TRANSACTIONS ON ELECTRICAL AND ELECTRONIC MATERIALS
Vol. 13, No. 5, pp. 256-261, October 25, 2012

Regular Paper

pISSN: 1229-7607 eISSN: 2092-7592
DOI: http://dx.doi.org/10.4313/TEEM.2012.13.5.256

A Multithreaded Implementation of HEVC Intra Prediction Algorithm for a Photovoltaic Monitoring System

Yungho Choi and Hyungkeun Ahn†
Department of Electrical Engineering, Konkuk University, Seoul 143-701, Korea

Received August 29, 2012; Accepted September 11, 2012

Recently, many photovoltaic systems (PV systems) including solar parks and PV farms have been built to prepare for the post fossil fuel era. To investigate the degradation process of the PV systems and thus, efficiently operate PV systems, there is a need to visually monitor PV systems in the range of infrared ray through the Internet. For efficient visual monitoring, this paper explores a multithreaded implementation of a recently developed HEVC standard whose compression efficiency is almost two times higher than H.264. For an efficient parallel implementation under a mesh-based 64 multicore system, this work takes into account various design choices which can solve potential problems of a two-dimensional interconnects-based 64 multicore system. These problems may have not occurred in a small-scale multicore system based on a simple bus network. Through extensive evaluation, this paper shows that, for an efficient multithreaded implementation of HEVC intra prediction in a mesh-based multicore system, much effort needs to be made to optimize communications among processing cores. Thus, this work provides three design choices regarding communications, i.e., main thread core location, cache home policy, and maximum coding unit size. These design choices are shown to improve the overall parallel performance of the HEVC intra prediction algorithm by up to 42%, achieving a 7 times higher speed-up.

Keywords: 2D interconnects, Multithread, Photovoltaic, Monitoring, Intra prediction

1. INTRODUCTION

A photovoltaic system (PV system) is known to be a promising clean energy. Therefore, many PV systems including solar parks and PV farms have been built to prepare for the post fossil fuel era. To trace the degradation process of the PV systems and thus, to efficiently operate PV systems, there is a need to visually monitor PV systems through the Internet [1-3].

In such visual monitoring systems, video compression technologies enable to store and transmit video data with fewer resources [4]. One of the compression technologies is HEVC (high efficiency video coding), which is a new draft of a video compression technology. To enhance compression efficiency, HEVC provides larger and more variable coding block sizes, adaptive loop filter, larger transform kernel size, etc. Despite its high compression efficiency, due to its high coding complexity and dependencies among tools, it suffers from a long coding time. This problem might be aggravated because each tool of HEVC has been selected based on its single thread performance.

In other words, when the suitability of algorithms for HEVC is examined, their performances are evaluated under an assumption that the single thread performance of processors will be continuously increased and thus, can support their complicated tools. This is not true because the single thread performance of commercial processors has been at a standstill for a while, migrating into multicore processors. Especially, the trend of processor architectures lies in containing more and more processor cores and connecting them together using an interconnection network with high connectivity instead of a bus with low con-

† Author to whom all correspondence should be addressed:
E-mail: hkahn@konkuk.ac.kr

nectivity. However, in terms of behaviors, the architecture having many processing cores and an interconnection network with high connectivity can be significantly different from the one having a small number of cores using a bus. This is because more cores and higher connectivity may cause an unexpected communication behavior including traffic congestion, synchronization competition and so on. Therefore, in implementing a parallelized HEVC for multicore processing systems, it is very important to understand behaviors regarding the interconnection network of a many-core system architecture and to exploit the behaviors efficiently. This is the key motivation of this work.

Generally, there are two ways to parallelize HEVC in multicore systems. One is message passing-based programming, e.g., MPI library, and the other is shared memory programming, e.g., pthread library. Message passing-based programming utilizes messages to enable communications among processing cores. This programming model requires delicacy in partition and a refined communication strategy, taking a considerable time in parallel implementation. In shared memory programming, multiprocessing cores share memory for communications. This programming style is generally easy to parallelize a given algorithm because this programming style is similar to a general sequential programming model except for maintaining critical memory sections, which require atomic operations. Additionally, a shared memory programming model accommodates cache protocol to make a complicated communication process among cores more transparent to programmers. Due to the easiness in programming, therefore, this paper parallelizes HEVC intra prediction algorithms in a shared memory and multithreaded programming model, utilizing a pthread library.

The contributions of this paper are summarized as follows. (1) HEVC intra prediction algorithms are parallelized. (2) The effects of multicore system architectures on the multithreaded performance of the HEVC intra prediction algorithm are analyzed. (3) A good multicore system implementation exploiting the explored architecture characteristics of a multicore system, is provided and its performance is evaluated. For (1), a wavefront scheme [5] is modified to parallelize HEVC intra prediction algorithms, which breaks off dependency chains among coding blocks and enables the execution of an intra-prediction algorithm concurrently in many cores. For (2) and (3), parallelized HEVC intra prediction algorithms are ported into a multicore system consisting of 64 processing cores. Through performance evaluations, this work will provide insight on how multicore system architectures affect the performance of multithreaded HEVC intra prediction algorithms. This insight will be helpful in implementing more effectively parallelized algorithms for HEVC.

The remainder of this paper is organized as follows: Section 2 briefly summarizes intra prediction algorithms to be explored and their parallelization. Section 3 describes how to port intra prediction algorithms into a multicore system for their parallel processing performance evaluations. In Section 4, a performance evaluation on algorithms is given. Lastly, Section 5 will conclude the study.

2. RELATED WORKS

2.1 HEVC intra prediction algorithm

The intra prediction algorithm for HEVC reduces image redundancy by predicting image pixel values based on neighboring pixels previously decoded as shown in Fig. 1(a). Since HEVC provides larger prediction units, HEVC employs more intra prediction modes than previous coding standards such as MPEG2, MPEG4, H.264. As shown in Fig. 1(c), in HEVC, 35 intra prediction modes are defined: intra_DC, intra_planar, intra_fromLuma, and 33 angular predictions [6,7]. The intra_DC uses the average pixel value of neighboring pixels to predict the image pixels for the underlying prediction unit. This works well when an image is a non-moving background image in a picture and does not have strong edges and textures. The intra_planar prediction utilizes interpolations to predict image pixels as shown in Fig. 1(b). This prediction better predicts image textures coming from the right side and the bottom side of the prediction unit. Additionally, HEVC provides 33 different angular direction predictions in Fig. 1(c). This increases the number of angular directions by 4 times compared to H.264, resulting in a better texture prediction. However, despite its higher compression efficiency of HEVC intra predictions, HEVC might suffer from a long coding time due to its complicated intra predictions. Furthermore, because intra predictions utilize neighboring pixels previously decoded, resulting in data dependency chains, the parallelization of HEVC intra predictions might not be easy. To resolve this problem, the next subsection describes the way to remove data dependency chains and thus, to enable the parallelization of intra predictions.

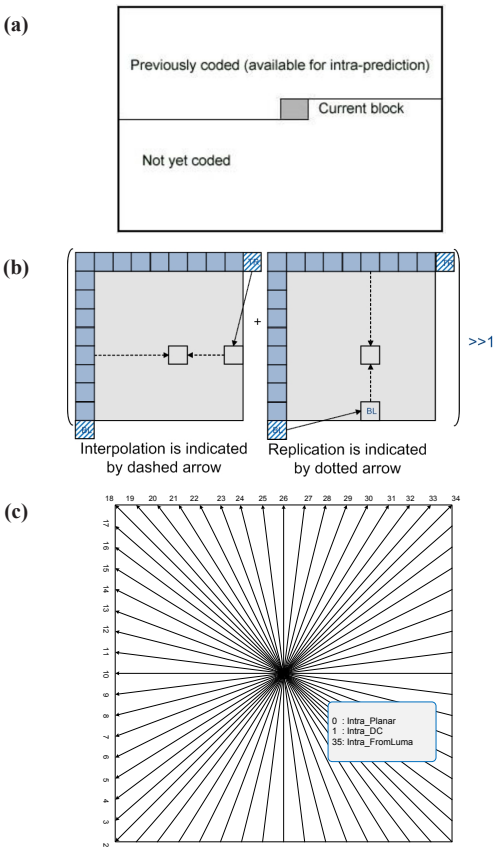

Fig. 1. HEVC intra prediction (a) intra prediction, (b) intra_planar prediction mode, and (c) intra prediction mode directions.

2.2 Parallelization of HEVC intra prediction algorithm

In order to evaluate the parallel performance of each intra prediction algorithm, this work ports intra prediction algorithms into a multicore system consisting of 64 processing cores [9]. As shown in Fig. 2, these processing cores are connected using a mesh network. The multicore system provides on-chip caches, in which local caching is allowed and thus, remote data accesses are expensive. This might decrease the parallel processing per-

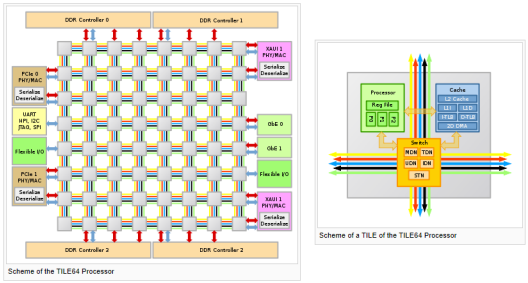

Fig. 2. Mesh interconnects-based 64 multicore system [9].

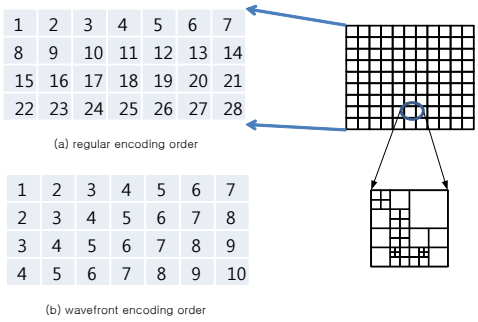

Fig. 3. Wavefront-style parallelization of an intra prediction algorithm.

formance of an intra prediction algorithm by accessing remote data more frequently. Each core in the multicore system has a 3 -way VLIW pipeline for instruction level parallelism. Additionally, this system provides 4 DDR2 controllers which mitigate the bottleneck problem of off-chip memory accesses by increasing off-chip memory bandwidth.

To parallelize intra prediction algorithms, a wavefront-style coding block encoding order [5] is employed. In an intra prediction, spatially neighboring pixels previously encoded are needed to predict macroblock pixel values. However, if coding blocks are encoded in a regular encoding order shown in Fig. 3(a), the coding block level parallelism cannot be achieved. For example, in Fig. 2(a), neighboring coding block 10 and 11 cannot be intra-predicted concurrently because, to predict coding block 11, coding block 10's reconstructed picture image is required, which is available only after coding block 10 is encoded. A wavefront-style coding block encoding order [5] can resolve this problem by changing the encoding order as shown in Fig. 3(b) and thus, enabling a coding block level parallelism. For example, in Fig. 3(b), when coding block-4s are encoded, the reconstructed pictures of coding block-3s are available because they are previously encoded. In the next section, using a multicore system and a wavefront-style coding block encoding order, a multithread performance evaluation of the HEVC intra prediction algorithm is given to provide an insight on how multicore system architectures affect the performance of multithreaded HEVC intra prediction algorithms.

3. PARALLEL IMPLEMENTATION OF AN -INTRA PREDICTION ALGORITHM

To efficiently implement the HEVC intra prediction algorithm, job partitioning for multicores is very important because this partitioning strongly affects communication among processing cores by determining network traffic, cache hit ratio, communi-

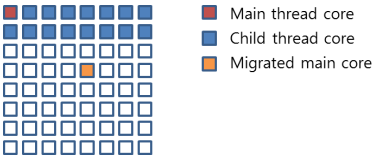

Fig. 4. Thread core configuration.

cation latency and so on. Therefore, this work takes into account the following implementation issues, which are closely associated with the parallel job partitioning task: (1) main thread core location, (2) cache home policy, and (3) maximum coding unit size. Each issue is introduced in the following subsection.

3.1 Main thread core location

The main thread in a parallel program is generally called the thread that starts the given parallel program, assigns incoming data to processing cores, maintains common data structures, forks and synchronizes child threads, and so on. Because the main thread needs to communicate with the child threads frequently, the location of the core executing the main thread, i.e., main thread core, is very important. Figure 4 presents an example of a thread core configuration. In this figure, each small square box represents a processing core and thus, 64 processing cores comprise a multicore system. As shown in the figure, if the main thread core is located in the upmost and the rightmost coordinates, the communication between the main thread core and the child thread cores might suffer since many communication paths between the main thread core and child thread cores are overlapped and thus, can cause heavy traffic congestion. However, if the main thread can be migrated into the orange-color core block in the figure, such communication path overlaps will be minimized, resulting in less congestion and less communication latency. The improved communication performance will increase the overall parallel performance of the HEVC intra prediction algorithm. Consequently, the location of the main thread core needs to be carefully selected so that the average distance between the main thread core and the child thread cores can be minimized in addition to the communication path overlaps also being minimized. This effect of the main thread core location will be evaluated in Section 4.1.

3.2 Cache home policy

The multicore system [9] does not provide a remote cache coherence protocol where remote data cannot be cached locally and therefore, to access remote data, requests must always go to remote home nodes. This significantly hampers parallel performance. Hence, to increase parallel performance, such remote data accesses need to be minimized. For this, this work evaluates two cache configurations: (1) no local cache and (2) local cache. In case of a no local cache configuration, data for child threads are not carefully and locally arranged and therefore, data required by threads are frequently brought in from remote processing cores, increasing traffic and congestion. In contrast, the local cache configuration carefully arranges data so that data frequently accessed by a child thread can be locally homed and cached. This minimizes remote data access and network congestion. This will be evaluated in Section 4.2.

3.3 Maximum coding unit size

The HEVC intra prediction scheme can determine its maxi-

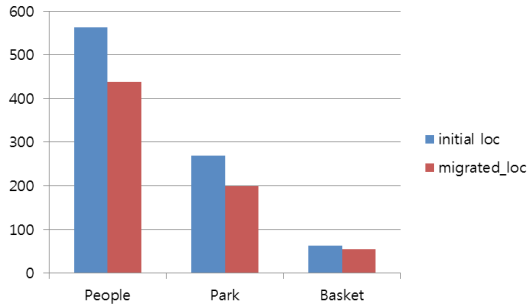

Fig. 5. Execution times for different main thread core locations.

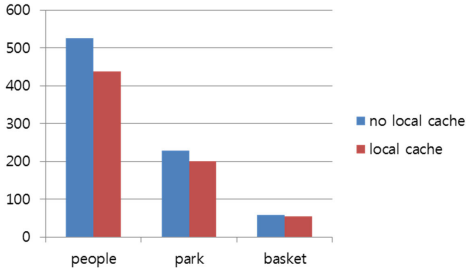

Fig. 6. Execution times for different cache home policies.

mum coding unit size. According to this maximum coding unit size, coding depths and coding parallelisms are defined. For example, assume that the HEVC intra prediction starts coding with a maximum coding unit size (128 pixels × 128 pixels). After completing intra prediction for the coding unit, HEVC intra prediction partitions the coding unit into 4 smaller coding units (64 × 64) for lower level intra predictions. This process is repeated until the coding unit size reaches the minimum coding unit size in order to locate an optimal intra prediction block partitioning configuration, defining coding depths. Generally, plain background images are well coded in a higher maximum coding unit size while fast changing and complicated images need to be coded in a lower maxim coding unit size.

Additionally, this maximum coding unit size determines the coding parallelism in a wavefront-style parallel implementation because parallel processing is done at the level of maximum coding units. In this implementation, smaller maximum coding unit size configurations can exploit higher parallelism because more coding units are concurrently coded as shown in Fig. 3(b). However, in this case, the multicore system might suffer from higher network traffic due to a higher parallelism and more communication messages. This will be evaluated in Section 4.4.

4. PERFROMANCE ANALYSIS OF MULTICORE IMPLEMENTATION

To identify an efficient implementation of parallel HEVC intra prediction schemes, this section evaluates the effects of implementation issues described in Section 3 on the parallel performance of HEVC intra predictions. For this evaluation, the HEVC intra prediction algorithm, TMuC 1.0 [8], is parallelized and executed in [9] under the following conditions: intra-only configuration, 3 video sequences (PeopleOnStreet 2560×1600, ParkScene 1920×1080, BasketballDrill 832×480), minimum coding unit size: 8×8, QP=32, 100 frames encoded. The effect of each implementation issue is evaluated by measuring the speed-up and/or the execution time for each case. The speed-up is obtained by dividing the single-thread execution time of a given case with the corresponding parallel execution time.

4.1 The effect of main thread core location

In this evaluation, two main thread core locations are examined: (1) initial location and (2) migrated location. These locations of the main thread core are shown in Fig. 4. The migrated location of the main thread core is selected by taking into account two things: (1) the average distance between the main thread core and child thread cores and, (2) the overlaps of the paths between the main thread core and child thread cores. Three video sequences, i.e., PeopleOnStreet, ParkScene, BasketballDrill, are encoded using HEVC intra predictions with a configuration of a 64×64 maximum coding unit size. Fig. 5 shows the execution times to encode three video sequences under two main thread core location configurations. As shown in the figure, the migrated location of the main thread core reduces the execution time for intra predictions by up to 26% (PeopleOnStreet). This shows that the careful selection of location in regards to the main thread core can considerably mitigate network traffic and congestion by reducing overlapped communication paths between the main thread core and child thread cores.

Additionally, in the figure, the larger video sequences, e.g., PeopleOnStreet, are shown to benefit more from the migrated main thread core location than the smaller video sequences, e.g., BasketballDrill. This is because in the wavefront-style parallelization, the parallelism of the larger video sequences are higher than that of the smaller ones and thus, the higher parallelism requiring more processing cores can have more communications among cores, causing higher network congestions and requiring a careful choice of a main thread core location. For example, the maximum parallelism in PeopleOnStreet is 25 and the maximum parallelism in BasketballDrill is 8 when the maximum coding size is 64. In this case, PeopleOnStreet requires 25 processing cores for its intra predictions while BasketballDrill needs only 8 processing cores, where PeopleOnStreet might suffer higher communication demands among cores and thus, needs to select the main thread core location more carefully.

4.2 Cache home policy

This section evaluates two cache policies described in Section 3.2, i.e., no local cache and local cache. For the evaluation, three video sequences are encoded by executing wavefront-style parallel HEVC intra predictions. A 64×64 maximum coding unit size is assumed. Fig. 6 shows execution times for encoding three video sequences in two cache home policies. As shown in the figure, the performance of a local home policy is higher than the one of no local home policy (up to 19% faster encoding speed). This is because, in a local home cache policy, most data requests can be satisfied by local cache while, in the no local home cache policy, most data requests always need go to their remote home, causing a long communication time. This is also verified in Fig. 7, which shows the speed-up of two different local home cache policies. In this figure, the local home cache policy achieves a higher speed-up of up to 22% compared to the no-local home cache policy.

Additionally, in Fig. 6, the effect of the home cache policy on performance is higher in larger images than in smaller images. This is because, in a small image, the number of requested processing nodes is small, due to its lower parallelism and thus, the average distance to remote data home nodes and its associated accessing time are small, mitigating the benefit of the local home cache policy. However, considering HEVC is developed for larger

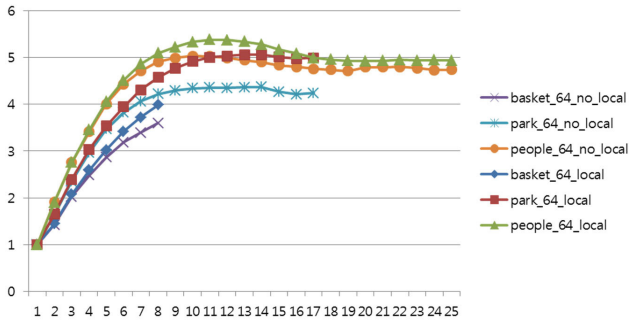

Fig. 7. Speed-ups for different cache home policies.

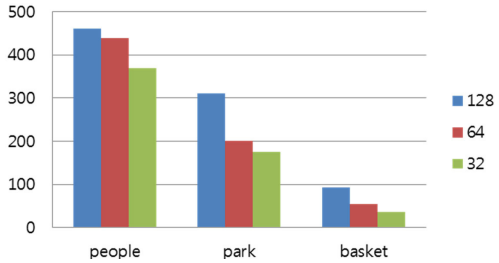

Fig. 8. Execution times for different maximum coding unit sizes.

video images, in most cases, the cache home policy needs to be taken into account for more efficient implementations.

4.3 Maximum coding unit size

As described in Section 3.3, a maximum coding unit size directly affects the parallelism of intra predictions in a wavefront-style parallel implementation. To evaluate the effect of maximum coding unit size on parallel performance, this section measures the parallel performance of three different maximum coding unit sizes, i.e., 128, 64, 32. The maximum parallelisms of these three coding unit sizes are 8, 25, and 51, respectively. Fig. 8 and 9(a~c) present the execution times and the speed-ups of three maximum coding unit sizes and three video sequences.

As shown in Fig. 8, smaller maximum coding unit sizes demonstrate higher parallel performance because smaller coding unit sizes allow a higher parallelism. Especially, the performance difference between 128 and 64 maximum coding unit sizes is higher than the one between 64 and 32. This is because too much parallelism requires a higher number of processing nodes, requiring a higher communication demand and suffering from long communication latencies. Furthermore, limiting the maximum coding unit size for a higher parallelism can hamper the coding efficiency of HEVC by reducing the chance to find a larger coding redundancy. Consequently, a careful determination of parallelism size in a parallel implementation is important not only for speed but also for compression efficiency. This result is verified in Figs. 9(a~c), which show the speed-ups of three maximum coding unit sizes in three video sequences.

4.4 Discussion

For an efficient parallel implementation, in this section, several parallel implementation choices are evaluated. From the provided evaluation results, the effect of main thread core location, cache home policy, and the maximum coding unit size on parallel performance is shown to be significantly large. Main thread core location can affect network traffic by determining

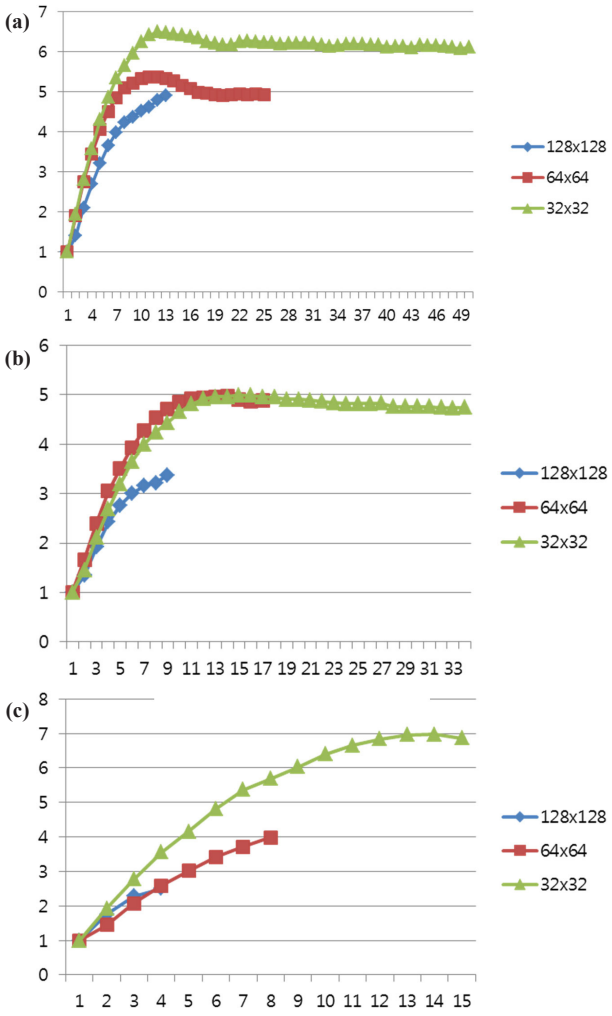

Fig. 9. Speed-ups for three video sequences using three maximum coding unit sizes (a) speed-ups for PeopleOnStreet (2,560×1,600), (b) Speed-ups for ParkScene (1,920×1,080), and (c) speed-ups for BasketballDrill (832×480).

communication path overlaps among processing cores, which significantly affects communication latencies and overall parallel execution times. The cache home policy also affects parallel performance by enabling to locally cache data, thereby significantly reducing communication overheads. Lastly, maximum coding unit sizes are shown to be important in increasing parallel execution speed and compression efficiency. Consequently, this paper investigates the effects of various parallel implementation choices in a mesh interconnection network-based multicore system. Unlike a simple bus-based small-scale multicore system, a more complicated communication activity occurs, affecting parallel performance. Consequently, in order to achieve a higher parallel performance, we need to minimize communication overheads by carefully determining main thread core location, cache home policy, and the maximum coding unit size.

5. CONCLUSION

This work investigates an implementation of a multithreaded HEVC intra prediction for a photovoltaic monitoring system which investigates the degradation process. This investigation evaluates three design choices for parallel intra predictions in ex-

ecution times and speed-ups. Through an extensive evaluation, main thread core location, cache home policy, and maximum coding unit size are shown to significantly affect the parallel performance of intra predictions by determining communication overheads. Carefully chosen design options for these three improves the parallel performance of HEVC intra predictions by up to 42% and increases speed-up by 7 times. This shows that minimizing communication overheads plays a critical role in implementing a multithreaded HEVC intra prediction. In other words, in a mesh-based many-core system, the network connecting processing cores might be a performance bottleneck by being saturated.

REFERENCES

[1] D. Sera, R. Teodorescu, P. Rodriguez, Partial Shadowing Detection based on Equivalent Thermal Voltage Monitoring for PV Module Diagnostics, IECON, 2009: 708-713 [DOI: http://dx.doi.org/10.1109/IECON.2009.5415006].

[2] L. Cristaldi, M. Faifer, M. Rossi, F. Ponci, Monitoring of a PV System: The role of the Panel Model, IEEE International Workshop on Applied Measurements for Power Systems, 2011: 90-95 576 [DOI: http://dx.doi.org/10.1109/AMPS.2011.6090437].

[3] G. Notton, V. Lazarov, L. Stoyanov, Optimal Sizing of a Grid-connected PV System for Various PV Module Technologies and Inclinations, Inverter Efficiency Characteristics and Locations, (Renewable Energy 35, 2010) pp.541-554

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, IEEE Trans. Circuits Syst. Video Technol. 2003 13(7): 560-576 [DOI: http://dx.doi.org/10.1109/TCSVT.2003.815165].

[5] Zhuo Zhao, Ping Liang, IEEE International Symposium on Circuits and Systems, 2006: 4-2672 [DOI: http://dx.doi.org/10.1109/ISCAS 2006.1693173].

[6] K. McCann, W. Han, I. Kim, J. Min, E. Alshina, T. Lee, J. Chen, V. Seregin, S. Lee, Y. Hong, M. Cheon, N. Shlyakhov, Video Coding Technology proposal by Samsung, JCTVC-A124, Dresden, German, Apr.

[7] K. Ugur, R. Andersson, A. Fuldseth, Video Coding Technology Proposal by Tandberg, Nokia and Ericsson, JCTVC-A119, Dresden, German, Apr.

[8] HEVC Reference Software TMuC1.0 [Online], Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/HM-1.0.

[9] B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, Liewei Bao, J. Brown, M. Mattina, Miao Chyi-Chang, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, J. Zook, Tile64 – Processor: A 64-Core SoC with Mesh Interconnect, ISSCC 2009: 88-598 [DOI: http://dx.doi.org/10.1109/ISSCC2008.4523070].