Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2008, Article ID 930250, 12 pages
doi:10.1155/2008/930250

Research Article

High Speed 3D Tomography on CPU, GPU, and FPGA

Nicolas GAC,[1, 2] Stéphane Mancini,[1] Michel Desvignes,[1] and Dominique Houzet[1]

[1] *Grenoble-Images-Parole-Signal-Automatique Laboratoire (GIPSA-lab), Grenoble Institute of Technology (INPG), BP 46, 38402 Grenoble Cedex, France*
[2] *Equipes Traitement des Images et du Signal (ETIS), Centre National de la Recherche Scientifique (CNRS), Ecole Nationale Supérieure de l'Electronique et de ses Applications (ENSEA), Université de Cergy-Pontoise, 95000 Cergy-Pontoise Cedex, France*

Correspondence should be addressed to Nicolas GAC, nicolas.gac@lss.supelec.fr

Received 1 March 2008; Revised 24 June 2008; Accepted 12 November 2008

Recommended by Dragomir Milojevic

Back-projection (BP) is a costly computational step in tomography image reconstruction such as positron emission tomography (PET). To reduce the computation time, this paper presents a pipelined, prefetch, and parallelized architecture for PET BP (3PA-PET). The key feature of this architecture is its original memory access strategy, masking the high latency of the external memory. Indeed, the pattern of the memory references to the data acquired hinders the processing unit. The memory access bottleneck is overcome by an efficient use of the intrinsic temporal and spatial locality of the BP algorithm. A loop reordering allows an efficient use of general purpose processor's caches, for software implementation, as well as the 3D predictive and adaptive cache (3D-AP cache), when considering hardware implementations. Parallel hardware pipelines are also efficient thanks to a hierarchical 3D-AP cache: each pipeline performs a memory reference in about one clock cycle to reach a computational throughput close to 100%. The 3PA-PET architecture is prototyped on a system on programmable chip (SoPC) to validate the system and to measure its expected performances. Time performances are compared with a desktop PC, a workstation, and a graphic processor unit (GPU).

1. INTRODUCTION

Tomography consists of reconstructing an object from its projections via numerical methods [1]. This process is used in medical scanners, such as computed tomography (CT) or positron emission tomography (PET) scanners. PET is a nuclear imaging modality; its goal is to measure the spatial and temporal distribution of a radiotracer perfused in a patient's body. PET imaging is used in oncology, to detect, track, and visualize tumors. After data acquisition, the 3D image of the radiotracer is reconstructed offline from the measures (called sinograms) to diagnose pathologies. Oncology and other clinical applications need a high-quality reconstruction as fast as possible (few minutes at most) to reduce the device occupation and allow a patient repositioning. (A patient cannot experience a radiotracer twice in a short while and has to wait several months before a new examination, in case of bad camera positioning.) Also, dynamic PET is in need of even faster reconstruction.

Moreover, tomography is required in many other medical imaging techniques, such as 3D magnetic resonance imaging and 3D ultrasound imaging, or in other domains such as synthetic aperture radar (SAR), contactless control, and industrial X-ray applications. Therefore, the acceleration of the reconstruction algorithm is of great interest for various applications.

Due to the large amount of the acquired data and the complexity of the algorithms, reconstruction is a very time-consuming process. From a computing point of view, reconstruction methods can be classified into two main techniques: analytic (direct) reconstruction and iterative reconstruction. They both include a back-projection (BP) step that accounts for 50% to 70% of the processing time.

In 3D reconstruction, the computational complexity of the standard algorithm to reconstruct an N^3 dataset from N angles of projection is $O(N^4)$. In the previous decade, several algorithms have been proposed to reduce BP complexity. The lowest cost obtained is $O(N^3 \log N)$ but generally with a lower quality of reconstruction; also it does not take into account some required data management, which delay the process. Although CPUs have gained sufficient computing power for 2D reconstruction, with 3D reconstruction, the increase of the amount of data for high-quality images leads to higher computing times. Iterative reconstruction algorithms may reach several hours of processing [2].

The algorithmic optimizations of reconstruction have reached some limits and it is becoming mandatory to reduce the computing time through architecture solutions. General purpose parallel computers benefit from recent competing technologies: the system on programmable chip (SoPC) and the general purpose graphical processing unit (GP-GPU).

This paper shows that a hardware implementation of the BP algorithm needs to overcome the memory bottleneck. This may be solved both by a loop reordering and the use of an efficient caching mechanism. Parallel hardware pipelines can be fed with a hierarchy of semigeneral purpose cache such as the 3D-AP cache [3]. The resulting architecture makes a better use of memory bandwidth than general purpose CPUs and GP-GPU.

The first parts of this paper present the use of the 3D BP algorithm and different solutions to accelerate it. Next, we present the memory bottleneck of a classical implementation of 3D BP to overcome. From this study, an efficient architecture is proposed: the pipelined, prefetch, and parallelized architecture for 3D PET BP (3PA-PET). The quality, complexity, and timing performance of the 3PA-PET architecture are also presented. Measures on its prototyping on a SoPC allow a comparison with the implementation of BP on CPU and GP-GPU.

2. 3D BP IN TOMOGRAPHY RECONSTRUCTION

In this section, we will first show that 3D PET BP and the 3D CT BP using, respectively, a parallel and a cone beam geometry are close algorithms. Then, we will present some related works on acceleration of these two BPs on several architectures.

2.1. BP algorithms

2.1.1. 3D parallel beam BP for PET

The detectors of a PET scanner are usually paving a cylinder and stacked in a set of rings of detectors [1]. The γ rays issued from the disintegration of a radiotracer particle are detected by a pair of sensors facing each other. The line which connects two sensors is called a line of response (LOR) and the coincidence events counted on one LOR are stored in a *bin*. All the bins are stored in a sinogram as illustrated in Figure 1. The reconstruction process attempts to estimate the image of the radiotracer distribution f that has produced the sinogram.

The sinogram p_{PET} is a 4D space along $(\Delta, \psi, u_{\parallel}, v_{\parallel})$. The coordinates $(\Delta, v_{\parallel})$ represent a couple of rings: Δ is the axial distance between the two rings (segment number) and $v_{\parallel}$ is the mean axial coordinate of the two rings (plane number). The coordinates $(\psi, u_{\parallel})$ represent one particular LOR between two rings: ψ is the azimuthal angle and $u_{\parallel}$ is the tangential coordinate (bin number).

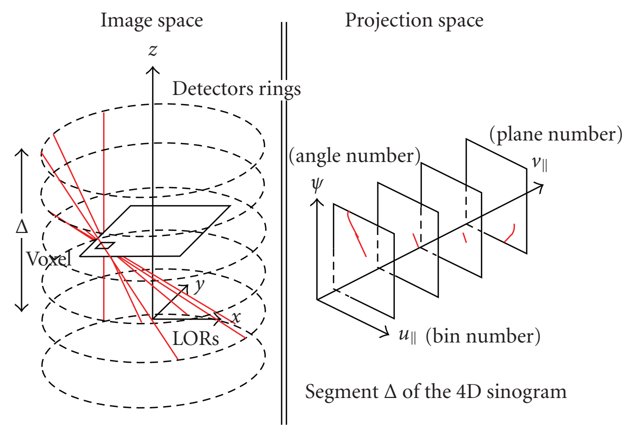


FIGURE 1: The acquired data are stored in a 4D sinogram, a sinogram bin corresponding to one particular LOR. To reconstruct one voxel, the data needed by the BP algorithm draw a 3D sinusoid in each segment Δ.

For each voxel (VOlume piXEL) of coordinate $\vec{r} = (x, y, z)$, the BP algorithm sums up all the sinogram bins corresponding to that voxel projection as follows:

$$f^*_{\text{PET}}(\vec{r}) = \iint p_{\text{PET}}(\Delta, \psi, u_{\parallel}(\psi, \vec{r}), v_{\parallel}(\psi, \Delta, \vec{r}))J_{\Delta} d\psi\, d\Delta, \tag{1}$$

J_{Δ} is a Jacobian and the parallel beam coordinates $(u_{\parallel}, v_{\parallel})$ are computed as

$$\begin{aligned} u_{\parallel}(\psi, \vec{r}) &= x \cos\psi + y \sin\psi + \text{offset}, \\ v_{\parallel}(\psi, \Delta, \vec{r}) &= \frac{\Delta}{2R_a} \cdot (x \sin\psi - y \cos\psi) + z + \text{offset}. \end{aligned} \tag{2}$$

Using $a_{ij}(\psi, \Delta)$ coefficients, we get

$$\begin{aligned} u_{\parallel}(\psi, \vec{r}) &= a_{00}x + a_{01}y + a_{03}, \\ v_{\parallel}(\psi, \Delta, \vec{r}) &= a_{10}x + a_{11}y + a_{12}z + a_{13}. \end{aligned} \tag{3}$$

2.1.2. Cone beam BP for CT

The cone beam BP used in CT imaging modalities uses a similar algorithm [4]. In CT, the data is the X-ray intensity reaching an X camera that rotates around the observed volume. It measures the attenuation due to the density of tissues. The density $f^*_{\text{CT}}(\vec{r})$ is computed from the measured data p_{CT} following:

$$f^*_{\text{CT}}(\vec{r}) = \int p_{\text{CT}}(\alpha, u_{\vee}(\alpha, \vec{r}), v_{\vee}(\alpha, \vec{r})) \cdot w(\alpha, \vec{r})^2 \cdot d\alpha, \tag{4}$$

where α is the trajectory parameter of the camera. The cone beam coordinates $(u_{\vee}, v_{\vee})$ are computed as

$$\begin{aligned} u_{\vee}(\alpha, \vec{r}) &= (c_{00}x + c_{01}y + c_{02}z + c_{03}) \cdot w(\alpha, \vec{r}), \\ v_{\vee}(\alpha, \vec{r}) &= (c_{10}x + c_{11}y + c_{12}z + c_{13}) \cdot w(\alpha, \vec{r}), \end{aligned} \tag{5}$$

where c_{ij} depends on α (i.e., $c_{ij} = c_{ij}(\alpha)$) and

$$w(\alpha, \vec{r}) = \frac{1}{c_{20}\cdot x + c_{21}\cdot y + c_{22}\cdot z + c_{23}}. \qquad (6)$$

2.1.3. Comparison of CT and PET

Although the CT BP is more complex due to the perspective transformation (6), these algorithms are quite similar. Indeed, the summation over α (trajectory parameter) for CT BP is equivalent to the summation over ψ and Δ for PET BP. Moreover, in these loops, both these BPs compute very similar projection coordinates $(u_\parallel, v_\parallel)$ and $(u_\vee, v_\vee)$. Nevertheless, the computation of the projection coordinates for CT BP needs a division by a distance weight $w(\alpha, \vec{r})$. Thus, the CT BP kernel has more arithmetic operations than PET BP has.

Supposing that one is able to design a pipeline that computes a sum update at each clock cycle, both for CT and PET BP, then the challenge is to fetch data along a complex path (a 3D sinusoid) in the acquired data (3D CT data or 4D PET sinogram). The method presented in this paper for solving the case of PET BP (parallel beam) could be transposed to solve the CT BP (cone beam).

2.2. Acceleration of reconstruction

Different computer architectures coupled with dedicated memory access strategies are used to accelerate the BP step of an analytic or iterative reconstruction, including general purpose processors [5–7], graphical processors [8–14], the cell processor [4, 15], or ASIC/FPGA architectures [16–20]. While most of these works have investigated cone beam BP, only a few of them have investigated 3D parallel beam BP [2, 5, 8, 9].

The parallelization of reconstruction algorithms on shared memory parallel general purpose computer [5] stays efficient only up to 4 processing units, because of conflictual accesses on the memory bus. Considering clusters of heterogeneous PCs [6, 7], efficiency of parallelization drops down quickly because of the costly communication between PCs. After 10 PCs, parallelization is not worthy. Yet on a distributed memory parallel computer, parallelization works very well. for 3D PET iterative reconstruction, Jones et al. [2] succeeded to get an acceleration factor of about 30 with 32 processors. Ni et al. [21] achieved an excellent acceleration factor of 300, when they parallelized the Katsevich algorithm, an exact cone beam BP with 300 CPUs.

Besides parallelization on several nodes of general purpose processors, more efficient engines such as the graphical processing unit (GPU) or the IBM Cell can be used. Current GPUs can be used either as a graphical pipeline, which is originally designed for [8–10], or as a multiprocessor chip thanks to the CUDA interface from Nvidia [10, 12–15]. For both options, the acceleration factor of GPU is high, about an order of magnitude for cone beam BP. Xu and Mueller [10] have observed that an implementation of the cone beam BP using the graphic pipeline is 3 times faster than the one made with the CUDA interface. Kachelrieß et al. [4] and Scherl et al. [15] present good result of acceleration of cone beam BP using the Cell processor. With its 1 + 8 cores, this architecture is an intermediate solution between general purpose parallel processors and GPU. The 8 vector engines have to be specifically programed. Nevertheless, Scherl et al. [11] have measured that a GPU with CUDA is 3 times faster than the Cell for the BP alone.

FPGA technology is an alternative to processors, allowing designers to make a customized architecture. Most often, it is used to prototype ASIC implementations. In this context, FPGA implementations of 2D parallel beam BP [16] and 3D cone beam BP [17–19] have been investigated. These architectures are made of several pipelines working in parallel. Moreover, like the image Pro by Siemens [18], several FPGA chips can be used in a single board to raise the computational power.

Two memory access strategies have been applied for all these architectures. In case the processor already has a memory cache, developers rely on it to optimize the external memory accesses. Otherwise, developers set up custom memory strategies in order to hide the memory access time. The most common approach is to use double buffering: the next required projection data is loaded from external memory, meanwhile the ongoing loaded projection data are back-projected. In this case, CPU and GPU memory strategies are based on an extensive use of the cache. For example Yang et al. [13] have observed that enabling a GPU cache is more competitive than software prefetching. On the other hand, the Cell and FPGA memory strategies have to be taken in charge by the software designers.

3. OVERCOMING THE MEMORY BOTTLENECK

In this section, we focus our study on finding out the best appropriate memory strategy to get the best fit between the 3D BP algorithm and a hardware architecture.

3.1. Memory access strategy

As the sinogram is kept in an SDRAM-like external memory, we need an efficient memory management to overcome its latency and allow a high level of parallelism. The main difficulty is to deal with the high strides of addresses due to the sinusoidal pattern of references in the 4D space. A cache would help to hide the high latency of the external memory despite these strides. Standard caches therefore are inefficient as they exploit temporal and address locality of references. Hence they are used at their best when the references follow a 1D linear pattern as a cache line is loaded when a miss occurs.

Indeed, as shown earlier, the reconstruction of a single voxel $f(\vec{x})$ needs to follow a 3D sinusoid in the 4D sinogram. Such a pattern is of poor address locality but has a high index locality. Moreover, because of the $v_\parallel$ dimension, the memory accesses for 3D BP have higher strides and are more distributed in the memory space than in the 2D BP case [22]. The challenge is to speed up these memory accesses in a 4D data structure.

```
for $n = 0$ to $n_{\max}$ do
    for Delta $= 0$ to Delta$_{\max}$ do
        for psi $= 0$ to psi$_{\max}$ do
            for $\vec{r} = \vec{r}_{\min}(n)$ to $\vec{r}_{\max}(n)$ do
$f(\vec{r}) += \text{bin}(\text{psi}, \text{Delta}, u_{\|}(\text{psi}, \vec{r}), v_{\|}(\text{psi}, \vec{x}))$
```

Algorithm 1: The loop reordering of the 3D BP improves spatial and temporal locality.

Therefore, a new cache mechanism is needed. Estimating which bins would be referenced would help the cache to download the needed bins during the computing process.

3.2. Improvement of spatial and temporal locality

A reconstruction with the voxel-driven bilinear interpolation (VBI) standard BP is made of three loops, as described in Algorithm 1: the first loop is on the voxels $\vec{r}$, the second on the segments (Delta), and the third on the angles (psi). Since voxels can be reconstructed independently, the loop on voxels can be split into two parts: one loop on blocks of voxels $(0, \dots, n_{\max})$ and the other loop on the voxels of a block $n(\vec{r}_{\min}(n), \dots, \vec{r}_{\max}(n))$.

A loop reordering increases the temporal and spatial locality of memory accesses. Indeed, for given psi and Delta values, the data $\text{bin}(\text{psi}, \vec{r})$ will be used several times for different voxels since the projection of a 3D block of voxels is a 2D plane in the 4D space of the sinogram.

Figure 2 shows that the proposed loop reordering allows to cache a part of the sinogram. The BP of a block of voxels makes the references follow a coherent 3D sinusoid in the sinogram along the time.

3.3. Mean bin reuse rate (MBRR)

To give a theoretical estimation of the best achievable cache efficiency, the mean bin reuse rate (MBRR) is defined as the ratio between the number of bins accessed in cache memory by the processing units and the number of bins loaded in cache memory from the external memory. The ideal MBRR can be computed analytically. It depends on the shape of the block of reconstructed voxels. Figure 3 presents this optimal MBRR computed for each segment versus the size of the block.

4. A 3P ARCHITECTURE FOR PET

In this section, we present the pipelined, prefetched, and parallelized architecture for PET (3PA-PET). The 3PA-PET architecture is made of a high-performance pipeline connected to a 3D adaptive and predictive cache (3D-AP cache). It allows to perform an update of a voxel value up to 1 operation per clock cycle (100% pipeline utilization), even for high latency memories.

4.1. Pipelined architecture

The pipeline in Figure 4 implements the different steps of the VBI standard BP: the computation of $u_{\|}(\text{psi}, \vec{r})$ and $v_{\|}(\text{psi}, \vec{r})$, the bilinear interpolation of the bin, and finally the accumulation of the voxel value. The forward flow control is done by packets passing through each stage of the pipeline.

The 4 bins needed for the bilinear interpolation are fetched through the memory bridge. This bridge controls the 3D-AP cache and can freeze (or not) the pipeline depending on the requested data availability. A backward flow control synchronizes the pipeline and the 3D-AP cache.

4.2. Prefetch architecture

The 3D-AP cache [3] masks the latency of the external memory so that the pipeline is no more systematically stalled. The memory bridge gets four bins from the cache at each clock cycle.

The 3D-AP cache is a semigeneric cache memory mechanism that prefetches references following a continuous path into a 3D memory space. It was originally designed as a cache for a computer vision lip tracking application [3] but it targets a large class of multidimensional processing algorithms. In the 3PA-PET architecture, the 3D-AP cache is tuned to follow the references needed to reconstruct a block of voxels, as shown in Figure 5. The pipeline issues spatial coordinates of the requested bin, here $(\text{psi}, u_{\|}, v_{\|})$, to the 3D-AP cache. A part of the sinogram, namely, the *cached zone*, is copied in an embedded memory. A tracking mechanism tries to maintain the center of the cached zone in the mean coordinate of the referenced data.

The 3D-AP cache estimates dynamically which data is likely to be requested in the future. This is done by a statistical analysis on each axis of the previous references. Moreover, the 3D-AP cache masks the data transfer between the external memory and the internal cache memory. In the mean time, the cache grabs new data from the external memory, the data shared by the old and the new cached zone stay available for the processing unit.

The 3D-AP cache parameters have to be set beforehand by the user. In this study, we set for each dimension the value of the following five parameters.

(i) *Cut off and sampling frequencies:* the mean coordinate is computed by a first-order low-pass IIR filter configured by these two frequencies.

(ii) *Cached zone size:* this zone is notified to the memory bridge to be available in cache. In this study, this size is a static parameter.

(iii) *Guard zone size* when the mean coordinate is out of this zone, the cache zone is updated.

(iv) *Cache speed:* it has to be set according to the speed of the data accesses performed by the application on each spatial dimension.

The cache is customized to allow four concurrent accesses to the bins needed to perform a bilinear interpolation. Figure 6 gives a simplified view of the 3D-AP cache to illustrate the

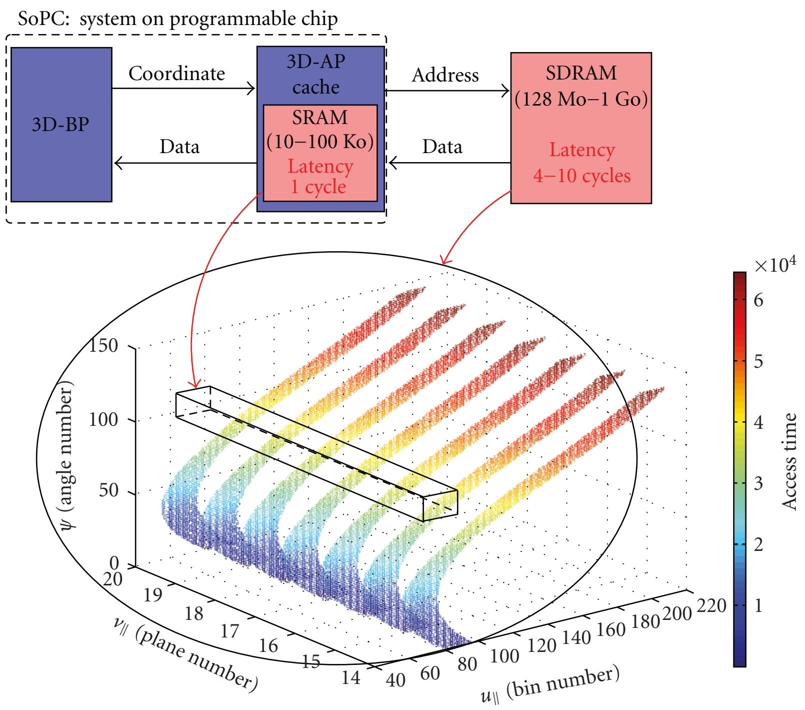


FIGURE 2: The memory access strategy is based on a fast and small cache memory inside the SoPC. The cache predicts the needs of the 3D BP unit and therefore succeeds to mask the high latency (4–10 cycles at 200 Mhz) of the slower and bigger external SDRAM memory.

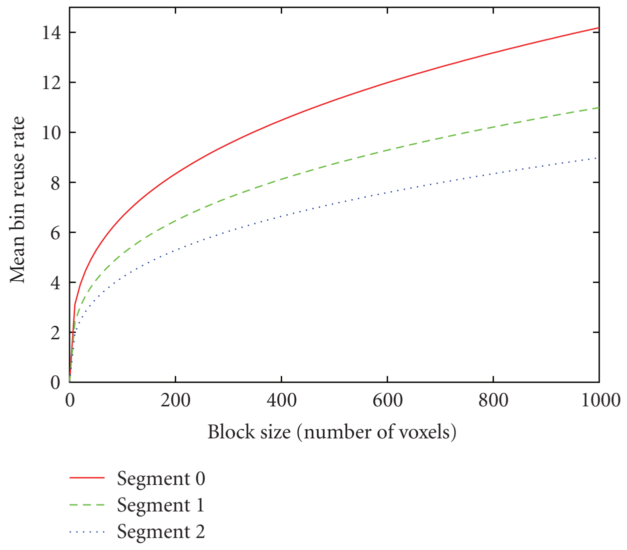


FIGURE 3: Mean bin reuse rate (MBRR) estimated for a 3D BP without bilinear interpolation versus the size of reconstructed blocks of voxels for each segment of a Siemens HR+ sinogram (span 9 with 96 angles of projection).

involved memory architecture. The cache control unit grabs data from the external memory and splits the incoming data words to the different embedded memories. The cache control unit also manages the cache misses that could occur for some requested bins.

4.3. Parallelized architecture

To increase the computing power, several pipelines are parallelized. A hierarchical cache reduces the memory bus occupation, when BP units work in parallel. In this hierarchical design, one leaf cache is associated to one 3D BP unit while a root cache is feeding each of these leaf caches.

The spatial locality of the references of the pipelines is enabled by reconstructing a set of neighbor blocs. A pipeline reconstructs one bloc of voxels and all the pipelines share a loop over psi. Some of the sinogram data are shared by the pipelines in the same way they are shared to reconstruct one bloc of data. The bins needed during the reconstruction of a set of blocs draw a 3D sinusoid. The cache concept presented previously with one unit applies here in the same manner. Each leaf cache stores a 3D sinusoid needed to reconstruct a bloc. A higher level cache stores the union of these sinusoids as presented in Figure 7.

5. 3PA-PET PERFORMANCES

5.1. Accuracy of reconstruction

The implemented VBI standard BP is a fixed-point version of the original algorithm. Moreover, the sinogram data is converted in *float* to *short integer* (16 bits). The accuracy of reconstruction of 3PA-PET is measured between a reference reconstruction software and a software bit-true model of 3PA-PET.

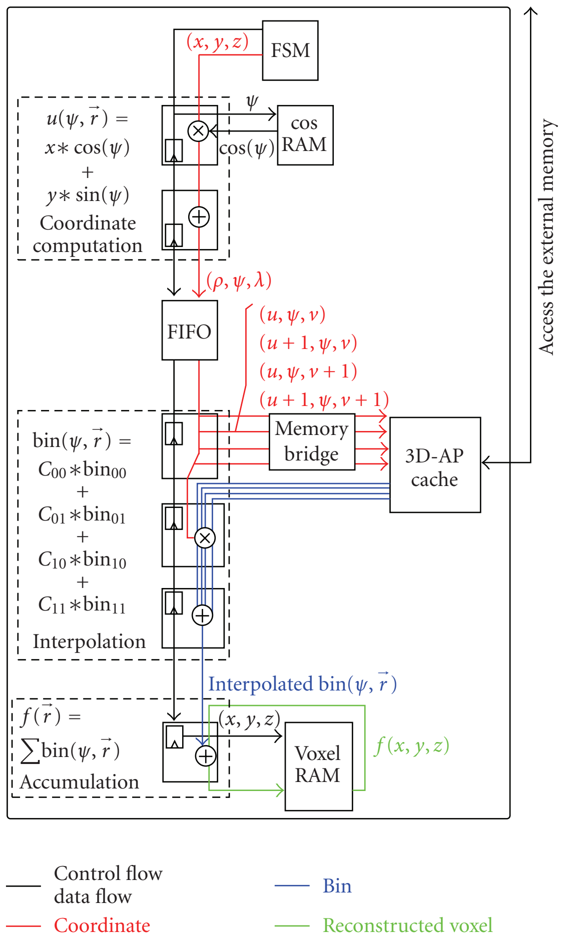


Figure 4: Pipeline of 3PA-PET.

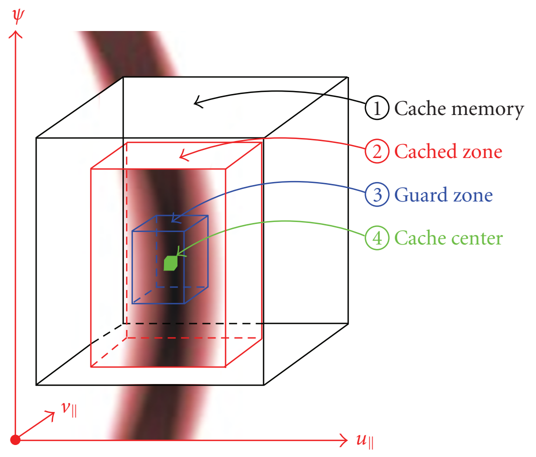


Figure 5: 3A-AP cache zones.

The reference dataset used is a sinogram of a 3D Shepp-Logan volume of 128 × 128 × 63 voxels. This phantom is a standard volume used in tomography to measure the accuracy of reconstruction. The sinogram is obtained from the STIR open source tool kit [23]. The volumes reconstructed by STIR and by 3PA-PET are shown on Figures 8 and 9.

Table 1: Accuracy of reconstruction and compared reconstructions for the Shepp-Logan phantom.

Compared volumes	Data	MAPE	PSNR
Accuracy of reconstruction			
STIR/original	Float	3.89%	10.5 dB
VBI-Flt/original	Float	3.88%	10.5 dB
VBI-Fix/original	Float	3.88%	10.5 dB
VBI-Flt/original	Int16	3.97%	10.5 dB
VBI-Fix/original	Int16	3.97%	10.5 dB
Compared reconstructions			
STIR/VBI-Flt	Float	0.35%	21.5 dB
VBI-Fix/VBI-Flt	Float	0.13%	26.2 dB
VBI-Fix/VBI-Flt	Int16	0.13%	23.0 dB
VBI-Fix/VBI-Flt	Int16/flt	1.1%	19.0 dB

The accuracy of reconstruction of the 3PA-PET BP is measured with two metrics: the mean absolute percentage error (MAPE) and the peak signal-to-noise ratio (PSNR). Both compare a volume f_1 with a volume of reference f_{ref}. The PSNR corresponds to the ratio between the maximum of f_{ref} (dynamic range) and the mean squared error (MSE) of f_1 compared to f_{ref},

$$\text{PSNR} = 20\cdot\log_{10}\frac{\max\left(f_{\text{ref}}\right)}{\sqrt{\text{MSE}\left(f_{\text{ref}}, f_1\right)}}. \tag{7}$$

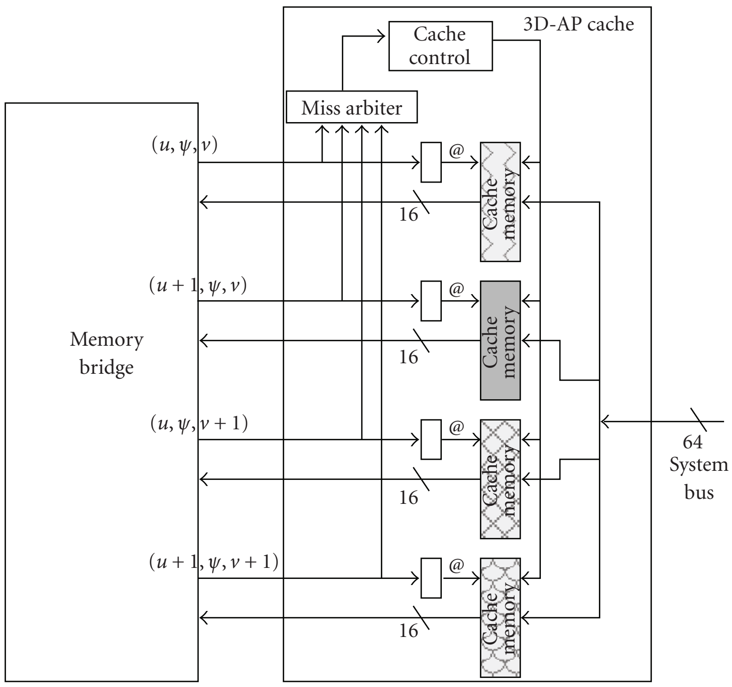
In Table 1, we compared the reference volume and the volumes reconstructed with STIR, with VBI floating-point arithmetic (VBI-flt) or with VBI fixed-point arithmetic (VBI-fix). All of the reconstruction methods have an intrinsic error around 3.9% with a PSNR of 10.5 dB when compared with the original volume. The floating-point and the fixed-point implementations have an MAPE of 0.13% and a PSNR of 23 dB. With different data types (*short int* versus *float*), the MAPE is about 1.1% and the PSNR of 19 dB. Thus we can conclude that the 3PA-PET implementation of the VBI BP is an accurate reconstruction system.

5.2. 3PA-PET complexity

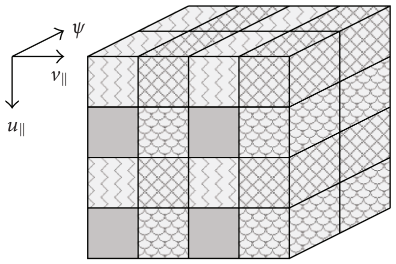
The hardware resources used by the 3PA-PET architecture are presented in Table 2. The main BP FSM and the root cache control are shared between all of the units of the 3PA-PET architecture. Therefore, the cost of an additional pipeline is only 800 slices. The sizes of a leaf and the root caches are, respectively, 2 KB and 18 KB. Hence, 9 BP units fit in a Xilinx Virtex 2 Pro VP30 chip and 16 units in a virtex 4 FX100.

5.3. Efficiency of reconstruction

In order to assess the efficiency of the 3PA-PET architecture, we have measured the BP time on an Avnet development board connected to a PC host through a PCI interface



(a) Customized concurrent 3D-AP cache architecture



(b) Mapping of bins to memory buffers

FIGURE 6: Memory architecture for bilinear interpolation.

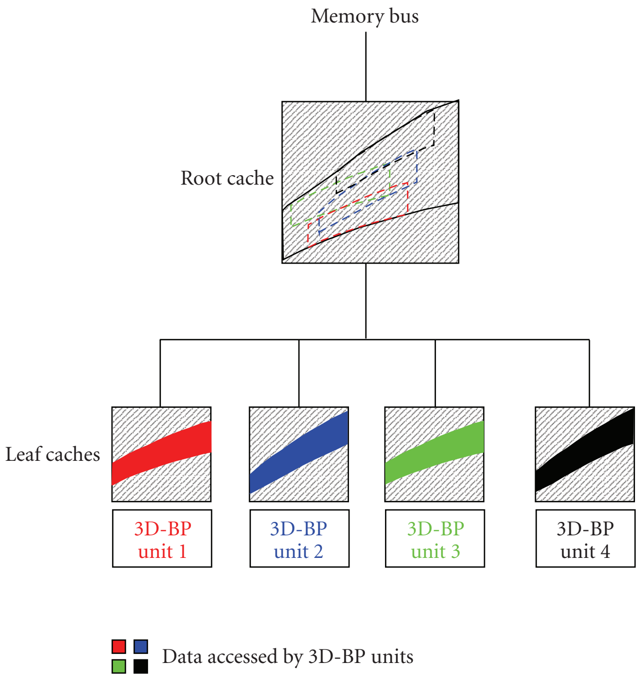


FIGURE 7: Each leaf cache is fed by the root cache.

TABLE 2: Hardware resources used by the 3PA-PET on a Xilinx Virtex 2 Pro VP30.

	1 unit	4 units	9 units
	3D BP		
CLB slices	573 (4.2%)	1817 (13.3%)	3924 (28.6%)
Multipliers	12 (9%)	48 (35%)	108 (79%)
	3D-AP cache		
CLB slices	672 (4.9%)	2830 (20.6%)	4804 (35.1%)
RAMs	2 kB (0.6%)	24 kB (7.8%)	36 kB (11.7%)
	3D BP + 3D-AP cache		
CLB slices	1245 (9.1%)	4637 (32.9%)	8728 (63.7%)

(see Figure 10). The board contains an external SDRAM memory and a Xilinx system on programmable chip (SoPC). In order to investigate the 3PA-PET behavior with respect to the memory features, we have plugged it with a fake memory bus which could be set with different values of memory latency (l_{mem}) and memory bandwidth (BW_{mem}). The memory simulator estimates the time to access N_{line} lines of S_{line} bytes following the relationship

$$t_{\text{mem}} = N_{\text{line}} \cdot \left(l_{\text{mem}} + \frac{S_{\text{line}} - 1}{BW_{\text{mem}}} \right). \quad (8)$$

The times of reconstruction presented in this section are in clock cycles and scaled to one operation. An operation corresponds to one update of a voxel. The number of voxel's

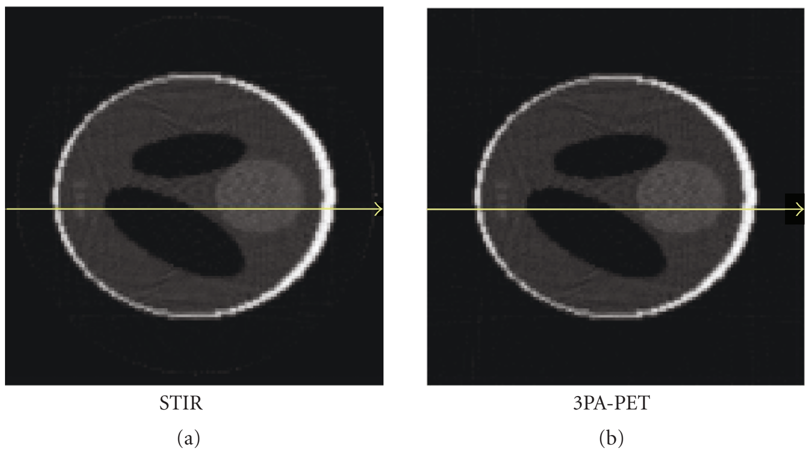


Figure 8: A slice of the 3D Shepp-Logan phantom reconstructed by STIR and 3PA-PET BP.

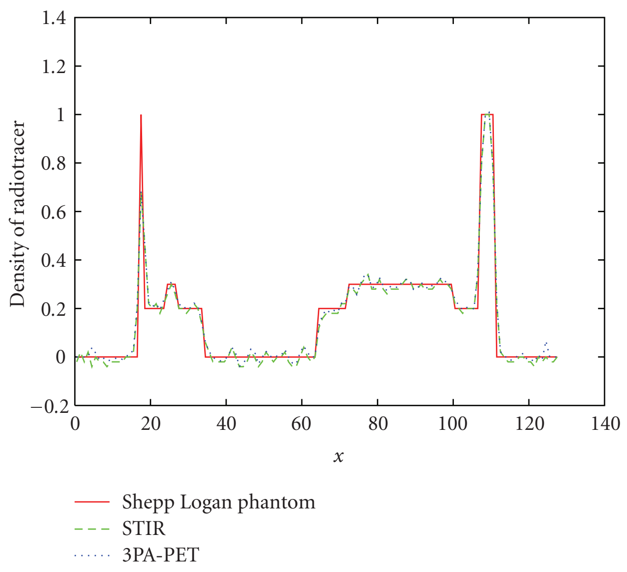


Figure 9: Profile of the 3D Shepp-Logan phantom slices corresponding to the lines on Figure 8.

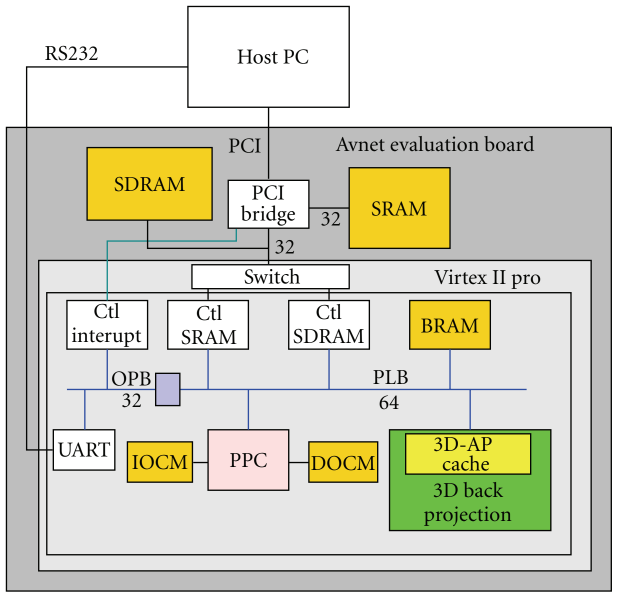


Figure 10: Evaluation system.

updates is equal to the number of voxels multiplied by the number of segments times the number of angles.

The results presented in Figure 11 are achieved with one BP unit for the segment +2 which represents the worse case because the memory accesses draw the most incurvated 3D sinusoid. 3PA-PET is robust to high latencies and low bandwidth: the pipeline computes a voxel update in about 1 clock cycle, even for a memory latency of 30 cycles. This shows that the 3D-AP cache succeeds to take advantage of the high spatial and temporal locality of the BP algorithm presented in Section 3. The 3D-AP cache follows the 3D memory path drawn during the BP process rather well. The cache miss rate stays low (about 0.05% with $l_{\text{mem}} = 5$ cycles and $BW_{\text{mem}} = 8$ bytes/cycle) which means that the 3D-AP cache prediction is satisfactory and manages to hide the external memory latency.

As illustrated in Figure 12, the parallel 3PA-PET performances are not plenty satisfactory. Indeed, the efficiency of parallelization decreases with the number of BP units. For instance, with a memory latency of 5 and for a complete BP, 4 units allow an acceleration of 3.2 (1.25 cycle/op per pipeline) and 8 units allow an acceleration of 4.7 (1.7 cycle/op per pipeline). Because the more units are working in parallel, the more busy the memory bus is. However, the hierarchical cache allows to make parallelization a little bit more efficient thanks to the exploitation of the spatial and temporal locality existing between the data retrieved by each BP unit. Moreover, the measured MBRR for 8 units between the leaf loads and the leaf requests stays close to the MBRR measured for a single unit. This MBRR is about 8 for 8 units and 9 for 1 unit.

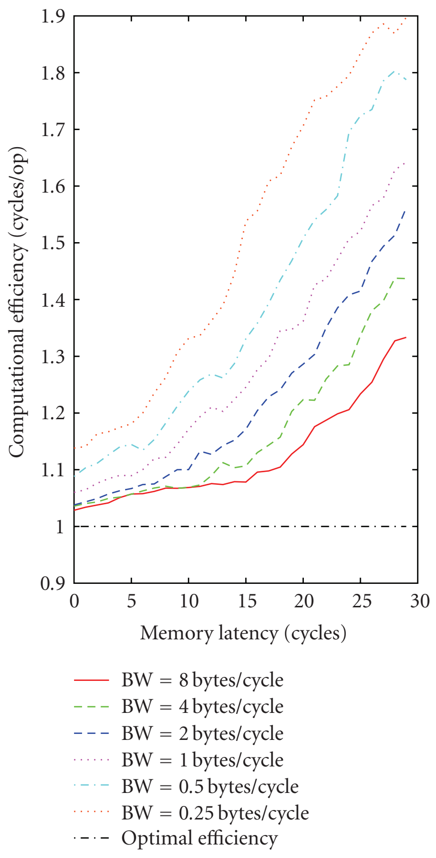


FIGURE 11: Cycles per operation for one unit of BP with respect to the latency and bandwidth (BW) of the external memory.

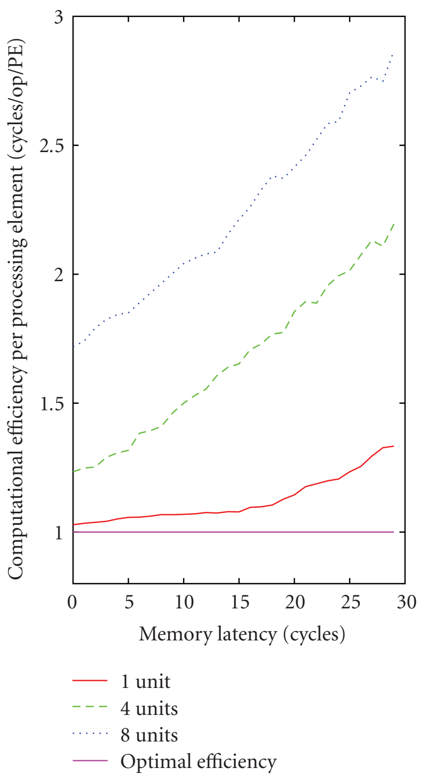


FIGURE 12: Cycles per operation per processing units for 1, 4, and 8 units of BP with respect to the latency and bandwidth of the external memory.

```
for $n = 0$ to $n_{\max}$ do
  for delta $= 0$ to delta$_{\max}$ do
    for psi $= 0$ to psi$_{\max}$ do
      $u_{\|} = xn_0 \cdot \cos\psi + yn_0 \cdot \sin\psi$
      for $xn = xn_0$ to $xn_{\max}$ do
        $u_{\|} = u_{\|} + \cos\psi$
        for $yn = yn_0$ to $yn_{\max}$ do
          $u_{\|} = u_{\|} + \sin\psi$
          for $zn = zn_0$ to $zn_{\max}$ do
$f(xn, yn, zn) += \text{bin}(\text{delta}, \text{psi}, u_{\|}, v_{\|})$
```

ALGORITHM 2: Reduction of operations to compute $u_{\|}$ (same techniques used for $v_{\|}$).

6. COMPARISON WITH GENERAL PURPOSE AND GRAPHICS PROCESSORS

In Table 3, the 3PA-PET execution times are compared with STIR and the ones from software VBI BP on a desktop PC, a workstation and a GPU.

6.1. *CPU implementation*

Different software versions of BP, nonoptimized (v1), and optimized (v2 and v3) have been tested and compared to the STIR one on a Pentium 4 and on a bi-Xeon dual core. Two techniques of optimization have been applied with an extensive use of the cache memory and a reduction of the arithmetical operations.

First, an acceleration factor of 3 is obtained due to the reconstruction through blocks of voxels. This software loop reordering increases the use of the L1 cache (16 Ko). Indeed, the time of reconstruction with and without introduction of data locality, is, respectively, 54.7 seconds (v1) and 17.4 seconds (v2).

Secondly, the reduction of the arithmetic operations to compute the projection coordinates allows an acceleration by a factor 7 (software v3/software v2 in Table 3). Indeed, the time performance is improved by a factor 2 due to an incremental computation of the coordinates as done by Kachelrieß et al. [4] and again by a factor 3.5 when the inner loop is over z. The optimized code (VBI-flt(v3)) is presented in Algorithm 2.

Finally, this code has been parallelized using the *pthread* C-library to use the four cores of a bi-Xeon dual core workstation. One thread is associated to the reconstruction of one block.

6.2. *GPU implementation*

Current GPUs are cost effective solutions for the implementation of 3D tomography reconstruction because of their

Table 3: Compared time performance for the 3D PET BP of a $128 \times 128 \times 63$ volume from a Siemens HR+ sinogram (5 segments, span 9, 96 angles of projection). Throughput of reconstruction (cycles per voxel update) is presented for the global architecture and per processing element (PE).

3D-BP algorithm	PE (threads)	Time	Cycles/Op /PE	Cycles/Op total
Desktop PC: Pentium 4 (core frequency = 3.2 Ghz, $BW_{\text{mem}} = 64$ GB/s)				
STIR[1]	1	11.13 s	70.4	70.4
VBI-flt(v1)	1	54.7 s	355	355
VBI-flt(v2)	1	17.4 s	113	113
VBI-flt(v3)	1	2.5 s	16	16
Workstation: bi-Xeon dual core (core frequency = 3 Ghz, $BW_{\text{mem}} = 10.6$ GB/s)				
STIR[1]	1 (1)	5.74 s	34.5	34.5
VBI-flt(v3)	1 (1)	1.17 s	7.1	7.1
VBI-flt(v3)	2 (2)	583 ms	7.06	3.53
VBI-flt(v3)	4 (4)	294 ms	7.12	1.78
GPU: GTS8800 (shader frequency = 1.2 Ghz, $BW_{\text{mem}} = 64$ GB/s)				
VBI-flt(v4)	96 (192)	99 ms	25.9	0.27
VBI-flt(v5)	96 (192)	50 ms	13.0	0.14
FPGA[2]: virtex 4 (frequency = 200 Mhz, $BW_{\text{mem}} = 0.8$ GB/s, $l_{\text{mem}} = 25$ nanoseconds)				
VBI-fix	1	2.5 s	1	1
VBI-fix	4	774 ms	1.25	0.31
VBI-fix	8	526 ms	1.7	0.21
ASIC[3]: one memory bank (frequency = 1.2 Ghz, $BW_{\text{mem}} = 4.8$ GB/s, $l_{\text{mem}} = 25$ nanoseconds)				
VBI-fix	1	499 ms	1.21	1.21
VBI-fix	4	214 ms	2.07	0.517
VBI-fix	8	135 ms	2.62	0.328
ASIC[3]: five memory banks (frequency = 1.2 Ghz, $BW_{\text{mem}} = 24$ GB/s, $l_{\text{mem}} = 25$ nanoseconds)				
VBI-fix	40	27 ms	2.62	0.065

[1] Time normalized to a $128*128*63$ volume. (STIR reconstructs $64^2\pi*63$ cylindrical volumes).
[2] 35 Mhz results scaled to 200 Mhz ($l_{\text{mem}} = 5$ cycles).
[3] 35 Mhz results scaled to 1,2 Ghz ($l_{\text{mem}} = 30$ cycles).

high level of parallelism. Moreover, the Nvidia GPUs are efficiently and easily programed with the CUDA environment.

The Nvidia Geforce 8880 family has 2 to 16 vector processors (12 in our case), each one having 8 stream processors. It is programmable using standard C language with a few extensions without any knowledge about graphics pipeline. A nonincremental code is parallelized to run efficiently on these 12×8 multithreaded stream processors. One thread is associated to one voxel reconstruction. Threads are grouped in blocks (16×16 in our case) which are scheduled at runtime, one block per vector processor. Each couple of vector processors are associated with an 8 KB L1 2D texture read-only cache memory with 1D and 2D hard-wired interpolation. Moreover, the GPU offers a high memory bandwidth ($BW_{\text{mem}} = 64$ GB/s) and uses floating-point computation. This makes it possible to efficiently parallelize the BP loops, as blocks of voxels correspond to 2D blocks of threads having access to the read-only sinogram organized in 2D arrays through the 2D cache memory. The voxels are also organized in 2D arrays, each divided in 64 16×16 blocks associated to a grid of 64 16×16 blocks of threads. Thus each thread is responsible of 63 voxels considering a $63 \times 128 \times 128$ volume.

Two versions of thread code have been implemented. In the VBI-flt(v4) thread code, the loop over ψ is the inner loop, while in VBI-flt(v5) thread code, the loop over z is the inner loop as it is done for the VBI-flt(v3) CPU code. This allows a reduction of the number of projection coordinate computation. A speedup factor of 2 is obtained with this code optimization (see Table 3).

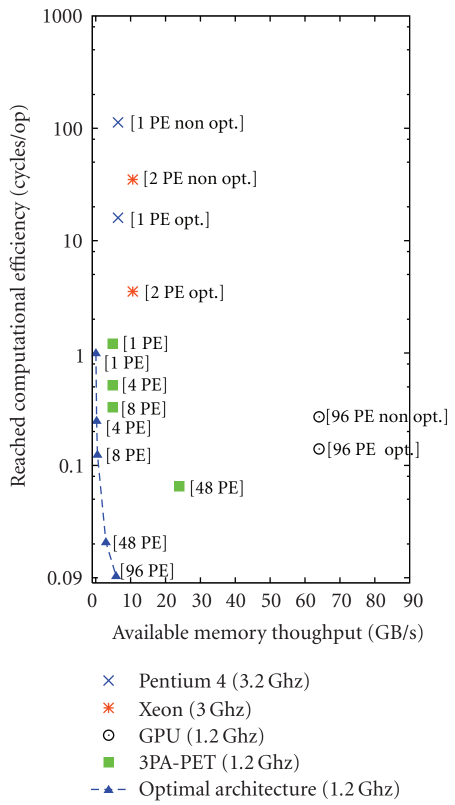


Figure 13: Memory throughput exploitation by CPU (optimized and nonoptimized code), GPU (optimized and nonoptimized code), and 3PA-PET implementations. 3PA-PET and optimal architecture results are obtained with $l_{\text{mem}} = 25$ nanoseconds.

6.3. *Discussion*

The reconstruction times and efficiencies, global and per processing element (PE), are presented in Table 3 for CPU, GPU, and our 3PA-PET. To fairly compare our architecture with other technologies, the time measured on a Virtex 2 Pro has been scaled to a virtex 4. Indeed, this technology is the same generation as the CPU and GPU used in this study. We have scaled the 35 MHz results to the GPU frequency (1.2 GHz) as well. This higher frequency could be reached through the design of a customized integrated circuit like an ASIC. Moreover, as Nvidia GTS 8800 GPU has five memory banks, we also present a prospective ASIC architecture which would have also five memory banks coupled with 5 processing blocks of 8 BP units each.

For the 200 MHz and the 1.2 GHz 3PA-PET, a memory latency (l_{mem}) of 25 nanoseconds has been used for the simulated memory bus. It corresponds, respectively, to a latency of 5 and 30 clock cycles.

On one hand, the GPU is the fastest hardware solution with a final reconstruction time of 50 milliseconds. The ratio of computation over memory access is high enough for the GPU to allow the automatic overlapping of memory accesses with computations by the thread scheduling mechanism. Thus, due to its greatest computational power (96 PEs and hard-wired interpolation), Nvidia 8800 GTS graphic processor is 10 times faster than 3PA-PET mapped on a virtex 4, 10 times faster than a Xeon dual core, and 50 times faster than a Pentium 4.

On the other hand, 3PA-PET is the most efficient architecture with a computational efficiency per processing element (PE) of about 2 cycles per operation for a processing block made of 8 units coupled with one memory bank. Because of the fewer available computational and memory resources, the FPGA technology does not allow to have an efficient 3PA-PET system compared to GPU. Nevertheless, considering that a market would exist to justify it, an ASIC with five memory banks and five units of 3PA-PET (8 pipelines each) running at 1.2 Ghz would be twice faster than the Nvidia GPU, 20 times than a Xeon dual core, and 100 times than a Pentium 4. Furthermore, a better tuning of the 3D-AP cache would allow to increase the available parallelism and again increase the speedup. Also, an ASIC implementation would be likely to have a lower consumption than a GPU (Nvidia 8800 GTS needs 130 Watts).

All the studied architectures succeed to benefit from the spatial and temporal localities without any developer effort to set a double buffering memory strategy. This is only possible because of their own memory cache (1D cache for CPU, 2D texture cache for GPU, and the semigeneral purpose 3D-AP cache for 3PA-PET). Nevertheless, 3PA-PET is the one that best exploits the memory throughput, as illustrated in Figure 13. In this figure, all the 3D BP implementations are placed according to their computational efficiency (cycles/op) and to their available memory throughput (GB/s). The "optimal architecture" in this figure corresponds to a hardware architecture that would have an optimal balance between its computational and memory throughputs. Each PE of this optimal architecture computes one operation per cycle and its prefetching memory strategy only loads the necessary data in cache and delivers it in time to the processing units. Of course, the more PEs it has, the greater the memory throughput has to be. As one can observe, 3PA-PET is the architecture with a cache-based memory strategy that is the closest to the optimal one. This makes 3PA-PET the architecture with the best potential of acceleration.

7. CONCLUSION

This paper presents several ways to speed up the BP algorithm on different target architectures: general purpose CPU, GPU, and FPGA/ASIC. These solutions exploit the temporal and 3D spatial locality that can be found in the BP algorithm. A suitable loop reordering shows to be efficient despite the high nonlinearity of the algorithm. The 3PA-PET (prefetched and parallelized architecture for PET) architecture is the one that makes the best use of this locality and allows a high level of parallelization with a high computational throughput.

Thanks to the 3D-AP cache together with a loop reordering, 3PA-PET architecture proves to be an efficient parallel architecture that overcomes the memory bottleneck. Indeed, as it has been measured on an SoPC prototype, the pipelines are seldom stalled and the high latency and low bandwidth of memories can be overcome. Moreover, the comparison between the 3PA-PET architecture with a general purpose processor and a GPU highlights 3PA-PET efficiency. On one hand, the GPU has the best reconstruction time on a wall clock (followed by 3PA-PET and the CPU), on the other hand 3PA-PET makes the best use of the pipeline and clock cycles. An ASIC implementation with the same technological resources than of GPU would be of lower power consumption and would be faster: it would be twice faster than today's GPUs and 20 times faster than CPUs.

To conclude, the method of loop reordering and the use of an appropriate cache could be extended to other algorithms. The architecture principles presented in this article could be applied for the cone beam BP needed in CT reconstruction.

REFERENCES

[1] P. E. Kinahan, M. Defrise, and R. Clackdoyle, "Analytic image reconstruction methods," in *Emission Tomography: The Fundamentals of PET and SPECT*, pp. 421–442, Elsevier Academic Press, San Diego, Calif, USA, 2004.

[2] J. P. Jones, A. Rahmim, M. Sibomana, et al., "Data processing methods for a high throughput brain imaging PET research center," in *Proceedings of the IEEE Nuclear Science Symposium (NSS '06)*, vol. 4, pp. 2224–2228, San Diego, Calif, USA, October-November 2006.

[3] S. Mancini and N. Eveno, "An IIR based 2D adaptive and predictive cache for image processing," in *Proceedings of the 19th Conference on Design of Circuits and Integrated Systems (DCIS '04)*, p. 85, Bordeaux, France, November 2004.

[4] M. Kachelrieß, M. Knaup, and O. Bockenbach, "Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware," *Medical Physics*, vol. 34, no. 4, pp. 1474–1486, 2007.

[5] M. Schellmann, T. Kösters, and S. Gorlatch, "Parallelization and runtime prediction of the Listmode OSEM algorithm for 3D PET reconstruction," in *Proceedings of the IEEE Nuclear Science Symposium (NSS '06)*, vol. 4, pp. 2190–2195, San Diego, Calif, USA, October-November 2006.

[6] D. W. Shattuck, J. Rapela, E. Asma, A. Chatzioannou, J. Qi, and R. M. Leahy, "Internet2-based 3D PET image recontruction using a PC cluster," *Physics in Medicine and Biology*, vol. 47, no. 15, pp. 2785–2795, 2002.

[7] T. He, J. Ni, and G. Wang, "A heterogeneous windows cluster system for medical image reconstruction," in *Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS '06)*, vol. 1, pp. 410–415, Zhejiang, China, June 2006.

[8] K. Chidlow and T. Möller, "Rapid emission tomography reconstruction," in *Proceedings of the 3rd Eurographics/IEEE TVCG Intenational Workshop on Volume Graphics (VG '03)*, vol. 45, pp. 15–26, Tokyo, Japan, July 2003.

[9] G. Pratx, G. Chinn, F. Habte, P. Olcott, and C. Levin, "Fully 3-D list-mode OSEM accelerated by graphics processing units," in *Proceedings of the IEEE Nuclear Science Symposium (NSS '06)*, vol. 4, pp. 2196–2202, San Diego, Calif, USA, October-November 2006.

[10] F. Xu and K. Mueller, "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," *Physics in Medicine and Biology*, vol. 52, no. 12, pp. 3405–3419, 2007.

[11] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger, "Fast GPU-based CT reconstruction using the Common Unified Device Architecture (CUDA)," in *Proceedings of the IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS-MIC '07)*, vol. 6, pp. 4464–4466, Honolulu, Hawaii, USA, October-November 2007.

[12] T. Schiwietz, S. Bose, J. Maltz, and R. Westermann, "A fast and high-quality cone beam reconstruction pipeline using the GPU," in *Medical Imaging 2007: Physics of Medical Imaging*, vol. 6510 of *Proceedings of SPIE*, San Diego, Calif, USA, February 2007.

[13] H. Yang, M. Li, K. Koizumi, and H. Kudo, "Accelerating backprojections via CUDA architecture," in *Proceedings of the 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 52–55, Lindau, Germany, July 2007.

[14] D. Riabkov, X. Xue, D. Tubbs, and A. Cheryauka, "Accelerated cone-beam backprojection using GPU-CPU hardware," in *Proceedings of the 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 68–71, Lindau, Germany, July 2007.

[15] H. Scherl, S. Hoppe, F. Dennerlein, et al., "On-the-fly-reconstruction in exact cone-beam CT using the cell broadband engine architecture," in *Proceedings of the 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 29–32, Lindau, Germany, July 2007.

[16] M. Leeser, S. Coric, E. Miller, H. Yu, and M. Trepanier, "Parallel-beam backprojection: an FPGA implementation optimized for medical imaging," *The Journal of VLSI Signal Processing*, vol. 39, no. 3, pp. 295–311, 2005.

[17] X. Li, T. He, S. Wang, G. Wang, and J. Ni, "P2P-enhanced distributed computing in EM medical image reconstruction," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA '04)*, vol. 2, pp. 822–828, Las Vegas, Nev, USA, June 2004.

[18] B. Heigl and M. Kowarschik, "High-speed reconstruction for C-arm computed tomography," in *Proceedings of the 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, pp. 25–28, Lindau, Germany, July 2007.

[19] I. Goddard and M. Trepanier, "High-speed cone-beam reconstruction: an embedded systems approach," in *Medical Imaging 2002: Visualization, Image-Guided Procedures, and Display*, vol. 4681 of *Proceedings of SPIE*, pp. 483–491, San Diego, Calif, USA, February 2002.

[20] Terarecon, http://www.terarecon.com.

[21] J. Ni, J. Deng, H. Yu, T. He, and G. Wang, "Analysis of performance evaluation of parallel Katsevich algorithm for 3-D CT image reconstruction," in *Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences (IMSCCS '06)*, vol. 1, pp. 258–265, Zhejiang, China, June 2006.

[22] N. Gac, S. Mancini, and M. Desvignes, "Hardware/software 2D-3D backprojection on a SoPC platform," in *Proceedings of the ACM Symposium on Applied Computing (SAC '06)*, vol. 1, pp. 222–228, Dijon, France, April 2006.

[23] K. Thielemans, S. Mustafovic, and C. Tsoumpas, "STIR: software for tomographic image reconstruction release 2," in *Proceedings of the IEEE Nuclear Science Symposium (NSS '06)*, vol. 4, pp. 2174–2176, San Diego, Calif, USA, October-November 2006.