithms with a separation between streaming data and operational data paths. Streaming data is data under processing (pixels in the case of vision) and is typically read from input ports and written to output ports (each bypassing the memory hierarchy) or system memory. Conversely, operational data is data used for realizing the computation (e.g. kernel density histogram or Gaussian parameters).

The operational data is accessed and updated per individual frame and directly hits system memory. The volume of operational data often exceeds any on-chip cache, thus driving up the system power consumption. As an example, Mixture of Gaussian (MoG) background subtraction [4] uses about 8GB of memory access while processing 128 MPixels/s. This imposes significant bandwidth limitations as well as significant power overhead. Our power analysis demonstrates that for the case of MoG around 90% of overall power is consumed by operational data access.

Much research focuses on analyzing the power / quality trade-off in the context of video coding [2][1]. However, the trade-offs mainly focus on adjusting computational complexity (as a matter of power) and the quality of video compression. New generation of algorithms (e.g. MoG, KLT, optical flow) have an adaptive nature with lots of data interaction between frames and thus far more demands for operational data. These algorithms are still implemented at a much lower resolution (300*200) [7] while market demand is already there for HD resolution.

In this paper, we focus on power/quality trade-off with respect to communication aspects of the design. We primarily zoom into the operational data access demand for adaptoive streaming applications. To simplify future designs and for exploration, we use our previously introduced communication-centric architecture template [5]. The template offers configurable knobs for exploring the quality/power trade-off. Through a methodological exploration, we formulate the trade-off and propose a tool to automate the process. The reported power numbers are based on an actual implementation of our proposed architecture template on Xilinx Zynq platform [6] with a running MoG kernel. We quantify our result for Mixture of Gaussian (MoG) background subtraction [4].

The remainder of this paper is organized as follows. Section 2 further explains the background and motivation of our work. Section 3 describes our approach and formulation in detail. Section 4 presents the experimental results on power/ quality exploration. Finally, Section 5 concludes the paper and touches on future work.

2. BACKGROUND

In this paper, we selected Mixture of Gaussian (MoG) [4] to explore and quantify our trade-off. Fig. 2 includes the MoG coarse-grain mathematical formulation. MoG uses multiple Gaussian distributions, to model a pixel's background. Each Gaussian has its own

GLSVLSI '16, May 18-20, 2016, Boston, MA, USA

DOI: http://dx.doi.org/10.1145/2902961.2903004

set of parameters: weight, intensity mean and standard deviation. With a new pixel coming in, all Gaussian parameters are updated at frame basis to track BG changes of the pixel. For an individual pixel, while the streaming traffic is only 17-bits (16-bit for input pixel and 1-bit for foreground mask), the operational traffic hit on the memory access is 960-bits combining both memory read and write-back access (assuming 32 bit per Gaussian parameter with 5 Gaussian distributions).

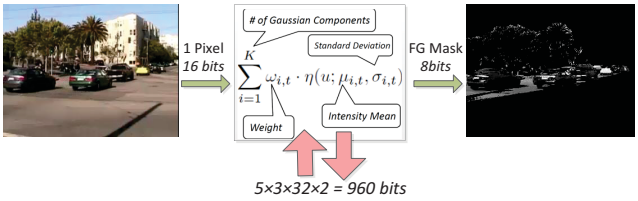
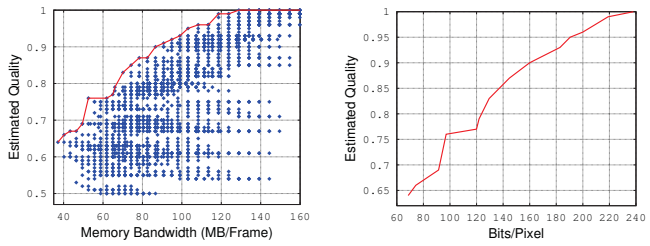


Figure 2: Memory access per pixel in MoG algorithm

To reduce the memory bandwidth for accessing operational data, we proposed adding a support block to compress / de-compress parameters in data memory read/write access [5]. The compression module is interfacing the running kernel and system memory and is configurable to compress/de-compress operational data at different bit width, while the internal processing is still being performed at the highest accuracy (e.g. 32 bits). Fig. 3a demonstrates the result of quality exploration as a trade-off between bandwidth per frame (1080*1920 resolution) on the x-axis and quality on y-axis. Fig. 3b presents the maximum quality over the increasing size of bits per pixel, derived from Pareto curve. We define bit per pixel as an indicator of quality metric. As an example, for 70-bits per pixel the quality would be only 0.64, or 244-bits per pixel for the maximum possible quality.



(a) Quality / Bandwidth trade-off (b) Quality over bits per Pixel

Figure 3: Quality exploration for 1080*1920 resolution

We also use our previously introduced communication-centric architecture template [5] (highlighted in Fig. 4). It consists of two clock domains: the computation clock domain driven by streaming data (pixels), and the communication clock domain set by the bus/interconnect frequency. The two clock domains are connected via *Async. FIFOs*. The compression/de-compression blocks provides run-time mechanism to reduce the bandwidth demand for operational data. Our architecture template offers a set of configurable knobs (design choices) including operational bits per pixel (as a quality indicator), Async. FIFO depth, DMA inline buffers as well as DMA channels and communication bus/interconnect width and frequency.

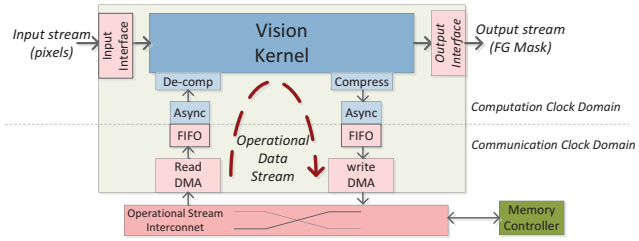


Figure 4: Architecture template for adaptive streaming kernels

3. APPROACH

In this section, we deploy a procedural method to traverse the axes of a design space. We consider the axes of exploration to be (a) requirements (e.g. quality), (b) architecture goals (e.g. operational bit width per pixel) (c) design options (e.g. bus bandwidth) and (d) metrics (e.g. power). After defining the axes of exploration, we need a procedural flow for step-by-step exploration and formulating of the problem.

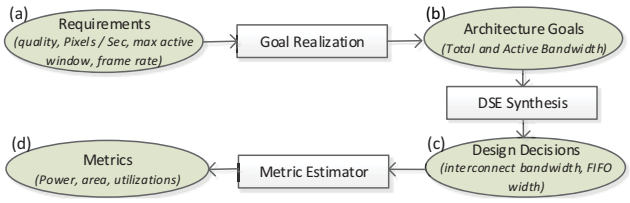


Figure 5: Requirement, design, metric exploration flow

Fig. 5 outlines the major steps of exploration. In a general view, the flow proposes three step exploration: (1) from requirements to architecture goals ((a)->(b)), (2) from architecture goals to design decisions ((b)->(c)), (3) from design decisions to metrics ((c)->)(d)). The goal is transferring the requirements to realizable architecture goals understandable in terms of architecture.

For MoG, $Quality$, $Frame\_Rate$, $Frame\_resolution$ and $Active\_resolution$ define the requirement set (highlighted by (1)). For simplicity, we did not define the concept of an active window until now. $Active\_resolution$ defines the window of interest under actual MoG processing. Although video can be processed full-frame, many developers in practice will focus on a part of the frame rather than the whole. The active window is a subset of original frame resolution representing the pixels under MoG processing. For high definition resolution (1080*1920), not all pixels are part of active window, but instead only a subset (window of interest) is processed at each period of time, which we call active window. Active window location and resolution can be changed over time.

$$Requirements = \{Quality, Frame\_Rate, Frame\_resolution, Active\_resolution\} \quad (1)$$

Similarly (2) defines the architecture goals as $Bits_{Pixel}$, BW_{active}, BW_{active_width}. $Bits_{Pixel}$ defines the bits per pixel as a quality indicator, BW_{active} defines the overall active bandwidth required for processing the active window of a frame. It can be derived from the requirements (highlighted in (2)). BW_{active_width} also demonstrates the bandwidth requirement for processing one row of the active_window (highlighted in (4)). BW_{active_width} is an architecture goal to make sure that the operational data are available during the processing of each active_window row.

$$Architecture\_Goal = \{Bits_{Pixel}, BW_{active}, BW_{active\_width}\} \quad (2)$$

$$BW_{active} = Frame\_Rate \times Width_{active} \times Height_{active} \times Bits_{pixel} \quad (3)$$

$$BW_{active\_width} = Width_{active} \times Bits_{Pixel} \quad (4)$$

As we highlighted in Fig. 5, after defining the requirements and deriving the architecture goals, it is time for DSE synthesis. The goal of DSE synthesis is identifying a set of design decisions that can satisfy the architecture goals. With respect to our architecture template, the design options are f_{Bus} (bus frequency), W_{Bus} (bus width)as

well as Async. FIFOs depth D_{FIFO}. The architecture knobs should be set in such a way as to satisfy the architecture goals. For that, two pre-conditions have to be verified: (a) total active bandwidth (BW_{active}) validation and (b) BW_{active_width} validation. The first step is validating that the communication bandwidth can supply the volume of bits demanded by the quality requirement. (5) describes the sustainable bus bandwidth. U_{bus} is bus utilization. The bus utilization is an indicator of how many bus cycles are transferring actual data. The architecture goal is satisfied only if the condition $(BW_{bus} > BW_{active})$ is true.

$$BW_{bus} = f_{bus} \times Width_{bus} \times U_{bus} \tag{5}$$

To meet condition B(BW_{active_width}), the communication system has to offer enough operational data per active row. (6) formulates the volume of data that can be supported by the bus per active row executed in computation domain which we name $\frac{f_{Bus}}{f_{comp}}$. It is an indicator of clock mismatch between the computation and communication clock domains; f_{Bus} represents communication clock frequency and f_{comp} is computation clock frequency. In practice, f_{Bus} appears to be lower than f_{comp}. Due to this the communication sub-system should guarantee that it can provide enough data per active row execution; otherwise, the application kernel is starved of data and does not operate correctly.

$$BW_{bus\_active} = width_{frame} \times width_{Bus} \times Util_{Bus} \times \frac{f_{Bus}}{f_{comp}} \tag{6}$$

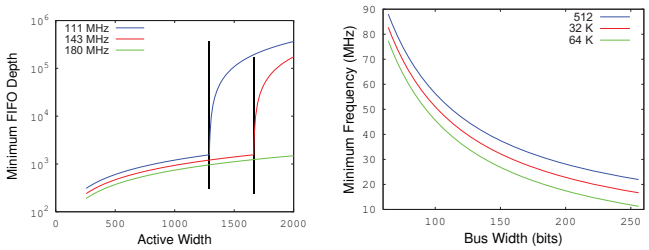
The Async. FIFOs are utilized for both bridging the two clock frequencies as well as compensating for the shortcoming of bus in supplying operational data. Satisfying $(BW_{Bus\_width} > BW_{active\_width})$ means that communication bandwidth can provide enough pixels per active row, resulting in smaller FIFO depth. In this case, FIFOs are just utilized for bridging between two clock domain. (7) formulates the FIFO size for condition of $(BW_{comm\_width} > BW_{active\_width})$. As $Width_{active}$ increases, a larger FIFO is required. Conversely, by increasing bus clock frequency f_{Bus} lower FIFO size is required.

$$if(BW_{Bus\_width} > BW_{active\_width}) \quad D_{FIFO} = Width_{active} \times U_{comm} \times \frac{f_{comp}}{f_{Bus}} \tag{7}$$

If the bus cannot satisfy $(BW_{comm\_width} > BW_{active\_width})$,a much larger FIFO would be required. In this case, FIFOs significantly contribue to compensating for the data supply shortcoming. Computation is halted while out of the active window so the communication bus can keep fetching data and fill the Async FIFOs to make sure that enough data are available during active window processing. (8) formulates the FIFO depth for these conditions. In this case, the maximum size of the active window $Height_{active}$ also contributes to the FIFO size; larger compensation is required maximum window size increases.

$$if(BW_{comm\_width} < BW_{active\_width}) \quad Depth_{FIFO} = \frac{(BW_{active\_width} - BW_{Bus\_width})}{Width_{bus}} \times Height_{active} \tag{8}$$

Note that, the purpose of (7) and (8) is not just for identifying the FIFO sizes. These equations can be used determine the minimum bus frequency or bus width for a predetermined FIFO sizes as well. Fig. 6b demonstrates the minimum required bus frequency over different bus widths with different FIFO sizes. In Fig. 6b, we assume a fixed active_frame size of 512*512 pixels with a fixed quality of 128-bits per pixel. Overall, what we observe is that the larger depth of FIFO size allows lower frequency and can potentially result in lower power usage. Furthermore, there is a trade-off between bus width and frequency while delivering identical bandwidth. The architecture knob sizing of Fig. 6b allows identical bandwidth but potentially different power costs. The architecture design knobs can be tweaked to achieve minimum power usage while maintaining a given quality requirement.



(a) FIFO depth over active width (b) Bus frequency over bus width

Figure 6: Design dimensions

We have now defined the design space exploration and narrowed down the exploration space to only design choices can satisfy the architecture goals. Based on our architecture template, each design choice can be described by (9) including f_{Bus}, $Width_{Bus}$ and $Depth_{FIFO}$.

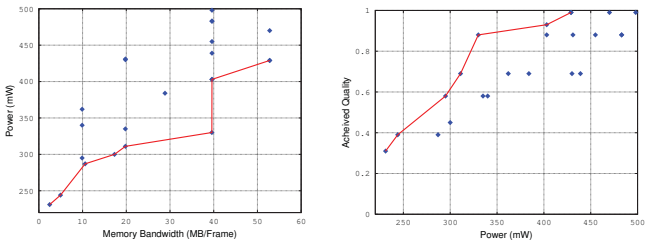
$$Choice_i = (Depth_{FIFO}, freq_{Bus}, Width_{Bus}) \tag{9}$$

4. RESULTS

To quantify the trade-off, we have prototyped our proposed architecture template on the ZedBoard with Zynq. Overall, the system power can be divided into off-chip and on-chip portions. Off-chip memory accesses for operational data impose a significant power overhead to the system (around 65% of entire system power in our implementation). By parameter compression and thus cutting access volume in half, the off-chip power halved as well. After optimizing off-chip accesses, on-chip communication elements need to be explored further for optimum power point per quality. We have analyzed the on-chip power by using Xilinx X-Power Analyzer tool after post synthesis and mapping of the entire design on Zynq platform.

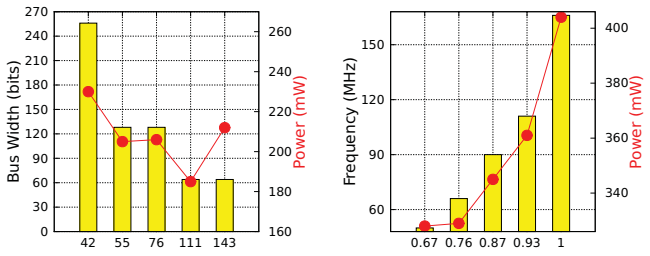
Fig. 7a describes the quality / bandwidth trade-off. With respect to formulation, we configure and synthesize our architecture templates to reflect the targeted bandwidth. Clearly, power demand increases with increasing communication bandwidth. However, we observe identical bandwidths with yet different power consumptions. This is due to the fact that different template configurations can result in identical bandwidth but differing power. Fig. 7b illustrates the trade-off between power and quality. Again based on the formulation, we calculated the design choices that can satisfy the quality point and change the architecture template to reflect this decision. We observe that some design choices lead to a lower power consumption while achieving same quality.

Fig. 8a demonstrates the bus width and power over increasing bus frequency. All these configurations can provide exactly enough bandwidth to satisfy the requirements for quality point of 0.8 (mean-



(a) Power over memory bandwidth (b) Achieved quality over power

Figure 7: Power / Quality exploration



(a) Frequency (b) Quality

Figure 8: Power / Quality trade-off

ing 128 operational bit width per pixel). Different configurations lead to different power consumptions. For example, the configuration with a 64-bit wide 111 MHz bus, has a lower power consumption than a 128-bit bus with 76 MHz. However, these two differ in quality (bit width per pixel). Fig. 8b presents the frequency and power for different quality points for a fixed bus width of 128-bits. Clearly, higher quality means larger bit-width per pixel and results in higher power demand.

Defining the step-by-step exploration (highlighted in Fig. 5) and formulation of problem for case of MoG set the stage for automating the exploration. Our proposed Design Space Synthesizer (DSS) takes as an input the application requirements (quality, active resolution) and converts them into architecture goals. Then, it identifies design candidates that can satisfy architecture goals based on the formulations. We implemented a prototype in Matlab, realizing the exact formulations.

The proposed DSS is a significant step toward reducing the exploration space complexity. The output of DSS is a set of design decisions which satisfy the architecture goals. Fig. 9 highlights all 150 considered design candidates for realizing MoG on the Zynq platform. Out of these design candidates, only four satisfy the requirement for a quality of 0.99, which leads to 244-bit operational data per pixel. The identified designs points satisfying the constraints are highlighted in red.

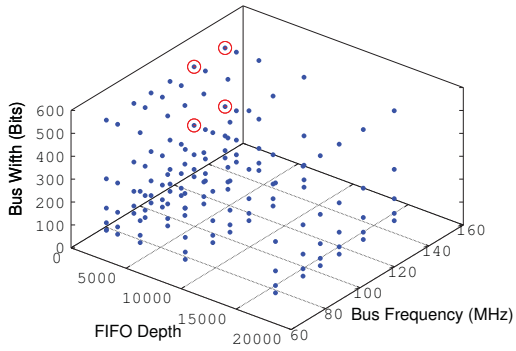


Figure 9: Identified design choices by the DSE tool

After identifying valid design configurations, their design metrics have to estimated (e.g. power, area ...) in order to achieve an optimization goal. In our example, the goal is to reduce power. Obtaining design metrics can be achieved by either an actual implementation (which is too time consuming), or by a metric estimation tools.

Many complex power estimators exist. For the purpose of initial estimation, we focus on first order rapid estimation that provide some boundaries for decision making. For this we develop a linear model. Our linear model is calibrated with a measured base power consumption and with incremental costs for each design choice dimension. For example, we assume a linear increase in power with FIFO size. We also assume linear relation with bus width. An increase in bus frequency has a multiplicative effect. The power model is driven by the actual sample points. The developed model, allows interpolation and estimation of new combinations.

Fig. 10 plots quality over power consumption for both estimated and real powers. The tool-estimated numbers are optimized for powers per quality point, while the actual powers (highlighted by red stars) are varies depending on their design decisions. The colors define the optimal design decisions (FIFO depth, bus frequency, bus width) made by tool for minimizing power consumption per quality. We observe that the estimated tool power numbers are very close to the actual powers (10 mW on average). Furthermore, the identified design decisions guide the user to the optimum power per quality point without need to exhaustive experimentation. Even the proposed first order estimation driven by the proposed DSS is a utility to quickly identify and evaluate design choices. We see a tremendous research potential in elaborating and refining automation of design space exploration that takes architecture and application knowledge into account.

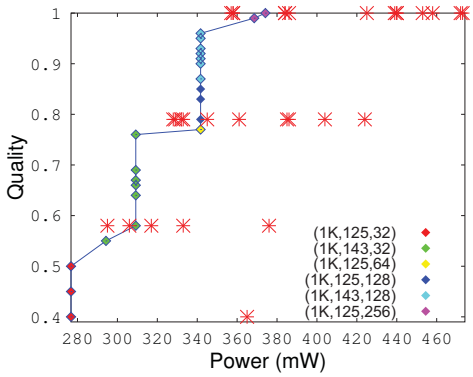


Figure 10: Quality over power for estimated and real execution

5. CONCLUSIONS

This paper explores the power/quality trade-off for communication-intense designs with a focus communication design aspect. The paper proposes a systematic exploration, which follows a procedural model to resolve design decisions in a step-by-step manner. We offer a formalization the exploration challenges for a case study of mixture of Gaussian (MoG) background subtraction. Our trade-off analysis, architecture template, and formalization example guide system designs realizing high-performance communication-intense streaming applications.

6. REFERENCES

[1] G. Chen, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and W. Wolf. Energy savings through compression in embedded java environments. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign*, CODES '02, pages 163–168, 2002.

[2] T. Lv, I. Ozer, S. Chakradhar, J. Xu, W. Wolf, and J. Henkel. A methodology for architectural design of multimedia multiprocessor socs. *Design Test of Computers, IEEE*, 22(1):18–26, 2005.

[3] H. Nikolov, M. Thompson, T. Stefanov, A. Pimentel, S. Polstra, R. Bose, C. Zissulescu, and E. Deprettere. Daedalus: toward composable multimedia mp-soc design. In *Proceedings of the 45th annual Design Automation Conference (DAC)*, pages 574–579, 2008.

[4] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –252 Vol. 2, 1999.

[5] H. Tabkhi, M. Sabbagh, and G. Schirner. Power-efficient real-time solution for adaptive vision algorithms. *IET Computers Digital Techniques*, 9(1):16–26, 2015.

[6] Xilinx. Zynq-7000 all programmable soc. In *Technical Reference Manual*, 2013.

[7] J. Xu, W. Wolf, J. Henkel, S. Chakradhar, and T. Lv. A case study in networks-on-chip design for embedded video. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 770–775 Vol.2, 2004.