Power Reduction in an H.264 Encoder Through Algorithmic and Logic Transformations

Maria G. Koziri
Department of Computer and Communication Engineering,
University of Thessaly
37 Glavani St.,
382 21 Volos, Greece
+30 2421074979
mkoziri@uth.gr

George I. Stamoulis
Department of Computer and Communication Engineering,
University of Thessaly
37 Glavani St.,
382 21 Volos, Greece
+30 2421074979
georges@uth.gr

Ioannis X. Katsavounidis
InterVideo, Inc.
46430 Fremont Blvd.,
Fremont, CA 94538, USA
+16 504921567
ioannis@intervideo.com

ABSTRACT

The H.264 video coding standard can achieve considerably higher coding efficiency than previous video coding standards. The keys to this high coding efficiency are the two prediction modes (Intra & Inter) provided by H.264. Unfortunately, these result in a considerably higher encoder complexity that adversely affects speed and power, which are both significant for the mobile multimedia applications targeted by the standard. Therefore, it is of high importance to design architectures that minimize the speed and power overhead of the prediction modes. In this paper we present a new algorithm, and the logic transformations that enable it, that can replace the standard Sum of Absolute Differences (SAD) approach in the two main prediction modes, and provide a power efficient hardware implementation without perceivable degradation in coding efficiency or video quality.

Categories and Subject Descriptors

B.2.4 [**Arithmetic and Logic Structures**]: High-speed Arithmetic – *algorithms, cost/performance.*

General Terms

Algorithms, Performance, Design.

Keywords

H.264 encoder, intra prediction, low-power implementation, motion estimation.

1. INTRODUCTION

Video has always been the backbone of multimedia technology. In the last two decades, the field of video coding has been revolutionized by the advent of various standards like MPEG-1 to MPEG-4 and H.261 to H.263, each addressing different aspects of multimedia. H.264[1] is a new standard which adds one more step in the endeavor towards video coding excellence and provides one-stop solution for wide range of applications. The standard has been developed by the Joint Video Team (JVT) comprised of both ISO/IEC and ITU-T. The primary goal of H.264 is to achieve higher compression while preserving video quality. The motivation for compression is to compensate for the ever-present constraints of limited channel capacity.

This increased compression efficiency of the new ITU-T H.264/MPEG-4 Advanced Video Coding (AVC) standard will lead to new application areas. Applications concerning broadcasting over cable, satellite, cable modem, terrestrial, etc., will benefit from the new standard. As for the field of mobile communication, H.264 will play an important role because the compression efficiency will be doubled in comparison to the coding schemes previously specified by Third-Generation Mobile (3GPP and 3GPP2) for streaming. The video coding technique used in H.264 follows a flexibility to be used in low-delay real-time applications.

Each picture of a video, which can either be a frame or a field, is partitioned into fixed-size macroblocks that cover a rectangular picture area of 16×16 samples of the luma component and 8×8 samples of each of the two chroma components (for the case of Y:U:V 420 color format). All luma and chroma samples of a macroblock are either spatially or temporally predicted, and the resulting prediction residual is transmitted using transform coding. For this purpose, each color component of the prediction residual is subdivided into blocks of various sizes. Macroblocks are grouped in slices, which generally represent subsets of a given picture that can be decoded independently. H.264/AVC supports five different slice-coding types. The simplest one is the I slice (where "I" stands for intra). In I slices, all macroblocks are coded without referring to other pictures within the video sequence (i.e. Intra Prediction). On the other hand, prior-coded images can be used to form a prediction signal for macroblocks of the predictive-coded P and B slices (where "P" stands for predictive and "B" stands for bi-predictive) (i.e. Inter Prediction). The remaining two slice types are SP (switching P) and SI (switching I), which are specified for efficient switching between bitstreams coded at various bit-rates.

One computational element that is met in both, Inter and Intra Prediction modes, is that of the Sum of Absolute Differences (SAD). In this paper we present a new algorithm that can replace SAD in the two main prediction modes, and the circuit that implements the proposed algorithm, which results in better speed and significantly lower power. The rest of this paper is organized as follows: Section 2 reviews Intra and Inter Prediction, as well as the basic concepts of SAD. In Section 3 a description of the proposed algorithm is introduced. Section 4 presents the coding efficiency and video quality results achieved from the software implementation of the new algorithm, while Section 5 presents the delay and power improvements achieved by the hardware implementation of the proposed algorithm.

2. INTRA AND INTER PREDICTION

2.1 Intra Prediction

In H.264, intra-prediction with block sizes of 4×4, 8×8 and 16×16 is used to compress I-Macroblocks. Intra coding refers to the case where only spatial redundancies within a video picture are exploited. The resulting frame is referred to as an I-picture. I-pictures are typically encoded by directly applying the transform to the different macroblocks in a frame. As a consequence, encoded I-pictures are large in size since no temporal information is used as part of the encoding process. In order to increase the efficiency of the intra coding process in H.264, spatial correlation between adjacent macroblocks in a given frame is exploited. The idea is based on the observation that adjacent macroblocks tend to have similar properties. Therefore, as a first step in the encoding process for a given macroblock, one may predict the macroblock of interest from the surrounding macroblocks, typically the ones located on top and to the left of the macroblock of interest, since those macroblocks would have already been encoded. After a prediction block, P, is formed based on previously encoded and reconstructed blocks, it is subtracted from the current block prior to encoding. For the luma samples, P is formed for each 4×4, 8×8 block or for a 16×16 macroblock. There are a total of nine optional prediction modes for each 4×4 luma block, nine modes for each 8x8 block, four modes for a 16×16 luma block and four modes for the chroma components [2].

2.2 Inter Prediction

For all other types of Macroblocks, H.264 uses inter-prediction to compress them. Inter coding uses the temporal model. In this case the predicted frame is created from one or more past or future frames (‘reference frames’). The process of finding the best predicted frame is known as Motion Estimation. The accuracy of the prediction can usually be improved by compensating for motion between the reference frame(s) and the current frame (Motion Compensation). A practical and widely-used method of motion compensation is to compensate for movement of rectangular sections or ‘blocks’ of the current frame. H.264 is a block-based motion-compensated hybrid transform codec, which supports a variety of block sizes (denoted as modes), varying from 16×16, 8×16, 16×8, 8×8, 8×4, 4×8 to 4×4 pixels. Usually the criterion to find the matching block is the energy in the residual formed by subtracting the candidate block from the current M×N block, and the candidate region that minimizes the residual energy is chosen as the best match. However, in order to reduce the computational complexity, most real world application, among them H.264, uses the sum of absolute differences (SAD) [3].

2.3 Sum of Absolute Differences (SAD)

The encoder typically selects the prediction mode, both in Intra and Inter Prediction, for each block that minimises the difference between the predicted block P and the block to be encoded, C. The selection is done by using SAD, also known as L_1 distance between the vectorized form of the two blocks, P and C.

L_1-distance is a positive-definite metric defined over vectors in k-dimensional vector spaces by the corresponding L_1 norm of the difference vector, as follows:

Let $\underline{x}, \underline{y} \in R^k$. Then,

$$L_1(\underline{x},\underline{y}) = \sum_{i=1}^{k} | x_i - y_i | \tag{1}$$

It is easy to show the following properties of the L_1 norm

1. $L_1(x,y) = L_1(y,x)$ (symmetric)
2. $L_1(x,y) \geq 0$, with $L_1(x,y) = 0 \Leftrightarrow x=y$ (positive-definite)
3. $L_1(x,y) + L_1(y,z) \geq L_1(x,z)$ (Triangle inequality)

In addition to the L1-norm, there are other metrics used in vector spaces. The well-known Euclidean distance is perhaps the most popular, also known as L2-norm, defined as

$$L_2(\underline{x},\underline{y}) = \left(\sum_{i=1}^{k} | x_i - y_i |^2 \right)^{1/2}$$

It is easy to show that L_2 has the same properties (1)-(3) as L_1.

We can consider the n^th-norm of two vectors as:

$$L_n(\underline{x},\underline{y}) = \left(\sum_{i=1}^{k} | x_i - y_i |^n \right)^{1/n}$$

and show that it has the same properties (1)-(3) above.

The limit case of $n \to \infty$, known as L_∞ can be shown to be

$$L_\infty(\underline{x},\underline{y}) = \lim_{n\to\infty} L_n(\underline{x},\underline{y}) = \lim_{n\to\infty} \left(\sum_{i=1}^{k} | x_i - y_i |^n \right)^{1/n} \Rightarrow$$

$$L_\infty(\underline{x},\underline{y}) = \max_{i \in \{1,\cdots k\}} | x_i - y_i |$$

Clearly, all L_n-norms try to quantify in a single number the amount of difference between two vectors. There is great debate over which norm is the best to use in order to express error in signal processing, and especially audio, image and video [4]. Traditionally, researchers use the L_2-norm as the minimization criterion for improving signal processing and compression. That is the main reason the peak signal-to-noise ratio (PSNR, a logarithmic representation of the L_2-norm) has been used throughout the signal processing literature to express signal quality. On the other hand, SAD, which is the L_1-norm, has been used as the basic computational block to find block matches in video compression, since it does not require the additional complexity of the multiplier needed for L_2-norms. This is a necessary compromise – one of many one needs to make – in order to have a practical implementation of a video encoder.

The spirit of our work is the same: we offer an alternative compromise to the L_1-norm, an alternative that is similar in spirit (but yet distinct) to the L_∞-norm that yields very good results at a significant reduction in computations and power, especially from a hardware point of view.

3. PROPOSED ALGORITHM

In this section we introduce a new technique for approaching the problem of both, Intra and Inter mode decision. The base of this technique is to avoid the stage of addition, which increases significantly the power and delay cost at the hardware level.

For a given 4×4 block, according to equation (1), a total of 16 subtractions and 15 additions are needed in order to produce the SAD for one mode. Therefore, for the nine modes used in the Intra Prediction Mode, for example, we need 144 subtractions and 135 additions. After computing all modes, a comparison between the results decides which mode will be used. In the stage of Intra Prediction we need to find the prediction mode, thus, a qualitative approach may give the same results as a quantitative one, thus, removing the aforementioned computational cost.

Based on the above observation, we propose to compare the differences for the available modes, after calculating the differences among the predicted and the original pixels, instead of adding them. This comparison will conclude to the mode with the most minimum differences. Thus, the addition stage is completely bypassed. It must be noted here that the new algorithm would not have achieved better delay and power results if we had used the standard comparator [5], which has inferior power and delay characteristics when compared to adders. It was through a synergy with the novel implementation described in Section 5 that we were able to achieve these goals with the proposed algorithm.

In order to implement the new approach, we first calculate the absolute difference between the corresponding pixels for each mode. This can be written as:

$$M_{k_{ij}} = |C_{ij} - P_{k_{ij}}| \quad (2)$$

where M, C, P are 4×4 arrays, k indicates the mode (in the case of the Intra Prediction k is in the range of $0 \le k \le 8$), and i, j are the indices specifying the single pixel within the 4×4 array.

Two successive M_k arrays comprise of a pair and a comparison among them is performed. The array with the largest number of minimum values is chosen. In the next step the array chosen from each pair forms a new pair with the array chosen from its successive pair and the same procedure is repeated until we end up with just one pair of arrays. The array chosen by the last comparison is the one which corresponds to the best mode.

The comparison among two arrays indicates the array with the largest number of minimum values in the following way. Let's assume that we have the following function:

$$f_{k_i} = \begin{cases} 1, M_{k_i} \le M_{k_i+1} \\ 0, M_{k_i} > M_{k_i+1} \end{cases} \quad (3)$$

$$F_k = \sum_{i=0}^{15} f_{k_i} \quad (4)$$

where M_k is the array with the differences for mode k and i (with $0 \le i \le 15$) is the number of differences . According to equation (4) we chose M_k if $F_k < F_{k+1}$, otherwise we chose M_{k+1}.

4. SOFTWARE MEASUREMENTS

The proposed mode decision scheme has been integrated with the H.264 JM10.2 [6] codec for performance evaluation. It is compared with the original codec (which uses SAD) of H.264 in terms of the average bits/frame and the average PSNR. The test sequences are the Akiyo, foreman, coastguard and hall QCIF video sequences. All four QCIF video sequences are 176×144 and have framerate 15 fps. The total number of frames used in the simulation is 300 for each sequence, with an I-Frame rate of 15 frames. The motion estimation scheme used is the "Full search" scheme with the search range set to 16 and number of reference frames set to 1.

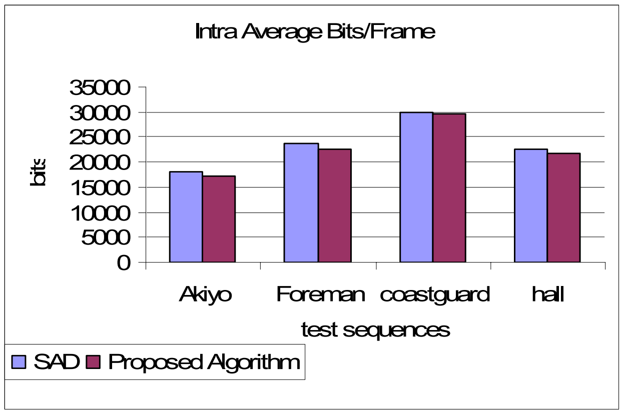

Figure 1. Average bits/frame for Intra Frames

In the simulation process the RD-optimization was set off, which does not allow any rate control. The quantization parameter (QP) was set to QP=28, which is the most common case. Figure 1 shows the average bits/frame of the intra frames in the video sequences produced by the encoder for the four test sequences. The respective average for the inter frames is shown in Figure 2. We see that the proposed scheme produces a small overhead in the average of the bits/frame.

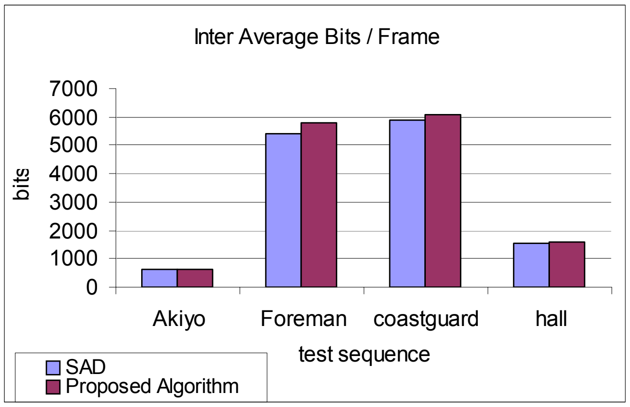

Figure 2. Average bits/frame for Inter Frames

The measurements used in the graphs are shown, with more details, in Table 1. In this Table the percentage difference for each sequence is also presented.

Table 1. Average bits/frame

Test Seq.	Average bits/frame					
	Intra			Inter		
	SAD	*Proposed Arch.*	*% dif.*	*SAD*	*Proposed Arch.*	*% dif.*
akiyo	18107.35	17276.40	-4.6	626.11	637.01	1.7
Foreman	23569.20	22572.80	-4.2	5403.14	5781.42	7.0
coastguard	29989.90	29593.55	-1.3	5888.91	6097.69	3.5
Hall	22439.75	21724.90	-3.2	1521.99	1575.49	3.5

Results show that for the intra frames we have an average decrease of 3.3% in the average bits/frame, and a respective average increase of 3.9% for the inter frames. The performance evaluation process also showed that the average PSNR remains practically the same, as the measurements gave us an average difference of 0.2% which is among the limits of statistical error. The aforementioned results confirm that the proposed algorithm achieves the same efficiency as that of SAD, with a significant reduction in the complexity of the encoder.

5. IMPLEMENTATION

The architecture described in Section 4 is shown in Figure 3, in which the "mode i" blocks refer to the absolute differences for each prediction mode and the "comparator" blocks implement the functions described by (3) and (4) by which the best mode, according to the proposed algorithm, is chosen.

This design is based on the Intra Prediction Mode for 4×4 blocks. That means that the circuit can compare 9 4×4 blocks at a time, as shown in Figure 3. Nevertheless, the circuit can be used as is in the Inter Prediction. For example, with a search range of 16×16 we have a total of 1089 available positions. Using the appropriate number of instances (i.e. 176) of the proposed circuit we can predict the position which gives us the smallest motion vector. The input word-lengths used were 128 bits (for an entire 4x4 block of 8-bit numbers), which is consistent with common video standards.

The comparison is not done by a standard comparator [7] but by specialized bit-wise circuits developed for this application. The comparison between absolute difference i of mode j and absolute difference i of mode k are done by the circuit, the block diagram of which is shown in Figure 4, while the circuit, the block diagram of which is shown in Figure 5 determined which of the two modes has the most minimum values. In case of a tie, the lower order mode is preferred.

The absolute difference selection circuit comprises of 3 stages, as shown in Figure 4. In the first stage we determine which of the 8 bits of the two inputs are equal. As output of this stage we have two 8-bit vectors, the vector *equal* which indicates the bits that are equal among the two input vector and the vector *comp* which is a copy of the second input.

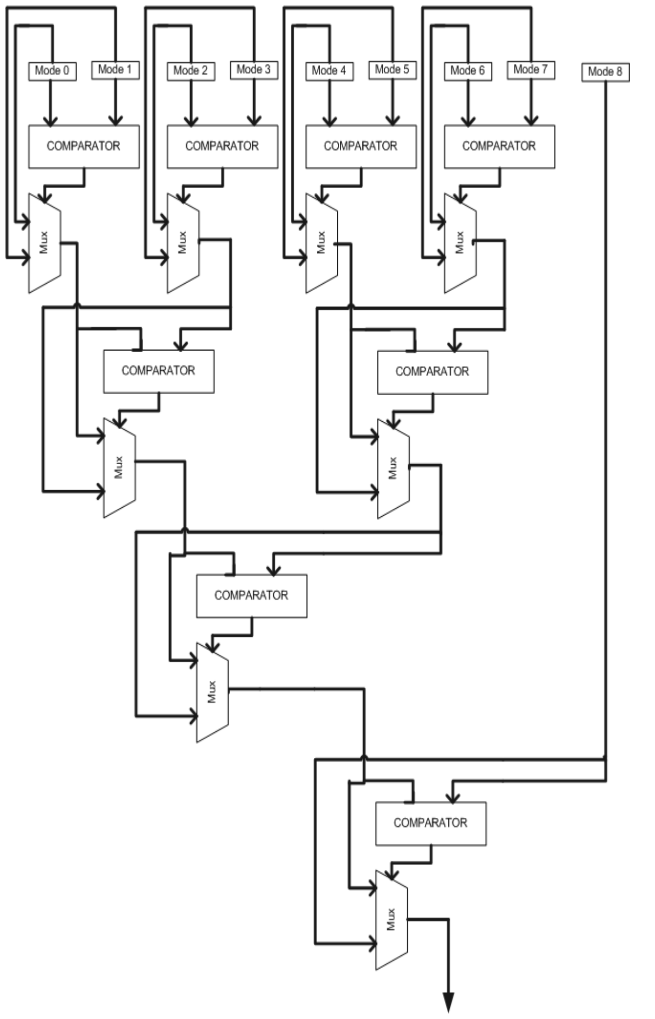

Figure 3. Block diagram of the implemented circuit

The other two stages are those that determine which one of the initial input vectors represents an 8-bit number with greater magnitude. In the second stage we have as inputs the vectors *equal* and *comp*. We part the 8-bit vectors in two triples and one dyad. For each one of these parts we perform a comparison that determines if the respective bits of *comp* are greater or not. For the triples we use a three-bit comparator and for the dyad we use a two-bit comparator.

The three-bit comparator works as follows. If the third bit of *comp* is set to 1 and, at the same time, the third bit of *equal* is set to 0, then the number represented by *comp* is greater than the other. The same applies when the third bit of *equal* and the second bit of *comp* are set to 1 and, at the same time the second bit of *equal* is set to 0, or when the third and second bit of *equal* and the first bit of *comp* are set to 1 and, at the same time the first bit of *equal* is set to 0. In the same sense works the two-bit comparator.

At the end of this stage we have as output 3 bits that show if we have equality and three other bits that show, for each part of the initial comp, if it is greater or not. In the last stage the output of the second stage is input in a three-bit comparator that gives a bit *comp,* that is 1 if the second of the initial vectors is greater and 0 if it is smaller, and a bit *equal* the two initial vectors are equal.

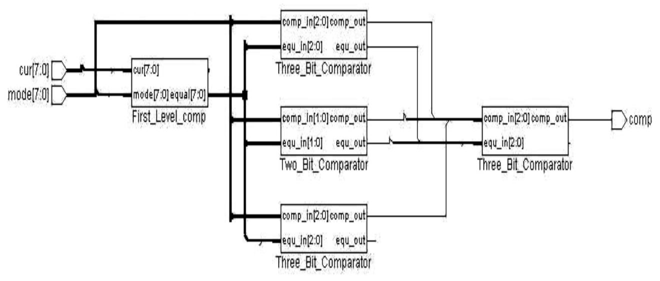

Figure 4. Absolute Difference Selection Circuit – Block Diagram

Each mode has a total of 16 absolute differences of pixel values i.e. 8-bit vectors. The circuit just described gives us a bit for each of these 16 differences. To be accurate it gives us 1 if the difference of the first mode is smaller than or equal to the difference of the second mode. So, at the end of this stage we end up with a 16-bit vector. The next thing to do is to determine which of the two modes has the most minimum values. This is done by the circuit, which's block diagram is shown in Figure 5 This circuit, also, consists of three stages.

In the first stage we part the 16-bit vector in four quadruplets. For each of the quadruplet we count the number of "1". This is done with the circuit named FirstStage in Figure 5. This circuit takes as input a 4-bit vector and has output a 5-bit vector named *sum*. The maximum number of "1" that can be found in the input is four. In this case the bit *sum(4)* is set to one. If we have three "1" in the input vector the bit *sum(3)* is set to 1, if we have 2 the bit *sum(2)* is set to 1, if we have 1 the bit *sum(1)* is set to 1 and finally if there are no "1" in the input vector the bit *sum(0)* is set to 1.

The bit-vectors produced in the first stage are the input to the second stage. In this stage we have two circuits named SecondStage. Each one of them takes as input two 5-bit vectors that represent a value between 4 and 0, and has as output an 8-bit vector. As in the first stage, each bit of the output represents a number of "1". So, if the 8th bit of the output is set to 1, that means that we a have a total of 8 "1", if the 7th bit is set to 1 we have 7 "1" and so on.

Finally we have the third stage, were the inputs are the two 8-bit vectors produced by the previous stage. Here we have the circuit named ThirdStage which has one bit as output. This bit is set to 1 if the total number of "1" (represented by the two 8-bit vectors as described) is greater than or equal to 8.

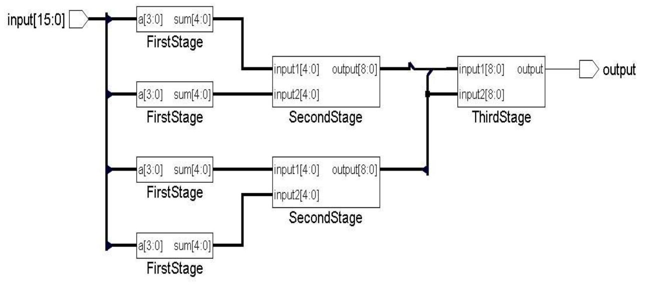

Figure 5. Mode Selection Circuit – Block Diagram

The implementation using SAD and the one using the new algorithm have been captured using VHDL and synthesized to allow comparisons of delay and power on a common reference. They have both been implemented on a 130nm CMOS technology (1.08V) using the UMC standard cell library. The architecture has been simulated using the Mentor Graphics ModelSim® SE 6.0 and synthesized using the Synopsys Design Compiler®. The power estimates for the two circuits were obtained by Synopsys PrimePower® using the four test sequences as inputs.

In order to assess the power-performance characteristics of both implementations we proceeded to form their respective power-delay curves, which are shown in Figure 6. The data points for the power-delay curves were determined as follows:

i. A specific delay target was selected for each implementation, starting from the best delay possible, and relaxing the timing requirement by 1ns at a time.

ii. The circuit was synthesized with Synopsys Design Compiler®, which also provided the delay for each case.

iii. The average power dissipation was calculated by Synopsys PrimePower®.

The operating frequency was set according to the maximum delay of the synthesized circuit, as reported in step (ii).

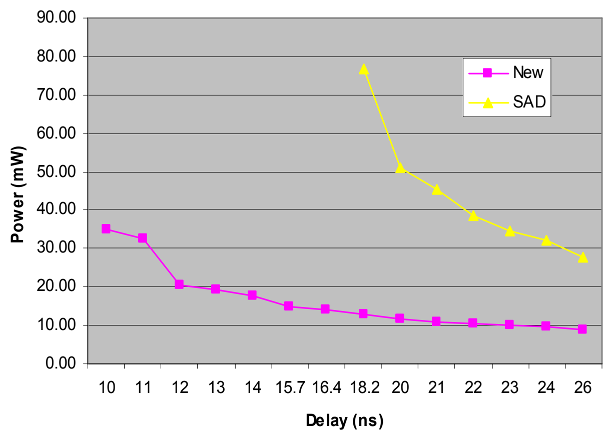

Figure 6. Power – Delay curves for both implementations

From Figure 6 it can be readily concluded that the new algorithm and its implementation outperforms the SAD approach in both power and performance, as its power-delay curve lies below and to the left of the one for SAD.

Table 2. Performance

Architecture	Performance		
	Critical Path (ns)	*Max Clk Frequency (MHz)*	*Cell area*
SAD	18.15	55	132356
Proposed Algorithm	10	100	58100

A more detailed analysis reveals that the design created contains less than 59k gates and can operate at frequencies of up to 100 MHz. The best delay for SAD that Design Compiler® could achieve was 18.15ns vs. 10ns for the proposed implementation, a 45% reduction, whereas the best area for SAD was 132k gates vs. 58k gates for the proposed algorithm, a 56% reduction.

The power reduction that was achieved ranges from 6X at 18.15ns to 3X at 26ns. The observed reduction is due to the reduced complexity of both the prediction algorithm and its circuit implementation. The power reduction at the best delay for both approaches is over 2X in favor of the proposed approach over SAD.

6. CONCLUSIONS

In this paper we presented a new mode selection algorithm for both Intra and Inter prediction modes in H.264 along with logic transformations for its hardware implementation. The proposed replacement of the SAD algorithm with a simple comparison was integrated with the H.264 JM10.1 codec for performance evaluation and was implemented using standard circuit synthesis tools. The resulting circuit has a clear advantage in power (3X-6X), delay (2X), and area (2X) over the one implementing the SAD algorithm, while software measurements show that the proposed algorithm produces an insignificant overhead in the bit rate, and the quality of the produced video sequence is almost the same as that of the one produced by the original H.264 codec.

Therefore, the new algorithm, by reducing the complexity of the computation, can achieve significantly faster execution time and significant reduction in power consumption without having any perceivable impact on the file size and the quality of the final video sequence.

7. REFERENCES

[1] ITU-T Rec. H.264 / ISO/IEC 11496-10, *Advanced Video Coding*, Final Committee Draft, Document JVT-J010d1, December 2003.

[2] Wiegand, T., Sullivan, G., Bjontegaard, G., and Luthra, A. , *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.

[3] Richardson, I., *H.264 and MPEG-4 Video Compression – Video Coding for Next-generation Multimedia*, John Wiley & Sons, 2003.

[4] Toivonen, T., Heikkila, J., *A new rate – minimizing matching criterion and a fast algorithm for block motion estimation*, International Conference on Image Processing (ICIP 2003).

[5] Mano, M., *Digital Design*, 3rd edition.

[6] Joint Video Team (JVT) Reference Software JM10.2, at http://iphome.hhi.de/suehring/tml/download/

[7] Wang, C., Lee, P., Wu, C., and Wu, H., *High Fan-In Dynamic CMOS Comparators with Low Transistor Count*, IEEE Transactions on Circuits and Systems – I:Fundamental theory and applications, vol. 50, no. 9, pp. 1216-1220, September 2003.