Smart Memory Synthesis for Energy-Efficient Computed Tomography Reconstruction

Qiuling Zhu, Larry Pileggi, Franz Franchetti
Dept. of Electrical and Comp. Eng., Carnegie Mellon University, Pittsburgh, PA, USA
Email: qiulingz@andrew.cmu.edu, franzf@ece.cmu.edu

***Abstract*—As nanoscale lithography challenges mandate greater pattern regularity and commonality for logic and memory circuits, new opportunities are created to affordably synthesize more powerful smart memory blocks for specific applications. Leveraging the ability to embed logic inside the memory block boundary, we demonstrate the synthesis of smart memory architectures that exploits the inherent memory address patterns of the backprojection algorithm to enable efficient image reconstruction at minimum hardware overhead. An end-to-end design framework in sub-20nm CMOS technologies was constructed for the physical synthesis of smart memories and exploration of the huge design space. Our experimental results show that customizing memory for the computerized tomography parallel backprojection can achieve more than 30% area and power savings with marginal sacrifice of image accuracy.**

***Index Terms*—Smart Memory; Hardware Synthesis; Computed Tomography; Parallel Backprojection;**

I. Introduction

Computationally intensive algorithms in medical image processing (e.g., computerized tomography (CT)) require rapid processing of large amounts of data and often rely on hardware acceleration [1], [8], [2]. Inherent parallelism in the algorithms is exploited to achieve the required performance by increasing the number of parallel functional units at a cost of power and area. The overall performance is often defined by the limited bandwidth of the on-chip memory as well as the high cost of memory access.

One approach to address these challenges is to optimize the on-chip memory organization by constructing a customized smart memory module that is optimized for a particular function for higher performance and/or energy efficiency [11], [10]. Recent studies of sub-20nm CMOS design indicate that memory and logic circuits can be implemented together using a small set of well-characterized pattern constructs [5], [6]. Our early silicon experiments in a commercial 14nm SOI CMOS process demonstrate that this construct-based design enables logic and bitcells to be placed in a much closer proximity without yield or hotspots pattern concerns. Moreover, such restrictive patterning enables the synthesis (not just compilation) of customized memory blocks with user control of flexible SRAM architectures and facilitates *smart memory compilation*.

To efficiently leverage this new technology, however, algorithms and hardware architectures need to be revised. In this paper we revisit the Shepp and Logan's backprojection algorithm that is widely used in the CT image reconstruction. It is observed that in the parallel implementation of the algorithm, the memory address differences are fairly small for adjacent projection angles and adjacent pixels. We exploit this property via a customized memory structure that could feed in-parallel running image processing engines with a large amount of required projection data in one clock cycle. The implementation is realized by embedding "intelligent" functionality into the traditional interleaved memories and allow multiple memory sub-banks to share the memory periphery. We further construct a smart memory design framework that provides the end user with finer control of the customized SRAM architecture parameters, thus enabling automatic generation of the specified implementation. Physical implementations were carried out in a commercial 14 nm SOI CMOS process. Our results indicate more than 40% area savings and 30% power savings. The marginal impact on accuracy is minimized with appropriate constraints on the algorithm.

II. Address Pattern Exploration

In a parallel-beam CT scanning system, the object to be scanned is placed between the evenly spaced array of an unidirectional X-ray source and the detector. Radiation beams from the source pass through the object and are measured at the detector. A complete set of projections is obtained by rotating the arrays and taking measurements for different angles over 180°, forming the Radon transform of the image (i.e., projection data). The inverse of the projection data allows to reconstruct the tomographic images (i.e., backprojection) [9], [1].

Shepp and Logan backprojection algorithm. The Shepp and Logan backprojection algorithm is the most well-known backprojection algortithm [2], [3]. For each pixel, P located at (x, y), and each projection angle θ_i, the first step in backprojection is to locate the pixel in an appropriate beam (ray). If the center of P is not on a ray, the distance (d) to its adjacent rays is calculated and the contribution from the adjacent rays to the pixel (Q_p) is computed according to the linear interpolation equation (1), assuming that pixel is enclosed by the t_{th} and $(t+1)_{th}$th rays,

$$Q_p(x, y, \theta_i) = R_t + (d/L) \cdot (R_{t+1} - R_t), \tag{1}$$

where R_t is the value of t_{th} ray, d is the interpolation distance, and L is the ray interval. Q_p represents the contribution of the projection angle θ_i to the current pixel value.

In the above equation, the address t to the projection data memory and the interpolation distance d are computed as follows (assuming the target image has the dimension size of $r \times c$):

$$t_{x,y,\theta_i} = \big(x - \frac{r}{2}\big) \cdot \cos\theta_i - \big(y - \frac{c}{2}\big) \cdot \sin\theta_i + t_{\text{offset}}. \tag{2}$$

$$d = t(\theta) - \lfloor t(\theta) \rfloor. \tag{3}$$

Address difference. The above procedures are to be repeated for every angle and for every pixel, which involves significant address computation and memory access operations. To illustrate the inherent address pattern, we show the address to the next projection of angle θ_{i+1} in (4):

$$t_{x,y,\theta_{i+1}} = \big(x - \frac{r}{2}\big) \cdot \cos\left(\theta_{i+1}\right) - \big(y - \frac{c}{2}\big) \cdot \sin\left(\theta_{i+1}\right) + t_{\text{offset}}. \tag{4}$$

The address difference (δt_1) between (2) and (4) could be as

$$\delta t_1 = \big(x - \frac{r}{2}\big) \cdot \delta cos_{\theta_i} + \big(\frac{c}{2} - y\big) \cdot \delta sin_{\theta_i}, \tag{5}$$

with $\delta cos_{\theta_i} = \cos(\theta_{i+1}) - \cos(\theta_i)$ and $\delta sin_{\theta_i} = \sin(\theta_{i+1}) - \sin(\theta_i)$. Using trigonometric identities, we can compute the bounds on (5) as

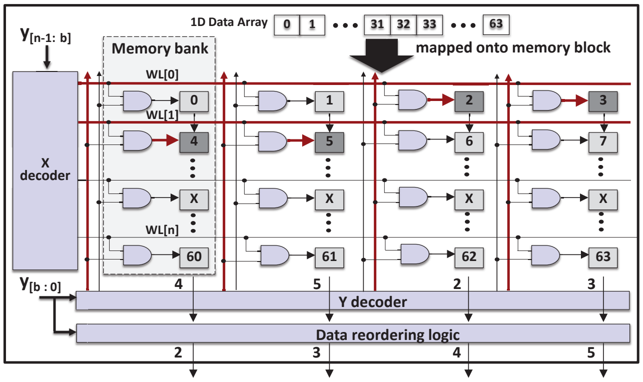


Fig. 1. Consecutive Access Memory.

follows (assuming $r = c$ and N is the total number of projections):

$$|\delta t_1| \le |2\cdot\sin\left(\frac{\pi}{N}\right)\cdot\frac{r}{2}\cdot\left(\cos\left(\frac{\pi(2i+1)}{N}\right)-\sin\left(\frac{\pi(2i+1)}{N}\right)\right)|. \tag{6}$$

(6) has a maximum bound of $\sqrt{2}\pi \cdot \frac{r}{N}$ for relatively large N. This shows that δt_1 is limited to a fairly small range when the appropriate ratio of r and N is selected. For example, the value is always less than one when $\frac{r}{N} \le \frac{1}{8}$.

This observation can easily extend to two scenarios below:

(a) The address difference between the next k projection memory of angle θ_k and the first memory of angle θ_1 for the same pixel $P(x, y)$ will increase proportionally to k:

$$|\delta t_k| = |t_{x,y,\theta_{i+k}} - t_{x,y,\theta_i}| \le \sqrt{2}\pi \cdot \frac{r}{N} \cdot k \cdot \approx 4.44 \cdot \frac{r}{N} \cdot k. \tag{7}$$

(b) The address differences when both pixel coordinate and projection angle are incremented are also bounded by a limited range. For demonstration purpose, we define the problem as to reconstruct four neighborhood pixels in parallel, that is, $(x,\ y)$, $(x+1,\ y)$, $(x,\ y+1)$, $(x+1,\ y+1)$. Then, their addresses in adjacent k projection memories for angles from θ_i and θ_{i+k} need to be computed. We denote the address of the first pixel $(x,\ y)$ in the first memory θ_i as the reference address ($t_0 = t_{x,y,\theta_i}$). Then we can easily prove that other addresses are all very close to t_0 for the required k, and the maximum possible address difference to t_0 is introduced by the last pixel $(x+1,\ y+1)$ in the last projection memory θ_{i+k},

$$\delta t_{max} = t_{x+1,y+1,\theta_{i+k}} - t_{x,y,\theta_i} = \cos\theta_i + \sin\theta_i + k\cdot\delta t_1. \tag{8}$$

It is easy to show that (8) has the maximum value of $\sqrt{2}+4.44\cdot\frac{r}{N}\cdot k$ and it is limited to small range, e.g., the value must be less than four when $\frac{r}{N} \le \frac{1}{8}$ and $k = 4$.

The basic idea is, as the address differences for adjacent projections angles and adjacent pixels are small, these addresses will activate the same or adjacent wordlines when such memories are located horizontally in parallel with each other. It leads to opportunities to share the memory decoder among these memories by programming "intelligent" logic functionalities into the memory periphery.

III. Smart Memory Customization for Parallel Backprojection Architecture

In this section, we describe our approach to optimize the memory organization and backprojection architecture based on the observed memory access patterns.

A. Consecutive Access Memory

As we mentioned, linear interpolation is an important procedure of the algorithm. Linear interpolation requires the access to two adjacent array addresses of the projection memory in a single clock cycle.

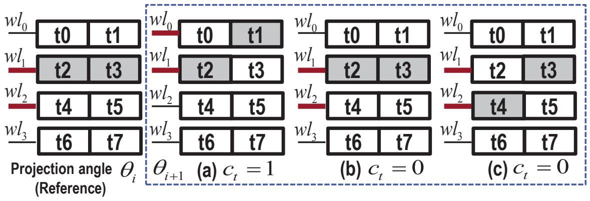


Fig. 2. Data Layout in Adjacent Projection Memories.

In our previous work [12], we have proposed a *rectangular-access smart memory* which is able to output an arbitrary rectangular block in a 2D data array. Its $1D$ simplified version, called $1D$ *Consecutive Access Memory*, can be used to output consecutive elements from a $1D$ data array. The functionality is defined as to support single-clock-cycle access of 2^b data points from a 2^n size data array. We build a parameterized memory which is first divided into 2^b memory banks and these memory sub-banks are located horizontally parallel to each other. Fig. 1 shows the organization of the memory block when $n = 6$ and $b = 2$. The main idea is to let these 2^b memory banks share one modified X-decoder. The X-decoder is specifically designed to activate two adjacent wordlines simultaneously (e.g.,$WL[0]$ and $WL[1]$). Another Y-decoder is used to select one of the two activated wordlines for each memory bank with the additional AND operations. This consecutive access memory serves as the basic memory structure in our method. In the rest of paper, we will propose more advanced memory sharing strategies customized for backprojection algorithms.

B. Smart Memory Organization and Parallel Backprojection

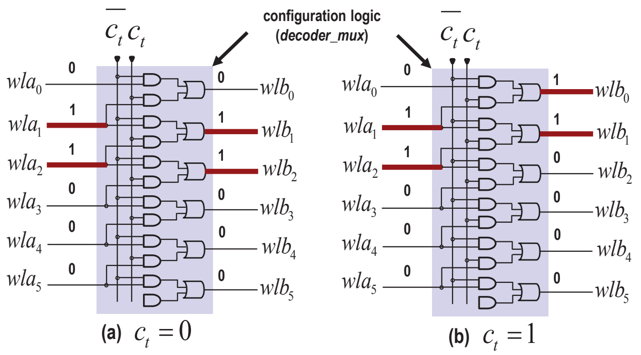
We will use a simple example to show the basic idea of the method. From the analysis of equation (6), we have derived that the address difference of the two adjacent memories (δt_{θ_1}) is less than one when $\frac{r}{N} \le \frac{1}{8}$. This implies that the two adjacent memory addresses after rounding must be either the same or adjacent to each other. In Fig. 2, we show the physical data layout in our consecutive access memory. If the address of projection θ_i is located in between t_2 and t_3 (denoted by $[t_2, t_3]$), then the address in the next adjacent projection θ_{i+1} for the same pixel has only three possible locations, that is, $[t_1, t_2]$, $[t_2, t_3]$ or $[t_3, t_4]$. In the illustration we highlight the corresponding active wordlines if implemented in the consecutive access memory. It's seen that if the active wordlines for the first memory is wla_1 and wla_2, then in the next memory, the active wordlines must be either the same (wlb_1 and wlb_2) or shifted upwards by one step (wlb_0 and wlb_1). We use a control signal c_t to differentiate these two situations and c_t can be calculated from the input address. Based on this observation, we propose two "smart" memory approaches which are named *decoder-mux* and *output-mux* respectively.

Decoder-mux. In the first approach, called *decoder-mux*, we eliminate the decoder of the second memory and let it share the same decoder with the first memory by adding some configuration logic (decoder-mux) in between the two sets of memory wordlines. This logic configures the wordlines of the first projection memory (wla_i) to generate the wordlines for the next adjacent projection memory (wlb_i) (see Fig. 3). The relationship between the wordlines of the two adjacent memories can be derived as

$$b_i = (-c_t)\cdot a_i + c_t \cdot a_{i+1}. \tag{9}$$

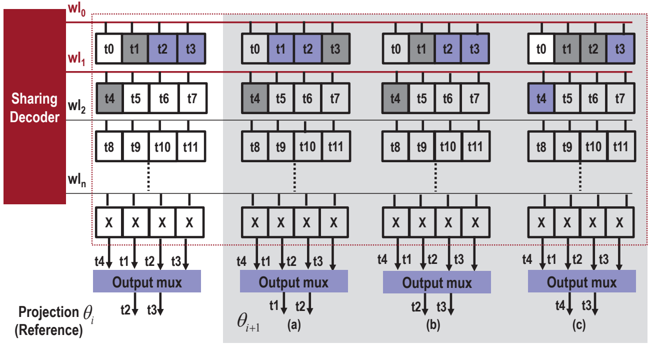
The configuration can be implemented using only AND and OR logic gates, which ensures the feasibility of the hardware implementation.

Output-mux. In the alternative approach named *output-mux* the two memories still share the decoder but the configuration logic is located outside of the memory (see Fig. 4). In this approach,



(a) $c_t = 0$ (b) $c_t = 1$

Fig. 3. Decoder-MUX.



(a) (b) (c)

Fig. 4. Output-MUX.

memories are designed as the 1×4 consecutive access memories to output more elements than required. In this example, t_2, t_3 along with their nearest neighbors t_1 and t_4 are all read out from the memories. Then the configuration logic (*output-mux*) is used to select the appropriate two elements from the four outputs. In this approach, the active wordlines for the two memories are always the same.

Horizontal and vertical parallel backprojection. To exploit the proposed smart memory to obtain superior hardware efficiency of the parallel backprojection, we propose two parallel approaches, *horizontal and vertical parallel backprojection*.

The horizontal parallel backprojection can perform more than two backprojections in parallel and all the involved projection memories share the same memory decoder using either *decoder-mux* or *output-mux* approach. Fig. 5 shows the example of accessing in eight adjacent projection memories. Assuming that the pixels addressed by the first memory addresses are t_3 and t_4, we highlight the possible locations of the two pixels accessed in the next seven memories. We observe that they are all clustered locally around t_3 and t_4, and are bounded by t_0 and t_7. Required pixels spread out further from t_3 and t_4 for memories that are further away from the first memory as explained by formulae (7), as the address difference of the next k projection memories from the first reference is increasing proportionally with k. Similar to the *output-mux* design shown in Fig. 4, we configure each projection memory as an 1×8 consecutive access memory to output all the shown eight pixels and use another 8-to-2 output-mux to select the appropriate two outputs from the eight outputs for each projection memory. In this way, all the eight memories could share the same decoder and seven memories decoders are saved. However, as the projection memories output more pixels than required, many memory outputs are actually wasted. An approach to use these wasted pixels is applying another vertical parallel backprojection, which performs the backprojections of multiple neighborhood pixels in parallel. E.g., in equation (8) we discuss the address differences for performing the backprojections of four neighborhood pixels concurrently. Backprojection of each pixel per projection angle requires one linear interpolation and involves memory accessing of two pixels, so totally it requires eight pixels to be accessed from each projection memory. (8) shows that these eight pixels will be contained in the outputs of the above 1×8 access memory in most situations. Therefore, the memory architecture needs no change for the vertical parallel backprojection since we just take advantage of the unused memory outputs from the horizontal parallel backprojection. By implementing both horizontal and vertical parallel backprojection concurrently using the modified consecutive access memory, all the memory outputs are utilized and a much higher throughput is achieved.

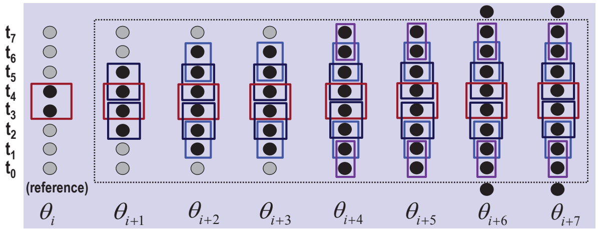


Fig. 5. Accessing in Parallel Projection Memories .

IV. Design Automation

Design tradeoff space analysis. Designing a CT image reconstruction system is a tradeoff problem involving algorithmic constraints, performance, hardware cost, and image accuracy. The discussion of address patterns in Section II shows that the ratio of image dimension size (r) and the projection numbers (N), r/N, is an important algorithm constraint. Smaller r/N indicates smaller adjacent address differences, which allows for more adjacent projection memories sharing the memory decoder, saving more hardware cost. However, it also limits the use of the method in applications with larger image size r and/or fewer projection angles N. For larger r/N, the corresponding larger address difference will limit the number of projection memories that can share the decoder. For example, in Fig. 5, the last two projection memories of θ_{i+6} and θ_{i+7} may require to access two pixels at the two ends, which are not accessible along with other eight pixels from the 1×8 consecutive access memory. To solve this problem we could increase the memory access width and apply more complicated configuration logic. However, this would increase the hardware cost. Alternatively, to lower hardware cost we could assign the nearest neighborhood pixels if the requested pixels are not available, which unfortunately will result in the loss of image accuracy. This shows that different design decisions will result in different tradeoffs. The combination of these design choices constitutes a huge design space. Further, exploring the design tradeoff space requires customized memory designs, which are traditionally prohibitively expensive. Thus, a strong design automation tool is required to make the hardware synthesis feasible.

End-to-end smart memory design framework. We have developed a *smart memory design framework* that provides designers with a graphical user interface to select design parameters, and automatically generate the optimized smart memory hardware IP [11], [10], [12]. As shown in Fig. 6, the tool frontend is built using the chip generator infrastructure "GENESIS" [7], [4]. It provides a user-configurable graphical interface that allows the user to input design specification and generates the optimized RTL automatically. The tool backend is a *smart memory compiler* for the physically synthesis of customized smart memory, which is developed based on the logic and memory co-design methodology [5], [6], [12]. Using this tool, embedded random logic and memory periphery are synthesized with the memory bitcells one shot to a small set of pre-characterized layout

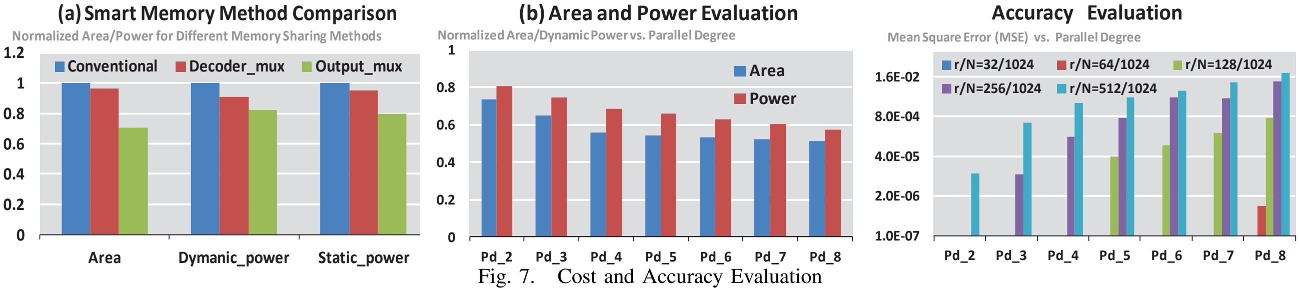
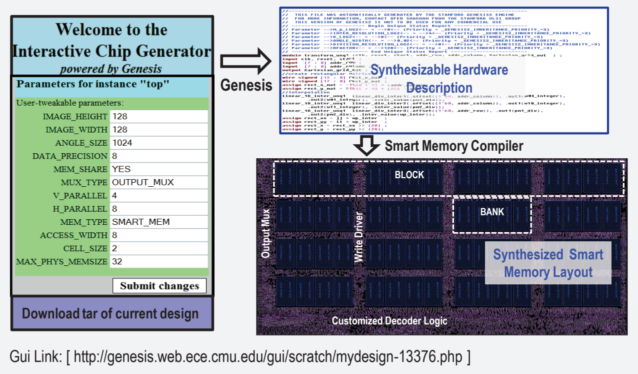


Fig. 7. Cost and Accuracy Evaluation



Parameters for instance "top"

User-tweakable parameters:

IMAGE_HEIGHT	128
IMAGE_WIDTH	128
ANGLE_SIZE	1024
DATA_PRECISION	8
MEM_SHARE	YES
MUX_TYPE	OUTPUT_MUX
V_PARALLEL	4
H_PARALLEL	8
MEM_TYPE	SMART_MEM
ACCESS_WIDTH	8
CELL_SIZE	2
MAX_PHYS_MEMSIZE	32

Submit changes

Download tar of current design

Gui Link: [http://genesis.web.ece.cmu.edu/gui/scratch/mydesign-13376.php]

Fig. 6. Smart Memory Design Framework

pattern constructs. Lithographic compliance between the co-designed logic and memory ensures the sub-20nm manufacturability of smart memory blocks. The architectural frontend and physical backend is combined to build an end-to-end smart memory design framework. Its input is the design specification and the output is ready to use hardware (RTL, GDS, .lib, .lef).

V. Evaluation and Results

In Fig. 7 (a), we first compare the hardware cost of two smart memory approaches (*decoder-mux* and *output-mux*) to the conventional memory. The memories studied here have the size of 4,096-words and wordlength of 16 bits, and we only consider two memories implemented as 1×8 consecutive access memories sharing the decoder with each other. We observe that the *output-mux* approach is more cost-efficient. The reason is that in *decoder-mux* each wordline is accompanied by a set of configuration logic, and each set of logic communicates with its local wordline. This explains also why *decoder-mux* achieves relatively higher power-efficiency compared to its area-efficiency. In contrast, *output-mux* only requires a single large configuration logic at the memory output. Due to the superiority of the *output-mux* method, it will be used for our backprojection system in the following discussions.

In Fig. 7 (b) we evaluate the hardware cost of the proposed memory architecture for reconstructing a 256×256-size image from 1,024 projections. The x-axis is the parallel degree P_d, which is defined as the number of adjacent backprojections that are performed concurrently and share the same memory decoder. We vary P_d from two to eight to show its impact on the cost. The y-axis shows the relative area and dynamic power compared to the conventional design where no memory sharing strategies are used. We see that more than 40% area savings and more than 30% power savings can be achieved with the increase of P_d. We also measure the mean square error (MSE) of the reconstructed image compared to the reference image (see Fig. 7 (c)). As expected, the error increases when either P_d or algorithm parameter (r/N) increases, which allows us to tradeoff image accuracy with hardware cost in applications where minor distortion is acceptable.

VI. Conclusion

The emergence of construct-based design facilitates the robust synthesis of cost-effective smart memory blocks that are customized for specific applications. This creates opportunities to re-design algorithms and re-architect the hardware structure to match the advanced technology capabilities. In this paper we propose smart memory architectures and the end-to-end design framework to implement them for the CT image reconstruction problems. The results in a 14nm CMOS process demonstrate significant improvements in area and power. Moreover, we present the opportunities to tradeoff hardware cost with acceptable image accuracy based on appropriate algorithm tuning. This paper demonstrates that the embedded memories in data-intensive computing can exploit the smart memory design methodology and the inherent address pattern of the algorithm to achieve superior power and performance efficiency.

Acknowledgement

The authors acknowledge the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

References

[1] I. Agi, P. J. Hurst, and K. W. Current. An image processing ic for backprojection and spatial histogramming in a pipelined array. *IEEE Journal of Solid-State Circuits*, 28(3):210–221, 1993.

[2] C. Chen, Z. Cho, and C. Wang. A fast implementation of the incremental backprojection algorithms for parallel beam geometries. *IEEE Transactions on nuclear science*, 43(6):3328–3334, 1996.

[3] Z. Cho, C. Chen, and S. Lee. Incremental algorithm - a new fast backprojection scheme for parallel beam geometries. *IEEE Transactions on Medical Image*, 9(2):207–217, 1990.

[4] [online] http://genesis2.stanford.edu/mediawiki/index.php.

[5] D. Morris, V. Rovner, L. Pileggi, A. Strojwas, and K. Vaidyanathan. Enabling application-specific integrated circuits on limited pattern constructs. *Symp. VLSI Technology*, June 2010.

[6] D. Morris, K. Vaidyanathan, N. Lafferty, K. Lai, L. Liebmann, and L. Pileggi. Design of embedded memory and logic based on pattern constructs. *Symp. VLSI Technology*, June 2011.

[7] O. Shacham. Chip multiprocessor generator: automatic generation of custom and heterogeneous compute platforms. *PhD Thesis, Stanford*, 2011.

[8] C. Srdjan, L. Miriam, and et al. Parallel-beam backprojection: An fpga implementation optimized for medical imaging. *FPGA*, 2002.

[9] H. Yu. Memory architecture for data intensive image processing algorithms in reconfigurable hardware. *Master Thesis*, 2003.

[10] Q. Zhu, C. R. Bergery, E. L. Turnerz, L. Pileggi, and F. Franchetti. Polar format synthetic aperture radar in energy efficient application-specific logic-in-memory. *ICASSP*, 2012.

[11] Q. Zhu, E. L. Turnerz, C. R. Bergery, L. Pileggi, and F. Franchetti. Application-specific logic-in-memory for polar format synthetic aperture radar. *HPEC*, 2011.

[12] Q. Zhu, K. Vaidyanathan, O. Shachamy, M. Horowitz, L. Pileggi, and F. Franchetti. Design automation framework for application-specific logic-in-memory blocks. *ASAP*, 2012.