Adaptive System on a Chip (aSoC) for Low-Power Signal Processing

Andrew Laffely, Jian Liang, Prashant Jain, Ning Weng, Wayne Burleson, Russell Tessier
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA. 01003
{alaffely, jliang, pjain, nweng, burleson, tessier}@ecs.umass.edu

Abstract

A*daptive* **S***ystem-***o***n-a-***C***hip, aSoC, is an on-chip communications architecture designed to promote scalability and flexibility in system-on-a-chip designs. This paper describes the use of aSoC to allow for dynamic power management of video signal processing systems. Content variation and perceptual tolerance in video signals can be exploited to gracefully trade quality for low power. The regularity and flexibility of the aSoC architecture provides a predictable framework for modeling the power, speed and area requirements of global communications that dominate configurable video processing.*

1 Introduction

Wireless video processing systems need to perform computationally intensive tasks with high throughputs while meeting stringent power constraints. Fortunately, VLSI technology, following Moore's law, has enabled completely integrated video systems [12]. While these early systems contain basic capabilities, more complex systems are required to fully implement the complexity and flexibility of present day video encoding standards [8].

Core based system-on-a-chip (SoC) implementations attempt to improve performance by taking advantage of application parallelism. They also manage complexity by enabling reuse of intellectual property (IP) cores. Performance improvement and reduction in power consumption depends on the following factors. First, breaking up the computational component introduces interconnect into the critical path. Second, algorithm partitioning may create hot spot bottlenecks, which limit performance. Coupled together, these two inefficiencies may introduce additional power consumption.

ASoC answers these concerns by using a novel, statically scheduled interconnect structure, which increases system throughput by up to 3.5 times that of bus-based systems [6]. Regularity of the interconnect structure and the predictability of static scheduling can be used to eliminate unnecessary interconnect switching activities. In addition, the aSoC architecture comes complete with effective application mapping tools and methodologies, which attempt to assure performance by balancing the computational load across all the cores. An asynchronous core interface allows for independent core operating frequencies thus mitigating the negative impacts due to unbalanced core loading. Finally, the core interface and cores themselves can be dynamically reconfigured to low power modes in response to data content variations and system operating environments.

This paper uses a partial video encoding system to demonstrate aSoC's dynamic power management capabilities. First, it shows how the aSoC interconnect supports the flexibility of dynamically reconfigurable cores. Specifically, aSoC is used to configure a flexible motion estimation core by selecting low power modes capable of reducing power by a factor of nearly 40. Second, dynamic reconfiguration of the interconnect and interface are shown to complement core power savings efforts by balancing core computation times and by turning off idle cores. It is also shown that dynamic configuration adds less than 10% overhead to the interconnect bandwidth and power budgets.

This paper proceeds as follows. Section 2 presents a brief overview of the aSoC architecture. Section 3 presents some VLSI power management techniques used in aSoC. The experimental approach is described in Section 4 and Section 5 shows how the aSoC architecture and reconfigurability support effective dynamic power management with low control overhead. Section 6 concludes the paper and suggests future work.

2 Scheduled Communication Architecture

Our approach to SoC integration is a tiled architecture that addresses both scalability and flexibility [6]. As shown in Figure 1, each tile represents a computational core and its interface to the network. The core interface supports the use of heterogeneous processing cores occupying one or more tiles. The system connects

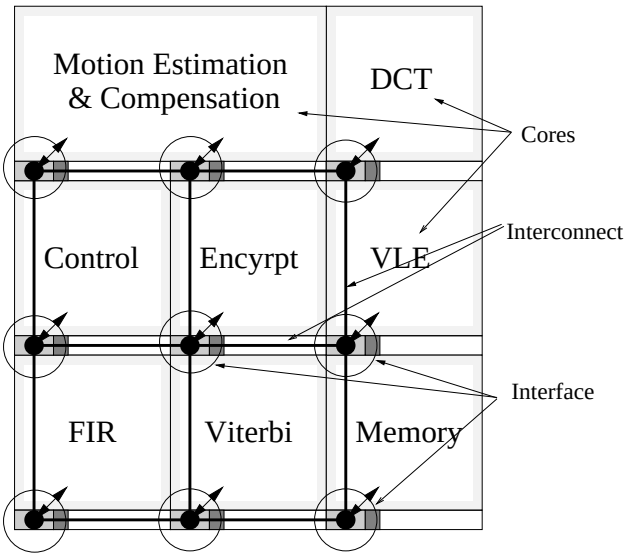

Figure 1: Tiled Architecture

these tiles using a statically scheduled mesh of interconnect, which assures predictable inter-core communications. Data moves between neighboring tiles in a communication pipeline, thus enabling fast clock rates and time sharing of interconnect resources. The ability to reconfigure both the cores and interconnect at run-time leads to the possibility of dynamic power management.

The core interface manages communications through each tile and synchronizes global communications. As shown in Figure 2, the instruction memory holds a list of the communication patterns required at run-time. The PC fetches these patterns in succession and a decoder converts them into switch settings for a crossbar. The crossbar routes data between the local core and the neighboring tiles (North, East South or West). Each incoming data word can contain local interface configuration information to be sent over the *local config.* line to the controller. The *Coreports* in Figure 2 use a simple protocol to interface communications between the potentially different clock domains of the core and interconnect. Multiple input and output *Coreports* can be used depending on the core and application requirements. During normal operations, the controller simply loops through a set of communication patterns, called a schedule.

Figure 3 shows the complete data transfer, or stream, between the cores of tile A and tile C built by the coordination of independent schedules in tile instruction memories. During cycle 1, tile 1 sends data from an output *Coreports* to the *East*. During cycle 2, tile B sets its own crossbar to transfer the data from its *West* input to tile C in the *East*. Finally during

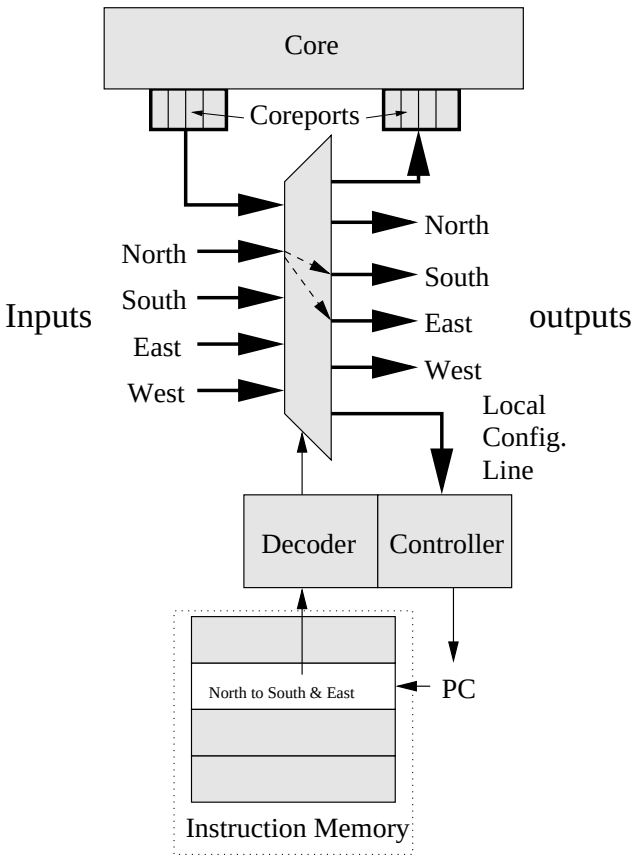

Figure 2: Core and Communication Interface

cycle 3, tile C moves the data from the *West* input to its own *Coreports*.

3 Dynamic Power Management for aSoC

Dynamic power management exploits run-time variations in data content and operational requirements to minimize one or more of the terms in the VLSI power equation [1], shown in Equation 1. At present aSoC supports and performs reduction of effective capacitance(C_{eff}) and frequency (f) with future goals of developing effective voltage (V_{dd}) scaling procedures [3][11] and associated hardware [2].

$$P_{ave} = C_{eff} \times V_{dd}^2 \times f \quad (1)$$

In some logic systems, C_{eff} may be reduced by eliminating excess switching activity [1]. For example, the discrete cosine transform (DCT) implementation discussed in [13] bypasses calculations on the higher order bits whenever their inclusion does not change the final result. An important example of C_{eff} reduction efforts in aSoC is the interconnect disabling. Although the scheduled communications guarantee the periodic

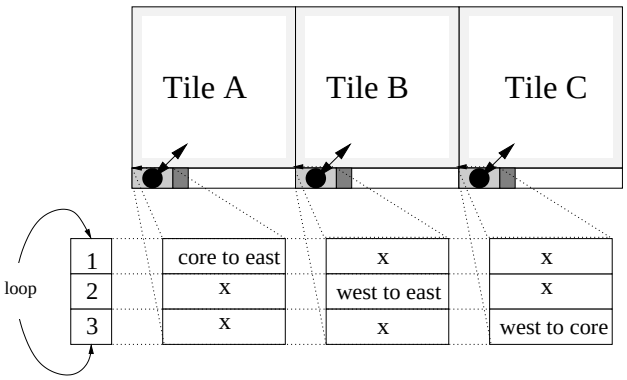

Figure 3: Example Data Stream

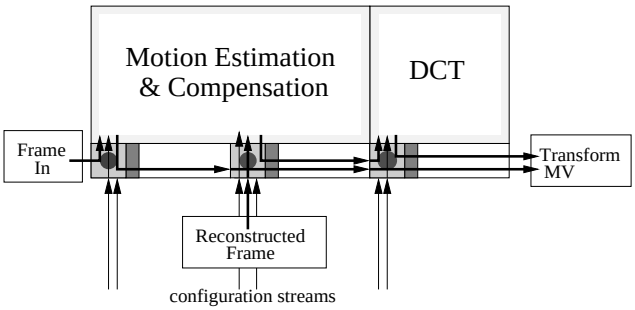

Figure 4: System Under Test

generation of specific streams, cores may not need or be able to use all stream instances. ASoC keeps track of the unused streams by including a valid bit with each data word. Using this valid bit as an enable for interconnect drivers eliminates switching activity for the unused data portion of the transfer.

We have built a motion estimation core based on the principles in [4] which can be dynamically configured to use one of three search methods: full, spiral, or three step. Method selection trades power consumption for motion estimation and possibly overall video quality. A control tile with access to the global power and quality requirements chooses the best search method and sends the information to an ME tile on a dedicated configuration stream.

Independently developed heterogeneous cores may require independent clock domains to meet critical path requirements. Additionally, reconfigurable IP cores may require reconfigurable clock domains. A simple configurable clock reference generator provided at each tile multiplies or divides the global interconnect synchronization clock by 2^n, where n represents a 3 bit binary number. The clock value is loaded at run-time through the tile *local config.* line to the controller. Such a system supports dynamic power management by reducing unnecessary slack in core computations and potentially improves battery life as well by reducing current spikes [7].

Finally, changes in the global schedule eliminate data transfers through unused tiles. System communication can be reconfigured at run-time using jump or load commands sent through the tile *local config.* line to the controller. A jump command changes a local pointer to a different schedule in the interconnect memory while a load command loads the contents of the new schedule.

4 Methodology

The demonstration system, shown in Figure 4, consists of two of the most power hungry components of video encoding; discrete cosine transform (DCT) and motion estimation (ME) [4]. These two cores are implemented at the register-transfer level (RTL) and include many dynamic power savings features. The DCT is a replicated row accumulate (RAC) unit implementation [5] and includes dynamic power savings mechanisms such as most significant bit (MSB) rejection and row column classification (RCC), found in [13]. The ME core permits selection of several search algorithms including full, spiral and three step [4]. It also allows selection of search range. These two cores combine to make an adequate test bench for the architecture as their incorporation demonstrates aSoC's asynchronous interface and aSoC's ability to deal with reconfiguration. The interconnect and interface system has been implemented as custom layout in 0.18 micron technology, and is modeled in HSPICE using the Berkeley Predictive Tools. A C-code motion compensation unit is attached to the ME core. It bridges the two cores of interest by generating a motion compensated difference frame for the DCT.

ASoC modularity creates an environment where system power is accurately modeled as the sum of individual component power provided that the timing and data activity are accurately captured and used in the individual power estimations. Three different simulators are used to analyze the performance and power consumption of this aSoC model. First, the two cores are simulated independently at the RTL level, using Synopsys tools, on representative bit streams from video benchmarks. The simulation includes TSMC standard cell libraries to approximate both performance and power consumption of cores for each mode of operation and desired frequency. The aSoC network simulator is used to determine system performance and network activity. Using the known core delays, the aSoC simulator accurately models the cycle-by-cycle

connectivity, flow control, and core usage. Core usage, showing idle and active time can be fed back through Synopsys to compute core power. Based on network activity and HSPICE circuit simulation of the interconnect, the network power consumption (P_{int}) can be tabulated as follows:

$$P_{int} = \sum_{t} P_{IF/D} + \sum_{s} 0.5 \times (33 \times N_{v_s}.P_s + N_{iv_s}.P_s) \quad (2)$$

Where t represents the number of tiles, $P_{IF/D}$ is the overhead of the instruction memory fetch and decode, and s is the number of streams. N_{v_s} and N_{iv_s} are the number of valid and invalid transfers for stream s while P_s is the power consumed in transferring 1 bit through stream s.

This system is setup to process 352x240, 8 bit pixel frames at a rate of 30 frames a second. The input compensated, reconstructed and transformed frame data are packed 4 pixels to a word. Each motion vector uses one full word of data.

5 Results

The following results showcase aSoC's capabilities. The first experiment uses the schedule shown in Table 1 to process MPEG P-frames. At each clock cycle the connectivity for each tile is established as shown. The PC cycles through instructions 0 to 9 and loops back to 0. The schedule length of 10 is used to imply that these three tiles are part of a much larger system. This experiment shows the benefit of allowing reconfiguration in addition to the benefit of clock signal configuration. Column 2 of Table 2 shows the clock rate of each subsystem required to meet the deadlines of the input video stream. Column 3 shows the power numbers of both cores and interconnect for a system, which processes data at these minimum frequencies. The last column dramatizes the pitfalls of forcing each tile to synchronize to the worst case clock frequency. In this part of the experiment core power increases dramatically as both cores are developed without clock gating systems of their own. The aSoC network simulator confirms that the clock modified system meets all input/output requirements and identifies network activity for the interconnect power tabulation shown in the last row.

When processing I-frame data, the large ME/MC system could be left unused if the input frame is rerouted directly to the DCT as shown in Table 3. A jump command used during run-time changes the pointer to select the modified schedule represented as instructions 10 through 19. During jumps, care must be taken to assure the new schedule does not change the other streams in the system. In this new schedule

	Tiles		
Cycle	ME	MC	DCT
0	(Frame in) $w \rightarrow ip_1$	(Saved Frame) $s \rightarrow ip_1$	(DCT Frame) $op_1 \rightarrow e$
1	(MV), $op_1 \rightarrow e$ (config.) $s \rightarrow ip_2$	(MC Frame), $op_1 \rightarrow e$ (config.) $s \rightarrow ip_2$	- (config.) $s \rightarrow ip_2$
2	- (config.) $s \rightarrow i$	(MV) $w \rightarrow e$ (config.) $s \rightarrow i$	(MC Frame) $w \rightarrow ip_1$ (config.) $s \rightarrow i$
3	- -	- -	(MV) $w \rightarrow e$
.	-	-	-
.	-	-	-
9	-	-	-
(Steam Name), op=output Coreport, ip=input Coreport n,s,e,w=north, south, east, west, → = connect, - = don't care			

Table 1: P Frame Communication Schedule

	Optimal Independent Clocks		Fixed Worst Case (110MHz)
IP:Mode	Frequency	Power	Power
ME:FS	105MHz	973mW	973mW
ME:spiral	9.9MHz	76mW	659mW[1]
ME:TSS	2.75MHz	25mW	580mW
DCT	9.6MHz	54mW	349mW[1]
Interconnect	6.34MHz	.14mW	.81mW
[1] Spiral search and DCT cannot run at 110MHz They run at 100MHz and 90MHz respectively.			

Table 2: Power for Modes and Clock Rates

	Tiles		
Cycle	ME	MC	DCT
10	(Frame in) $w \rightarrow ip_1$	(Saved Frame) $s \rightarrow ip_1$	(DCT Frame) $op_1 \rightarrow e$
11	- (config.) $s \rightarrow ip_2$	(Frame in), $w \rightarrow e$ (config.) $s \rightarrow ip_2$	- (config.) $s \rightarrow ip_2$
12	- (config.) $s \rightarrow i$	- (config.) $s \rightarrow i$	(Frame in) $w \rightarrow ip_1$ (config.) $s \rightarrow i$
.	-	-	-
.	-	-	-
19	-	-	-

Table 3: I Frame Communication Schedule

of Table 3, the input frame in the ME tile is rerouted to the east in cycle 10 and passed through the MC tile in cycle 11. Making these changes does not affect the other streams of data so long as they do not use the wire between the ME and MC tiles in the first cycle. To assure the availability of this resource, a place holder is used during compilation preventing its use by any other stream.

A final result shows the overhead of the configuration streams. One main concern is the fact that configuration streams must be scheduled periodically along with the data. In the P-frame example there are potentially 6 configuration streams and only 5 data streams used by the tiles. Fortunately, aSoC has an abundance of interconnect bandwidth. In the example, for the 10 cycle schedule shown, these cores can support up to 150 streams (= 5 streams/cycle $\times$ 10 cycle). The result is that configuration only uses 4% of the available bandwidth. If needed, loop unrolling techniques could further increase the ratio of data to configuration bandwidth.

Power overhead is another major concern since in our example the number of scheduled transfers for configuration exceeds those for data. This is where the flow control system is helpful. Configuration streams go unused most of the time. For example, the motion estimation core uses at most 1 word of configuration data per macroblock and node schedule reconfiguration happens at most once per frame. Disabling the interconnect for invalid transfers eliminates most of the potential power used. For the optimal and worst case interconnect power shown in Table 2, only 4% and 9% respectively are due to the configuration streams.

6 Summary and Future Work

This paper presents the dynamic power management capability of aSoC applied to video processing systems. Reconfigurable clock based system balancing creates an environment of just in time computing which can reduce overall power usage. Taking advantage of interconnect flexibility allows a system to dynamically change functionality and avoid unused computational units. Interconnect power consumption is shown to be low and the overhead due to configuration streams is shown to be under 10% for both bandwidth and power.

In our future architectures we are looking to add reconfigurable voltage regulation systems to each tile. These systems would allow us to fully take advantage of the just in time computing capabilities of our clocking system. In addition, we are attempting to fabricate a 4 tile demonstration system.

Acknowledgments

In addition to the authors listed, four other students have significantly contributed to the aSoC project; Sriram Srinivasan, Manoj Sinha, Srividya Srinivasaraghavan, and Subramanian Venkatraman. This work is based upon work supported by the National Science Foundation under Grant Numbers CCR-0081405, CCR-9988238, CCR-9875482.

References

[1] L. Benini, Giovanni De Micheli, *Dynamic Power Management, Design Techniques and Cad Tools*. Kluwer Academic Publishers., 1998.

[2] F. Ichiba, et al, "Variable Supply-Voltage Scheme with 95%-Efficiency DC-DC Converter for MPEG-4 Codec," *Proceedings: Intl Symposium on Low Power Electronics and Design*, 1999.

[3] I. Hong, et al, "Power Optimization of Variable Voltage Core-Based Systems," *Proceedings: Design Automation Conference*, 1998.

[4] P. Kuhn, *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Kluwer Academic Publishers. 1999.

[5] V. Lakamraju, *A Power-Aware Synthesizable Core for The Discrete Cosine Transform*, Master's thesis, University of Massachusetts, Department of Electrical and Computer Engineering, 2001.

[6] J. Liang S. Swaminathan, R. Tessier, "aSOC: A Scalable, Single-Chip Communications Architecture," *Proceedings, IEEE International Conference on Parallel Architectures and Compilation Techniques*, Oct. 2000.

[7] T. Martin, D. Siewiorek, "A Power Metric for Mobile Systems," *Proceedings: International Symposium on Low Power Electronics and Design*, 1996.

[8] *MPEG-4 Overview: ISO/IEC JTC1/SC29/WG11 N4030*, Moving Picture Experts Group, 2001.

[9] J. Rabaey, *Digital Integrated Circuits, A Design Perspective*, Prentice Hall Inc., 1996.

[10] D. Shoemaker, C. Metcalf, and S. Ward, "NuMesh: An Architecture Optimized for Scheduled Communication," *Journal of Supercomputing*, V10, pp 285-302, 1996.

[11] T. Simunic, et al, "Dynamic Voltage Scaling and Power Management," *Proceedings: Design Automation Conference*, 2001.

[12] *CXD1922Q: An Industry Breakthrough in MPEG-2 Technology (web-page)*, Sony Electronics Inc, 2001, www.sel.sony.com/semi/cxd1922qwp.html.

[13] T. Xanthopoulos, A. Chandrakasan, "Low-Power DCT Core Using Adaptive Bitwidth and Arithmetic Activity Exploiting Signal Correlations and Quantization," *Proceedings, IEEE Journal of Solid-State Circuits*, V35, No 5, May 2000.