A 242mW, 10mm² 1080p H.264/AVC High Profile Encoder Chip

Yu-Kun Lin, De-Wei Li, Chia-Chun Lin, Tzu-Yun Kuo, Sian-Jin Wu, Wei-Cheng Tai, Wei-Cheng Chang, and Tian-Sheuan Chang
Institute of Electronics, National Chiao-Tung University
Hsinchu, Taiwan 300
{yklin, tschang}@twins.ee.nctu.edu.tw

ABSTRACT
A 1080p high profile H.264 encoder is designed by the robust reusable silicon IP methodology and fabricated in a 0.13μm CMOS technology with an area of 10 mm² and 242mW at 145MHz. Compared to the state-of-the-art design targeted at 720p baseline, this design reduces 53.4% power and 46.7% area through parallelism enhanced throughput and cross stage sharing pipeline.

Categories and Subject Descriptors
C.3 [Special-Purpose and Application-Based Systems]: Signal processing systems; B.7.1 [Integrated Circuits]: Types and Design Styles—Algorithms implemented in hardware, VLSI.

General Terms
Algorithms, Design.

Keywords
H.264, Encoder, High Profile, 1080p.

1. INTRODUCTION
The H.264 standard has been adopted as the major coding standard in recently popular high definition video due to its excellent coding efficiency. Because of its high complexity, several works have been presented [1-3] but their performance is limited to baseline 720p (1280x720) [1-2] or SDTV (640x480) [3]. Besides, the design targets of previous designs are focused on H.264 baseline profile which only provides the basic video quality and compression efficiency. But the high resolution applications such as High Definition Television (HDTV), HD-DVD, and BD all adopt 1080p (1920x1080) H.264 high profile for higher compression efficiency and better video quality, which cannot be supported by previous works. However, the main stream 1080p high profile application presents a series of new design challenges in throughput, cost and power because of at least 4X higher complexity than in the 720p baseline.

Thus, a 0.13μm 1080p (1920x1080) high profile H.264 video encoder is presented by adopting parallelism enhanced throughput algorithm and cross stage sharing pipeline. This design not only supports real-time encoding 1080p video by H.264 high profile specification but also achieves 46.7% and 53.4% reduction in area and power respectively when compared to a state-of-the-art 720p baseline design [2].

The organization of the paper is shown below. Section 2 introduces the overview of the whole system. The details and design features of the important functional modules are described in Section 3. The design flow is presented in Section 4. The verification methodology and chip test result are shown in Section 5. The chip specification, comparison and power are shown in Section 6. Finally, Section 7 concludes this paper.

2. SYSTEM OVERVIEW
Figure 1 presents the system overview. The new high profile coding tools are included as the shaded parts. An important challenge is to add these new coding tools with the similar throughput and minimum hardware overhead. The detailed solution is discussed in the Section 3.

As shown in Figure 1, the system architecture of the proposed encoder has three macroblock pipelined stages. The first stage is the integer motion estimation (IME) stage which occupies the most computation and memory resource of the entire H.264 encoder. In the second stage, intra prediction and fractional motion estimation (FME) are placed in the same stage to share the current block buffer and pipelined buffer. Intra prediction uses the neighbor pixels to predict the current block and the FME refines the result of IME stage. The third stage is the entropy coding stage including Context-Adaptive Variable Length Coding (CAVLC) and Context-Adaptive Binary Arithmetic Coding (CABAC), which both provide high compression efficiency to generate the final bit-stream. However, the intra prediction and FME in the same stage could cause timing conflict in the reconstruction of inter and intra prediction and thus reconstruction hardware has to be duplicated. This conflict is that the intra predictor in the second stage needs the reconstructed boundary pixels of previous block immediately for intra mode decision and thus its reconstruction must finish in the second stage. But the reconstruction after the final mode decision must execute after all predictions are done and should occur in the third stage. To solve the reconstruction hazard, we place the reconstruction stage cross the second and third stages so that the intra and inter predictions can share the same hardware in different time slots. With the above reconstruction sharing technique, we can eliminate one extra reconstruction hardware unit and its power.
3. FUNCTION DESCRIPTION OF IMPORTANT MODULES

3.1 8-pixel Parallelism Intra Encoder

Figure 2 (a) presents the overview of this design except ME and entropy coding. The new 1080p high profile coding tools including intra8x8 prediction and 8x8 integer DCT increase the complexity by 37.5% and the throughput by 2.5X compared to the baseline profile. Besides, the structure and data hazards will occur since the new high profile tools such as intra8x8 predictor need extra reconstruction and different reconstructed boundary data and thus will conflict with intra4x4 modules. For the data hazard, we adopt independent boundary buffer for intra8x8 prediction to eliminate it.

To solve the high throughput request and structure hazard, this design adopts eight-pixel parallelism. To further improve throughput, we parallel process intra8x8 and intra4x4/16x16 and use interlaced scheduling to minimize the stall cycles by data hazards. However, direct implementation will cause high cost due to eight-pixel parallelism. To reduce cost, we merge the reconstruction of intra4x4/16x16 and intra8x8 into one and further share it with reconstruction of ME as stated above. To further decrease the cost, we adopt intra8x8/16x16 recomputation so that the best mode result and its prediction value of intra8x8/16x16 are not saved and recomputed if being chosen. With this, 2560 bits of memory can be reduced. Besides, using eight-pixel architecture in reconstruction and quantization phases can save the extra buffers between different pixel-parallelism phases.

Figure 2(b) shows the performance of these proposed algorithms. The cross-stage reconstruction component can reduce 24.2% gate count, and then the intra8x8/intra16x16 recomputation can save 10.9% gate count. The merged reconstruction module for intra and recomputation schemes can save the gate counts by 7.7% and 9.26%, respectively. In summary, 42% gate counts can be reduced by these techniques than direct implementation.

3.2 High Throughput Motion Estimator

Figure 3 shows the motion estimation (ME) algorithm and its architecture. To achieve 1080p resolution with bi-directional prediction, we adopt parallel single step processing for ME. Thus, for integer motion estimation (IME), we use a parallelized subsampling algorithm, Parallel Multi-resolution ME (PMRME) [4]. It searches three subsampling levels of different search ranges in parallel so that all searches are done within 256 cycles in single step. This provides higher throughput than [2] and [3] using two and four steps in IME respectively. Within 256 cycles, the high throughput IME can process bi-directional predictions sequentially while still meeting 1080p requirement. Thus, a single IME module cost is enough for both directional predictions.

Besides, to support search range (SR) ±128 and reduce the hardware cost within the limited quality loss, level 1 and 2 provide different subsampling ratio according to the search range for large motion vectors, and the search centers of the two levels are at (0,0) for further data reuse. Furthermore, to compensate quality loss of subsampling to meet the 1080p requirement, the level 0 without subsampling is centered at the motion vector predictor (MVP) to cover the most occurred motion vectors.

After IME, we use Mode Filtering (MF) to select only two best modes for FME refinement so that FME tests at most 18 motion vectors instead of 41 motion vectors in [1]. As for the fractional motion estimation part, we use the six pixels only instead of 17 pixels and 25 pixels in [1][3], and reduce at least 64.7% complexity 76% of processing units used in [2]. Besides, in order to improve the throughput, we finish the FME stage within a single iteration to double the throughput than previous works. To further reduce the bandwidth of FME, we use the Non-Subsampling Reference Memory Sharing (NSRMS), another cross stage technique, which uses three SRAM banks to enable sharing of level 0 reference memory of IME and that of FME. FME searches only six candidates in a single step so that only six processing units are needed, which eliminates 76% of processing units used in [2].
Figure 4(d) shows the trade-off between the video quality loss and the search point reduction for motion estimation. In previous work [2], the fast motion estimation algorithms may result in 0.6dB PSNR loss, which may cause obvious image distortion. In our proposed ME algorithm, only 0.1dB quality loss is required, but the search points (search complexity) can reduce 98.7%. The fast IME (PMRME) can save the complexity by 91.7%. As for mode filtering and fractional motion estimation, they can reduce the complexity by 56% and 64%, respectively. It means the three algorithms all can reduce the complexity a lot.

In addition to the complexity, our proposed architecture also reduces the hardware cost including area, internal SRAM size and memory bandwidth. Compared with [1][5], the proposed gate count for motion estimation can decrease 30% and 62.5% gate count in IME and FME part as shown in Figure 4(b). These benefits come from the reduction of processing units to calculate the sum of absolute difference (SAD). For local memory reduction shown in Figure 4(c), the memory size can save 86% because PMRME uses the subsamping techniques and only the sampled pixels are necessary to be stored in local memory.

Finally, the memory access can reduce 46% due to the subsampling PMRME technique as Figure 4(a) presents.

4. DESIGN FLOW

Figure 5 shows the completed design flow and the adopted design tools. This flow follows the reusable silicon IP design methodology. We first decide the system specification like video size and profile, and then we develop the golden C model by modifying the reference software of H.264 [6] to include our own developed fast algorithms. During the development, we analyze their performance such as video quality and bit-rate. After all components in the software model are verified, the model becomes the golden model and we will generate the test patterns from it and directly compare the output of hardware design with the results of the golden model.

For the hardware design, we first decide the overall system architecture by the analysis from software model. Then, the design is implemented by Verilog and simulated by Cadence NC-Verilog. The outputs of the hardware components are compared with the result of golden model to check the correctness. In addition to the simulation, we also use the linting tools, nLint.
from SpringSoft, to make sure the good silicon IP coding style. After simulation, we synthesize our design by the synthesis tool, Synopsys Design Compiler. During the synthesis, we optimize the power consumption by adding the gated clock and gate level optimization through the Synopsys Power Compiler. Moreover, the scan chains can be easily inserted for manufacturing test. After synthesis and power optimization, we adopt the equivalence check tool, Conformal LEC, to check the equivalence between RTL and gate level netlist. Then, we place and route this design by using Candence SoC encounter and do the DRC and LVS checking by Mentor Calibre. The post-layout netlist is further analyzed by Synopsys Prime Time and Prime Power to get the timing analysis and preliminary power estimation. The design is further refined based on the analysis result until it meets our specification.

5. VERIFICATION AND TEST STRATEGERY

5.1 Verification Flow

The major problem to verify the video encoder is how to generate various test patterns. To simplify the pattern generation, we use the golden C model to dump its intermediate values for block level verification and the final result for system level verification. Besides, we use two types of video sequences for verification, the artificial videos for corner cases and the natural videos for other random cases. Finally, the output bit-stream of proposed design is also sent to a general software H.264 decoder to check the decoding results. The artificial video sequences are designed to achieve 100% functional coverage for our design. Figure 6 shows an example of these artificial sequences. For example, Figure 6 (a) shows the test blocks for inter and intra prediction respectively. The blocks for inter prediction contain all block sizes from 16x16
These blocks can test all block sizes
The blocks for testing inter predict modes
Change the location of these blocks to test all motion vectors

(a) (b)

Figure 6. The artificial test sequences for 100% functional coverage. (a) is for intra frame (b) is for inter frame.

to 4x4 and correspond to the macroblocks in Figure 6 (b). Therefore, we can change the motion vectors by adjusting the location of these macroblocks in Figure 6 (b). As long as we move these blocks to all possible positions in the frame, all motion vectors can be tested for the inter prediction modules. As for the intra predictor, these test blocks in the second row in Figure 6 (a) include all intra prediction modes. The first (leftist) macroblock tests intra8x8 mode, and then the second and the third macroblock examine the vertical and horizontal modes for intra4x4 and 16x16. Finally, the last macroblock checks the correctness of intra4x4 mode. After all possible corner cases are verified by these artificial videos, the natural video sequences are used as the test patterns. By these schemes, the functional coverage of this design is 100%.

Besides the artificial video sequences, we use natural video sequences to find any possible errors. Figure 7 presents the decoded images of bit-streams from golden model and our design, respectively. This sequence contains the complex texture in the calendar and the background to test intra prediction and includes the components with high motion (the train) to test inter prediction.

By these verification strategies, our design can be guaranteed to encode the videos with our proposed specification correctly.

5.2 Chip Testing Methodology

The chip testing is divided into block level and chip level. For block level, the major problem is the limited number of I/O pins to control and observe internal signals. To solve this problem, we add a debug mode control in our design and multiplex the important internal signals with original signals to the external pins. These internal signals includes the key signals such as motion vectors, selected block sizes by the inter prediction modules, the intra prediction modes from intra predictors, and the residue data after quantization. With this, we can switch to the debug mode and check the result of these internal modules to know which part fails if the output bit-stream of the chip is wrong. By the debug mode, we can check the internal signals of the chip without increasing the I/O pins.

For chip level testing, we use the functional test patterns in chip level verification as the test signals in the Agilent 93000 SoC Test System to test our chip. Besides, the chip power can be also calculated by measuring the current through the tester.
6. CHIP SPECIFICATION AND PERFORMANCE

Table 1 shows the specification and comparison with other works. Our design is fabricated by UMC 0.13µm process. The core size is 3.17x3.17mm. The equivalent gate count except memory is 593K and the internal SRAM is 22KB. Our design can support H.264 high profile level 4 and can real-time encoding 1080p video in 30fps. The maximum search range of ME is ±128. The operating frequency is 62.5MHz and 145MHz for 720p and 1080p respectively. Compared to the state-of-the-art design [2] targeted at 720p baseline in low power mode, the operating frequency are 13% lower than [2] due to our enhanced throughput and complexity reduction. The PSNR loss is only 0.1dB that is better than [2] that has 0.6dB quality loss at the low power mode.

Table 1 Chip specification and comparison

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CMOS Tech.</td>
<td>UMC 0.13µm</td>
<td>TSMC 0.18µm</td>
<td>TSMC 0.13µm</td>
<td>TSMC 0.18µm</td>
</tr>
<tr>
<td>Core Size</td>
<td>3.17x3.17 mm²</td>
<td>7.68x4.13 mm²</td>
<td>4.3x4.3 mm²</td>
<td>3.47x3.7 mm²</td>
</tr>
<tr>
<td>Chip Size</td>
<td>3.74x3.74 mm²</td>
<td>N/A</td>
<td>4.9x4.9 mm²</td>
<td>N/A</td>
</tr>
<tr>
<td>Package</td>
<td>CQFP 208-pin</td>
<td>N/A</td>
<td>CQFP 160-pin</td>
<td>N/A</td>
</tr>
<tr>
<td>Profile</td>
<td>H.264 High/Level4</td>
<td>H.264 Baseline</td>
<td>H.264 Baseline</td>
<td>H.264 Baseline</td>
</tr>
<tr>
<td>Maximum</td>
<td></td>
<td>1280x720@30fps</td>
<td>1280x720@30fps</td>
<td>640x480@30fps</td>
</tr>
<tr>
<td>Support</td>
<td>(1080p)</td>
<td>(720p)</td>
<td>(720p)</td>
<td>(SDTV)</td>
</tr>
<tr>
<td>Resolution</td>
<td>1920x1080@30fps</td>
<td>1280x720@30fps</td>
<td>720p</td>
<td></td>
</tr>
<tr>
<td>Maximum</td>
<td></td>
<td>H: -128→+127</td>
<td>H: -32→+31</td>
<td></td>
</tr>
<tr>
<td>Search Range</td>
<td>V: -128→+127</td>
<td>V: -32→+31</td>
<td>V: -32→+31</td>
<td></td>
</tr>
<tr>
<td>Quality Loss</td>
<td>0.1dB</td>
<td>N/A</td>
<td>Up to 0.6dB</td>
<td>N/A</td>
</tr>
<tr>
<td>Gate Count</td>
<td>593K</td>
<td>922.8K</td>
<td>470K</td>
<td>458.2K</td>
</tr>
<tr>
<td>SRAM Size</td>
<td>22KB /-frame</td>
<td>34.72KB</td>
<td>13.3KB</td>
<td>16.95KB</td>
</tr>
<tr>
<td>Operating</td>
<td>145MHz@1080p</td>
<td>81MHz @D1</td>
<td>108MHz@720p</td>
<td>54.5MHz@SDTV</td>
</tr>
<tr>
<td>Frequency (for</td>
<td>62.5MHz@720p</td>
<td>72/108MHz@720p</td>
<td>81MHz@720p</td>
<td></td>
</tr>
<tr>
<td>30fps)</td>
<td>28.5MHz@D1</td>
<td>30–96MHz@D1</td>
<td>72/108MHz@720p</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7.2MHz@CIF</td>
<td>10–28MHz@CIF</td>
<td>108MHz@720p</td>
<td></td>
</tr>
<tr>
<td>Power@baseline</td>
<td>785mW@720p</td>
<td>183mW@720p</td>
<td>67.2mW@SDTV/1.5V</td>
<td></td>
</tr>
<tr>
<td>Profile (for</td>
<td>581mW@D1</td>
<td>27mW@D1/0.9V</td>
<td>15.9mW@CIF/1.3V</td>
<td></td>
</tr>
<tr>
<td>30fps)</td>
<td></td>
<td>7mW@CIF/0.7V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power@high</td>
<td>242 mW@1080p/1.2V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>profile (for</td>
<td>116.61 mW@1080p/1.2V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30fps)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td></td>
</tr>
</tbody>
</table>

The operating frequency is 62.5MHz and 145MHz for 720p and 1080p respectively. Compared to the state-of-the-art design [2] targeted at 720p baseline in low power mode, the operating frequency are 13% lower than [2] due to our enhanced throughput and complexity reduction. The PSNR loss is only 0.1dB that is better than [2] that has 0.6dB quality loss at the low power mode. For the area, our core size is only 54% of [2]. In summary, the design outperforms other works to achieve 1080p processing with the less power consumption and area.

Figure 8 presents the power consumption of the proposed design. The power consumption is only 116mW and 242mW for 720p and 1080p respectively. Compared to the state-of-the-art design [2] targeted at 720p baseline in low power mode, the power consumption is 53.4% lower than [2]. For small resolution video, the level 1 and level 2 of IME are turned off when the video size is below CIF, and level 2 is off for D1 video size. Therefore, the power consumption is 6.74mW, only 42.6% of [3] and comparable to the lower power mode in [2] for baseline CIF video.

7. CONCLUSION

In this paper, we propose a high throughput and low power H.264 encoder. The design not only supports the high profile specification and more coding tools, but also uses smaller area and lower power by parallelism enhanced throughput and cross stage sharing pipeline. Besides, by the golden model and robust verification and testing strategies, our chip can work correctly and can be used as an IP for further reuse.

8. REFERENCES