Parallel Deblocking Filter for H.264/AVC on the TILERA Many-Core Systems

Chenggang Yan[1,2], Feng Dai[1], and Yongdong Zhang[1]

[1] Multimedia Computing Group, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2] Graduate University of Chinese Academy of Sciences, Beijing
{yanchenggang,fdai,zhyd}@ict.ac.cn

Abstract. For the purpose of accelerating deblocking filter, which accounts for a significant percentage of H.264/AVC decoding time, some studies use wavefront method to achieve the required performance on multi-core platforms. We study the problem under the context of many-core systems and present a new method to exploit the implicit parallelism. We apply our implementation to the deblocking filter of the H.264/AVC reference software JM15.1 on a 64-core TILERA and achieve more than eleven times speedup for 1280*720(HD) videos. Meanwhile the proposed method achieves an overall decoding speedup of 140% for the HD videos. Compared to the wavefront method, we also have a significant speedup 200% for 720*576(SD) videos.

Keywords: Video decoding; H.264/AVC; Deblocking filter; Parallel algorithm; Many-core systems.

1 Introduction

H.264/AVC is the newest coding standard which brings a high efficient video compression method to the multimedia industry [1]. Due to its compression efficiency, it has been applied to several important applications including video telephony, video storage, broadcast, video streaming, HD-DVD. An in-loop deblocking filter has been adopted by H.264/AVC, though provides 10–15% bit rate saving, brings heavy computation [2]. For high definition videos, Chen et al. [3] shows that the deblocking filter contributes 38% computation time. Therefore, it is important to improve the H.264/AVC deblocking performance.

With the development of IC technologies, multi-core processors are commonplace and many researchers have tried to use multi-core platforms to implement parallelization for H.264/AVC decoder [4-6]. The wavefront method is a commonly used MB-level data partition [6,7]. As the number of cores per chip increases, we are entering into many-core ages such as TILERA many-core systems. It is not easy to develop an efficient data partition schedule for deblocking filter that can efficiently utilize so many cores. In order to solve such problem, we review the deblocking filter within MBs. We study the data dependencies in low level and change the edge filtering orders. It leads to a significant change of data dependencies between neighboring MBs, while the final results are not changed. Based on this modification we can divide the deblocking filter into more tasks executed on many-core processors.

K.-T. Lee et al. (Eds.): MMM 2011, Part I, LNCS 6523, pp. 51–61, 2011.
© Springer-Verlag Berlin Heidelberg 2011

The remainder of this paper is organized as follows. First, we briefly introduce the TILERA Many-core systems in Section 2. An overview of H.264/AVC deblocking filter and the wavefront method are described in Section 3. The proposed deblocking filter speedup techniques are presented in Section 4. Experimental results and analysis are given in Section 5. Finally, we conclude this letter in Section 6.

2 Short Overview of TILERA Many-Core Systems

Centralized monolithic processor designs, such as single core and shared bus structures, are not scaling, leading to multi-core design becoming the norm [8-10]. Research highlights the benefits of mesh networks connecting these cores [11,12], which can solve the multi-core scalability problem.

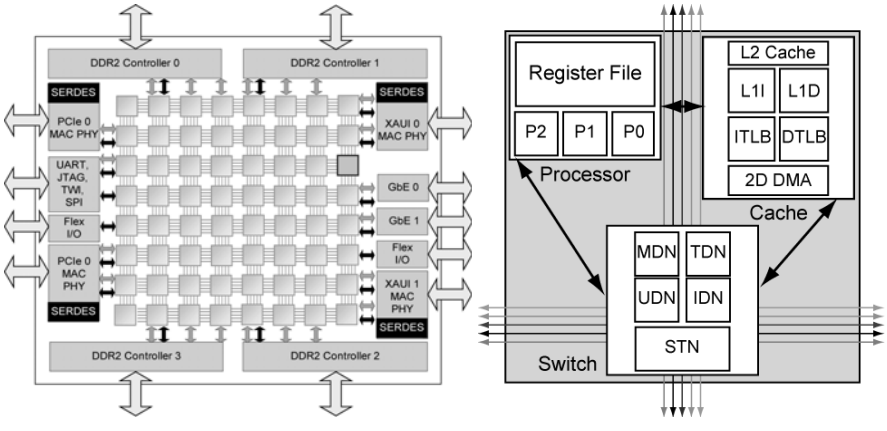

Fig. 1. 64-core TILERA block diagram and Single tile block diagram figure (a) 64-core TILERA (b) single tile

TILERA many-core processor family features devices with 16 to 100 identical processor cores. Fig.1a illustrates a block diagram with 64-core TILERA arranged in an 8×8 array. These cores run at 700MHz and connect through a scalable and high-speed 2D mesh network. Using four 72-bit 800MHz DDR2 interfaces, the chip peak memory bandwidth is over 25GB/s. Two PCI-e ×4 interfaces, two XAUI interfaces and two RGMII interfaces provide over 40Gb/s of I/O bandwidth. The low-speed interfaces provide seamless integration with a variety of systems [13].

Each tile processor (Fig.1b) is a full featured processor and provides a three-way Very Long Instruction Word (VLIW) architecture allowing up to 3 instructions per cycle. A single core contains a 16KB L1 cache, 8KB of data L1 cache, and 16KB of combined L2 cache. Each core also has its own DMA engine and Translation Lookaside Buffer (TLB) which allow memory virtualization and the ability to run a modern O/S on each core [14].

3 Deblocking Filter and the Wavefront Method

H.264/AVC video coding standard adopts an in-loop deblocking filter which removes block-edge artifacts that are produced by block transformation and block motion compensation. “In-loop” implies any inappropriate modification of the in-loop deblocking filter causes serious error propagation to succeeding frames.

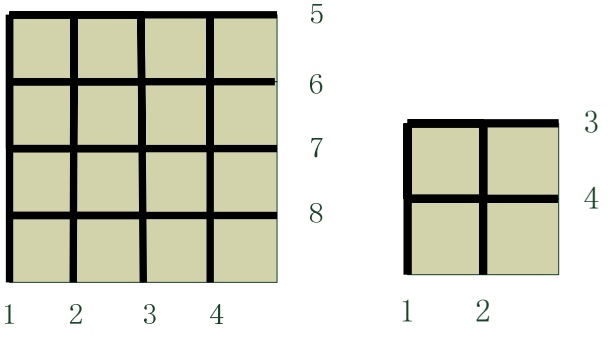

Fig. 2. Edge filtering order in each MB (a) 16x16 Luminance data (b) 8x8 Chrominance data

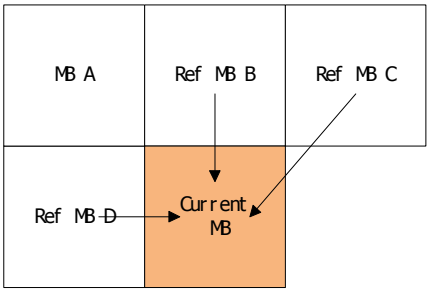

Fig. 3. Data dependencies between neighboring MBs in deblocking filter

The deblocking filter stage is applied to each luminance and chrominance edge within one MB (Fig. 2). All edges are processed from left to right and from top to bottom. Vertical boundary edges are processed first. Usually MBs in a frame are processed in scan order during deblocking filter which should be followed to ensure both the encoder and the decoder having the same result. The current deblocking MB has dependencies on its adjacent left, upper, and upper-right MBs (Fig. 3). When deblocking the current MB, the left, upper, and upper-right MB should have completed processed.

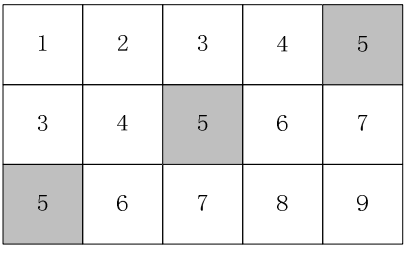

Fig. 4. A wavefront data partition example: each rectangular indicts an MB

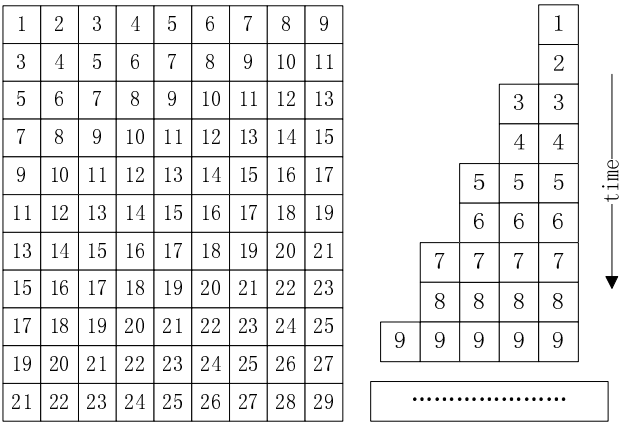

Fig. 5. An example of wavefront MB-Level deblocking filter for a video frame

The wavefront method is widely used for data partition [7] which processes the data in wavefront order(Fig.4). The MBs are processed according to their numbers which indicate the time stamp. MBs with the same numbers are processed concurrently. Therefore, some researchers adopt this method to parallel deblocking filter. In this paper we call it wavefront MB-level deblocking filter method and wavefront method for short. Fig.5 shows an example of wavefront method for a video frame. At time stamp 4, there are two independent MBs. The maximum number of independent MBs(MNIM) is first available at time stamp 9 and it depends on the resolution. Table1 states the MNIM using the wavefront method. The column "MBs" indicates the horizontal and vertical numbers of MBs in a frame.

There are some disadvantages for this kind of MB-level parallelism. The first disadvantage is that there are few numbers of independent MBs at the start of deblocking filter. Fig.5 shows that the MNIM increases one at two stamps addition. What's more, when the number of cores is enough, this method has not discovered the potential capacity of the MNIM. As shown before the MNIM increases with the resolution of the frame. Considering the number of cores, we can calculate the maximum parallelism as follows:

$$MP_{wavefront} = \begin{cases} \min(ceil(\frac{W}{2}), H) & \text{if} \quad \min(ceil(\frac{W}{2}), H) < C \\ C & \text{if} \quad \min(ceil(\frac{W}{2}), H) >= C \end{cases} \tag{1}$$

Where ceil function returns the value of a number rounded upwards to the nearest integer, min function returns smaller value of the two integers, H and W indicates the horizontal and vertical numbers of MB in a frame, C indicates the number of cores used for deblocking filter.

Table 1. The MNIM using the wavefront method

Format	Resolution	MBs	MNIM
QCIF	176x144	11x9	5
CIF	352x288	22x18	9
SD	720x576	45x36	18
HD	1280x720	80x45	23
FHD	1920x1088	120x68	34

Another drawback of the wavefront method is that we should not ignore the communication between the MBs. When the current MB is processed, it has to communicate with up to three MBs processed on other cores. The amount communication overhead incurred in wavefront method is approximately 3*W*H. Assuming the deblocking time of each MB is identical and the number of cores is enough. We can get the speedup of data parallelism in wavefront method as follows:

$$SP_{wavefront} = \frac{H * W}{H + 2 * (W - 1) + 3 * k * W * H} \tag{2}$$

Where k indicates the time it takes to communicate between two MBs.

4 The Proposed MB-Level Deblocking Filter

As described before, the order of deblocking filter should not change arbitrarily which can influence the results of H.264/AVC decoder. MBs can be processed out of scan order provided these dependencies are satisfied. The wavefront method studies the dependencies in MB-Level. In this section we study it in low level.

Fig. 6 shows the data dependencies between the current luminance MB and edges in neighboring MBs. As shown in Fig. 6, the four left neighbour 4x4 blocks of current MB are named L1–L4; the four upper neighbour 4x4 blocks of current MB are named T1–T4; V1-V7 are the vertical edges; H1-H6 are the horizontal edges; V1, V5, V7, H1 and H5 are edges between the MBs. Current MB can be processed when the neighbor blocks L1-L4 and T1-T4 don't change.

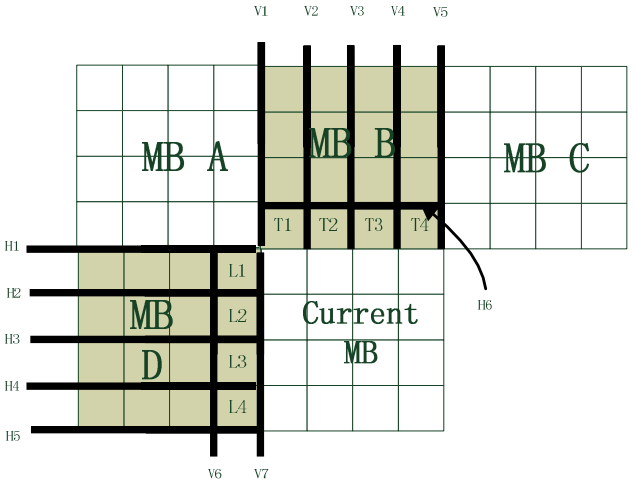

Fig. 6. Data dependencies between the current Luminance MB and neighboring edge filters in deblocking filter

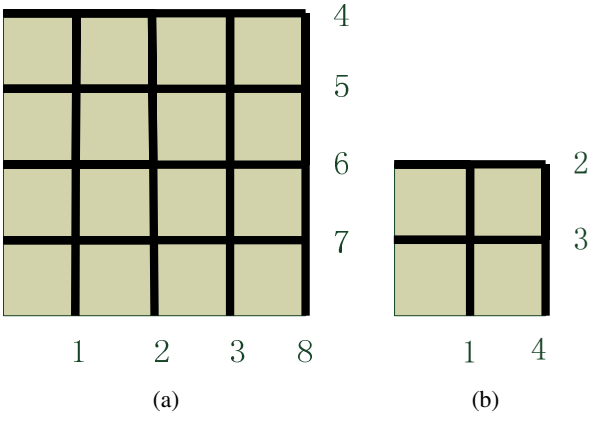

Fig. 7. Edge filtering order in our proposed method (a) 16x16 Luminance data (b) 8x8 Chrominance data

For edge filters, at most four pixels on either side of each edge are evaluated. Up to three pixels on either side of the edge may be influenced[1]. So when the V1-V5 and H6 have the conduction of deblocking filter, the T1-T4 will not change; when the H1-H4 and V6 are processed, the L1-L4 will not alter, as well. Though H5 could change the result of L4, it happens after the deblocking filter of current MB based on the global deblocking order for MBs. Therefore, it will not influence the current MB. MB C has impact on the current MB only through the vertical edge V5. If the vertical edge V5 belong to the deblocking filter of MB B, the MB C is irrelevant to current MB. Meanwhile V1 belongs to MB A and V7 belongs to MB D by this rule. In this

condition, it seems that the current MB has an additional relevance MB A. But when the MB B and MB C are processed, the MB A must have been ready. So the current MB has nothing to do with the MB A.

Based on this observation we proposed a new MB-Level deblocking filter. First of all, we changes the edge filtering order within an MB(Fig. 7) but not the overall edge filtering order in one frame. The data dependencies between neighboring MBs in deblocking filter decreases (Fig. 8). The current deblocking MB only has dependencies on its adjacent left and upper MBs. Assuming every MB adopts the edge filtering order we proposed, we can have a new method of data partition. Fig. 9 shows an example of the proposed data partition for a video frame. At time stamp 4, there are four independent MBs. The MNIM depends on the resolution as the wavefront method. Table2 states the MNIM using our proposed MB-level deblocking filter.

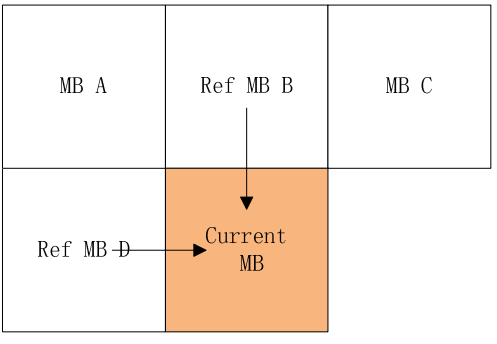

Fig. 8. Data dependencies between neighboring MBs in our proposed method

Table 2. Maximum number of independent MBs using our proposed MB-level deblocking

Format	Resolution	MBs	MNIM
QCIF	176x144	11x9	9
CIF	352x288	22x18	18
SD	720x576	45x36	36
HD	1280x720	80x45	45
FHD	1920x1088	120x68	68

Compared with the wavefront method, there are some advantages for this kind of MB-level parallelism. The first advantage is that there are more numbers of independent MBs at the start of deblocking filter. Fig. 9 shows that the MNIM increases one at one stamp addition. What's more, when the number of cores is enough, the MNIM is bigger (Table 2). Considering the number of cores, we can calculate the maximum parallelism as follows:

$$MP_{new} = \begin{cases} \min(W,H) & \text{if} \quad \min(W,H) < C \\ C & \text{if} \quad \min(W,H) >= C \end{cases} \tag{3}$$

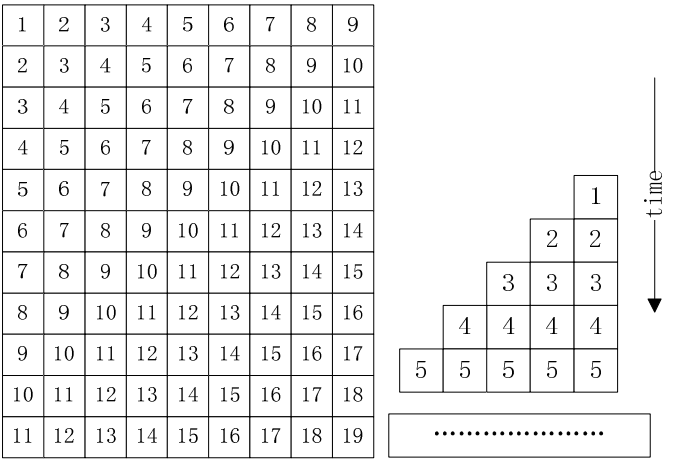

Fig. 9. An example of our proposed MB-Level deblocking filter for a video frame

Obviously equation (1) and (3) shows that with enough cores, the maximum parallelism of our method is better. Further each MB in our method to be processed has at most two connections with others. The amount of communication overhead incurred in our proposed MB-level deblocking filter is 2*W*H approximately which is less than the wavefront method. When the number of cores is enough, we can also get the speedup of data parallelism in our proposed method as follows:

$$SP_{proposed} = \frac{H * W}{H + W - 1 + 2 * k * W * H} \tag{4}$$

Where k indicates the time it takes to communicate between two MBs.

From equation (2) and (4), we find that the theoretical speedup in our method is better than that in wavefront method. In the next section, we will compare them from actual experiments.

5 Experimental Results

In this section, we compare our scheme on deblocking filter with the wavefront method. To compare the two methods, we adopted a decoder migrated from H.264/AVC reference software JM15.1 without any optimization as baseline profile. The decoder includes the stages of entropy decoding, de-quantization, inverse integer transform, intra prediction and motion compensation. We implemented the parallel algorithm on a 64-core TILERA which was described in section 2.

The input videos in our experiments contain a list of standard test sequences named *mobile, highway, hall, mother-daughter, riverbed ,rush_hour pedestrian, container and blue_sky* with four resolution levels, 1280x720(HD), 720x576(SD), 352x288(CIF) and 176x144(QCIF), which are encoded by the reference software JM15.1. The speedup of deblocking filter which is gained by our proposed method compared to the wavefront method can be measured by:

$$Speedup = \frac{Wavefront(ms)}{Proposed(ms)} \tag{5}$$

Table 3. The execution time between wavefront MB-level deblocking filter and our proposed method

sequences	Format	JM15.1(ms)	Wavefront method(ms)	Proposed method(ms)	Speedup
blue_sky	SD	268.8	49.7	23.8	2.09
blue_sky	HD	596.5	76.5	42.7	1.79
pedestrian	SD	225.1	45.3	22.9	1.98
pedestrian	HD	526.7	75.3	43.0	1.75
riverbed	SD	168.3	35.4	19.4	1.82
riverbed	HD	382.9	69.2	40.6	1.70
rush_hour	SD	212.4	46.7	22.7	2.06
rush_hour	HD	508.5	76.7	43.9	1.75
container	QCIF	15.8	8.6	4.2	2.05
container	CIF	55.5	18.6	9.3	2.00
hall	QCIF	15.3	8.4	4.2	2.00
hall	CIF	61.7	19.5	9.6	2.03
highway	QCIF	13.4	7.5	3.8	1.97
highway	CIF	58.9	18.6	9.2	2.02
mobile	QCIF	11.1	6.6	3.2	2.06
mobile	CIF	46.0	16.7	8.2	2.04
mother-daughter	QCIF	14.4	6.9	3.9	1.77
mother-daughter	CIF	60.6	19.4	9.6	2.02

Table 4. The average execution time between wavefront MB-level deblocking filter and our proposed method

Format	JM15.1(ms)	Wavefront method(ms)	Proposed method(ms)	Speedup
QCIF	14.0	7.6	6.3	1.21
CIF	56.5	18.6	12.4	1.50
SD	218.7	44.3	22.2	2.00
HD	503.7	74.4	42.6	1.75

Table 3 compares the performance of deblocking filter between wavefront method and our proposed method. The average execution time is summarized in Table 4. Results demonstrate that the proposed scheme can achieve more than eleven times speedup for HD videos compared to the JM15.1 software. We also can have a speedup of 200% for SD videos compared to the wavefront method.

Table 5 shows the overall decoding speedup compared with the JM15.1 software. From Table 6, we find that the proposed method achieves better speedup as the resolution of videos increases. The proposed method achieves a speedup of 140% for the HD videos.

Table 5. The overall decoding speedup

sequences	Format	JM15.1(ms)	Proposed method(ms)	Speedup
blue_sky	SD	777	560	1.39
blue_sky	HD	1621	1131	1.43
pedestrian	SD	651	477	1.36
pedestrian	HD	1396	949	1.47
riverbed	SD	687	542	1.27
riverbed	HD	1449	1143	1.27
rush_hour	SD	629	468	1.34
rush_hour	HD	1369	941	1.45
container	QCIF	69	60	1.15
container	CIF	203	170	1.19
hall	QCIF	67	59	1.14
hall	CIF	201	163	1.23
highway	QCIF	58	51	1.14
highway	CIF	177	141	1.26
mobile	QCIF	85	79	1.08
mobile	CIF	258	224	1.15
mother-daughter	QCIF	63	55	1.15
mother-daughter	CIF	184	146	1.26

Table 6. The average overall decoding speedup

Format	JM15.1(ms)	Proposed method(ms)	Speedup
QCIF	68.4	60.8	1.13
CIF	204.6	168.8	1.21
SD	686.0	511.8	1.34
HD	1458.8	1041.0	1.40

6 Conclusion

The deblocking filter is considered as the most computationally expensive part of the H.264/AVC decoder. A new MB-level parallelism is proposed to speedup deblcoking filter which is better than the widely used MB-level data partition schedule named wavefront method. The proposed scheme was shown to reduce the deblocking computational load by more than eleven times for HD videos in comparison with the reference software JM15.1. It also has a speedup of 200% for SD videos compared to the wavefront method. The overall decoding time is reduced significantly, as well.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (60802028, 60873165), National Basic Research Program of China (973Program, 2007CB311100), Co-building Program of Beijing Municipal Education Commission, Beijing New Star Project on Science & Technology (2007B071).

References

1. Joint Video Team of ITU-T and ISO/IEC JTC1. Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050 (2003)
2. List, P., Joch, A., Lainema, J., Bjntegaard, G., Karczewicz, M.: Adaptive deblocking filter. IEEE Transactions on Circuits and Systems for Video Technology 13(7), 614–619 (2003)
3. Chen, T.C., Fang, H.C., Lian, C.J., Tsai, C.H., Huang, Y.W., Chen, T.W., et al.: Algorithm analysis and architecture design for HDTV applications-a look at the H. 264/AVC video compressor system. IEEE Transactions on Circuits and Devices Magazine 22(3), 22–31 (2003)
4. Zhao, Z., Liang, P.: Data partition for wavefront parallelization of H.264 video encoder. In: IEEE International Symposium on Circuits and Systems, ISCAS 2006, pp. 21–24 (2006)
5. Lee, J.-Y., Lee, J.-J., Park, S.M.: Multi-core platform for an efficient H.264 and VC-1 video decoding based on macroblock row-level parallelism. IET Circuits, Devices & Systems (2010)
6. Meenderinck, C., Azevedo, A., Alvarez, M., Juurlink, B., Mesa, M.A., Ramirez, A.: Parallel Scalability of Video Decoders. Delft University of Technology (2008)
7. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: principles, techniques, and tools. Addison-Wesley Longman, Boston (2007)
8. Friedrich, J., McCredie, B., James, N., et al.: Design of the Power6TM Microprocessor. ISSCC Dig. Tech. Papers, pp. 96–97 (2007)
9. Dorsey, J., Searles, S., Ciraula, M., et al.: An Integrated Quad-CoreTM Opteron Processor. ISSCC Dig. Tech. Papers, pp. 102–103 (2007)
10. Nawathe, U., Hassan, M., Warriner, L., et al.: An 8-Core 64-Thread 65b Power-Efficient SPARC SoC. ISSCC Dig. Tech. Papers, pp. 108–109 (2007)
11. Taylor, M., Kim, J., Miller, J., et al.: A 16-Issue Multiple-Program-Counter Microprocessor with Point-to-Point Scalar Operand Network. ISSCC Dig. Tech. Papers, pp. 170–171 (2003)
12. Vangal, S., et al.: An 80-tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. ISSCC Dig. Tech. Papers, p. 98 (2007)
13. Agarwal, A., Bao, L., Brown, J., et al.: Tile Processor: Embedded Multicore for Networking and Digital Multimedia. Hot Chips (2007)
14. Bell, S., Edwards, B., Amann, J., et al.: TILE64-Processor: A 64-Core SoC with Mesh. In: Interconnect Solid-State Circuits Conference (2008)