A Low Power Video Processor

Uzi Zangi
Zoran Corporation
Advanced Technology Center, Haifa 31024, Israel
Tel +972-4-854-5759
uzi@zoran.com

Ran Ginosar
VLSI Systems Research Center
Technion—Israel Institute of Technology
Haifa 32000, Israel
ran@ee.technion.ac.il

1. Abstract

Multiple power saving methods were applied to a video processor for color digital video and still cameras. Architectural level methods failed to save power: asynchronous design, dynamic voltage scaling, bus switching minimization, pipeline stage merging, reduction of switching times and clock gating. However changing the algorithm to work on pixel differences yielded 3-15% power reduction in typical cases.

1. Introduction

Common designers of ASIC systems for low power applications, such as battery operated consumer products, are typically constrained to employ an existing cell library and an existing fabrication process. Thus, many proposed power reduction techniques that apply to the basic circuit or the device levels are inapplicable. On the other side of the spectrum, the designer is often constrained by system level specifications that cannot be changed, thus also prohibiting low power redesign at that level. What remains is for the designer to use best judgement at the architectural levels (RTL and behavioral), and at the algorithmic level.

We have investigated the efficacy of such commonly proposed methods in the case of a video processor for still and video portable cameras. The processor receives the digitized signal from the color image sensor (CCD or CMOS) and converts it to standard digital video, available for display, compression, storage and transmission [1]-[5]. We have found that while most common architectural level methods did not save any power, an algorithmic change yielded 3-15% reduction in power. The algorithmic approach exploited the same features that make video amenable to compression.

The architecture of the baseline video processor is described at an algorithmic level in Sect. 2. Some common power saving methods were rejected at an early stage, as explained in Sect. 3. Other methods, discussed in Sect. 4, were tried but proven useless in this application, as shown in Sect. 5. The winning method is described in Sect. 6, and its simulation is discussed in Sect. 7. A more detailed report of this research is provided in [11].

2. The Color Camera Video Processor

A digital camera video processor receives digitized outputs of an image sensor having a complementary color mosaic filter, and generates a standard video signal, typically in the YC_RC_B color space [1]-[5]. The processor comprises four datapath units and a controller (Fig. 1). The Input Unit averages all black pixels and subtracts that offset from each image pixel. Two delay lines generate a simultaneous neighborhood of three pixels one above the other. The Luminance Unit reads in the three pixels simultaneously. It first generates luminance levels '*y*' by adding two successive pixels. Then, a spatial low pass filter is implemented by adding up all nine or eight pixels of the neighborhood. The result is used to derive a high pass filter and for edge-enhancing *y*. This and all other units also contain registers for pipeline balancing and logic for range limiting, rounding, scaling, and handling boundary conditions around the perimeter of the image.

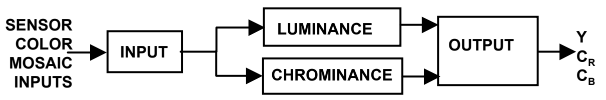

Fig. 1: **Block diagram of the video processor**

The Chrominance Unit receives the same inputs and generates color difference signals d_R, d_B. The Output Unit converts yd_Rd_B to *rgb*, applies white balance and gamma correction, converts the result to YC_RC_B, and performs "chroma suppression". Full details are given in [11].

3. Power Saving Methods That Were Rejected

Two power reduction methods were investigated which we decided to reject:

Asynchronous Design: Three factors are typically expected to reduce power: The clock network is eliminated, each module receives inputs only when it needs to compute, and dynamic voltage scaling may be employed. This method was shown to save up to 80% of total power during periods of low activity, when the processor may be slowed down [6],[7]. We employed 'bundled data' methodology with delay lines [8] and full handshake interconnect [9], but found that the extra power required by the delay lines and the handshake circuits far exceeded the power saved by the elimination of the clock. This was due to the very low frequency of the clock (13.5MHz, the video input/output rate).

Bus Switching Reduction: This is possible by selecting between sending a value or its complement [10]. Hamming distance logic on the sender side determines which of the value or its complement incurs less switching, compared to the previous value that is dynamically stored on the bus. Our analysis shows that, for the average conditions at this video processor, the bus load must exceed 1pF before this method shows any benefit [11]. Thus, it is inapplicable to this small processor.

4. Power Saving Methods That Were Tried But Did Not Work

Three standard power reduction methods were attempted, but turned out to provide no savings:

Pipeline Stage Merging: Thanks to low clock frequency, more logic could be integrated in each stage, in order to save power by reducing the number of registers. Successive stages were merged wherever possible, removing 100 bits out of 521 registers (19% reduction). It resulted in reduction of capacitance and load, but also lead to spurious transitions due to hazards in the longer reconverging combinational paths. As shown below, it yielded a negative net effect on power.

Clock Gating: This method was applied to the circuit after pipeline stage merging, but fell short of saving any power (as discussed below).

Switching Time Acceleration: Faster switching time of a gate shortens the duration of the short circuit current I_{SC} of other gates driven by it. This speedup can be achieved by increasing the driver, but this in turn consumes more dynamic power at the driver. This tradeoff is not always positive, and should be examined in each case. Having used the Synopsys RTL synthesizer to generate our circuits, we specified to the synthesizer the maximum allowed switching delay as either 1, 2, or 3ns to examine this effect. The result, detailed in [11], shows no power saving.

5. Simulation Results (I)

The methodology is schematically shown in Fig. 2. The algorithm was implemented in C++, to verify functional correctness and accuracy of the various Verilog RTL models, executing on a diverse set of images. The RTL was synthesized with the Synopsys tool, using a 0.35µm Compass library. The netlist was placed and routed on Cadence Opus, and the extracted physical netlist, including interconnect parasitics, was simulated (with the same image input data) on Epic PowerMill to obtain current (and power) estimates.

Three different designs are examined in Tab. 1: A baseline *A* (as described in Sect. 2 above), a circuit *B* after merging pipeline stages, and a circuit *C* with both merged stages and clock gating.

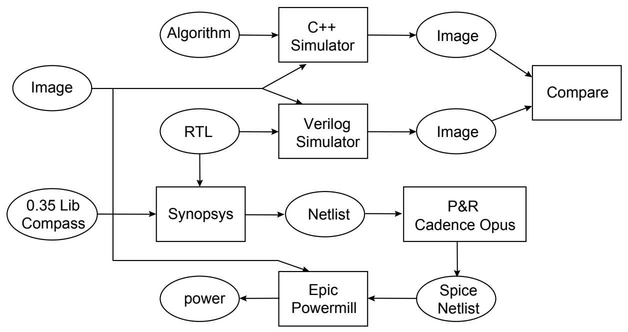

Fig. 2: **Design and simulation methodology**

As can be seen in Tab. 1, merging pipeline stages reduced the registers by 19% from 512 to 412, and register current indeed declined by 20%. The current consumed by the combinational logic, on the other hand, was 20% *higher* in the 'improved' circuit relative to the baseline. This effect is caused by spurious transitions in the longer pipeline stages, dissipating more dynamic and short circuit power. Hazard-free logic may be required to overcome this problem.

With clock gating, register current has been slightly reduced thanks to clock gating at idle times, but the logic current increased more than the savings. Since register activity in the video processor is very high (~90%), the savings achieved by clock gating is very low, and it does not justify the overhead. This method is probably efficient for chips with higher clock frequency and lower register activity.

All these techniques involve a tradeoff that must be carefully evaluated, and none has been effective in our case. The DSP nature of the processor, where each cycle the data flow through all parts of the processor, precludes clock gating and pipeline merging.

Circuit	A	B	C
Num. Gates	13163	12501	13182
Num. Registers	521	421	421
Logic current (mA)	6.8	8.25	8.76
Registers current	3.8	3.03	3.0
Total current	10.7	11.3	11.8

Tab. 1**: Merging pipeline stages and clock gating did not save power in the video processor.**

6. The Winner: Algorithmic Transformation

We constructed a different video-processing algorithm that yielded almost the same result at a lower switching activity. We have taken advantage of the facts that video pixels are often spatially correlated, and that most of the processing algorithm is linear.

Thus, we resorted to computing the difference of every two successive pixels, and converting the linear section of the algorithm to work on those differences. The differences are mostly zero or 1-2 bit numbers, and the logic exploits it.

This observation is obviously false near edges in the image. Due to rounding errors, the difference algorithm performs poorly after any sharp image gradients. We have retained the original circuitry and have employed it each time an edge has been encountered. Once the gradient has subsided and relatively stationary pixel levels have been re-established, the difference algorithm is turned back on and the original algorithm is shut off.

The new combined original/difference algorithm has been designed to create output that deviates by no more than a single digital value from the original (a 'single lsb' error), and simulations have verified this on all our test images.

7. Simulation Results (II)

Five images were simulated on the new circuit which yielded different portions of pixel differences (Tab. 2). Images A-C exhibit typical edge contents, and result in less than 50% pixels which could be processed as differences, and in up to 15% power saving. Image D has little edges, and flat image E contains but one value. Both exhibit very high ratio of pixel differences. However, they require less power in the baseline processor, so that there is no saving in both cases. Actually, there is some loss, due to the additional overhead of the more complex architecture. Ignoring these extreme cases, the new architecture is useful for power saving on more typical images.

Image	Baseline Current [mA]	Pixel Differences [%]	Reduced Current [mA]	Power Saving [%]
A	7.4	5	6.3	15%
B	5.9	9	5.4	8%
C	5.7	37	5.3	7%
D	4.2	66	4.3	-3%
E	0.8	98	0.9	-12%

Tab. 2: **Power savings with the pixel difference algorithm**

8. Conclusions

We have found that for a small (13,000 gates), low frequency (13.5MHz), video DSP ASIC, a number of commonly advocated power saving methods at the logic and architectural levels were inapplicable. Algorithmic change, on the other hand, yielded a 3-15% power reduction. The study was based on PowerMill simulation of actual image data of a 0.35μm standard cell design, compiled with Synopsys from an RTL description and physically laid using Cadence Opus P&R tools. The new algorithm exploits inter-pixel correlation and operates on pixel differences rather than on the original, resulting in lower power small-number arithmetics.

References

[1] L.J. D'Luna and K.A. Parulski, "A systems approach to custom VLSI for a digital color imaging system," *IEEE JSSC,* **26**(5), pp. 727-737, May 1991.

[2] H. Ohtsubo *et al.*, "A 0.8 μm CMOS digital signal processor for a video camera," *IEEE Trans. Consumer Electr.,* **39**(3), pp. 407-412, Aug. 1993.

[3] J.C, Wang, D.S. Su, D.J. Hwung and J.C. Lee, "A single chip CCD signal processor for digital still cameras," *IEEE Trans. Consumer Elec.,* **40**(3), pp. 476, Aug. 1994.

[4] W.H. Chan and C.T. Youe, "Video CCD based portable digital still camera," *IEEE Trans. Consumer Elec.*,. **41**(3), pp. 455, Aug. 1995.

[5] S.S. Wang, C.H. Wu and N.Y. Hu, "A real-time digital signal processor for use with the interline transfer color CCD imager," IEEE Workshop on Charge-Couple Devices and Advanced Image Sensors, 1995.

[6] L. S. Nielsen, C. Niessen, J. Sparso, and C. H. van Berkel, "Low-power operation using self-timed circuits and adaptive scaling of the supply voltage," *IEEE Trans. VLSI,* **2**(4), pp. 391-397, Dec. 1994.

[7] K. van Berkel, R. Burgess, J. Kessels, A. Peeters, M. Roncken, and F. Schalij, "Asynchronous circuits for low power: a DCC error corrector," *IEEE Design & Test*, **11**(2), pp. 22-32, Jun. 1994.

[8] S. Hauck, "Asynchronous design methodologies: An overview," *Proc. IEEE*, **83**(1), pp. 69-93, Jan. 1995.

[9] T. H.-Y. Meng, R. W. Brodersen, and D. G. Messerschmitt, "Asynchronous design for programmable digital signal processors," *IEEE Trans. Signal Processing*, **39**(4), pp. 939-952, Apr. 1991.

[10] M. Stan and W. Burleson, "Limited-weight codes for low-power I/O," Proc. Int. Workshop on Low Power Design, pp. 209-214, Apr. 1994.

[11] U. Zangi and R. Ginosar, *Low power video processor,* Technical Report CC244, Electrical Engineering Department, Technion, Feb. 1998.