Reconfigurable Processor for Energy-Efficient Computational Photography

Rahul Rithe, Student Member, IEEE, Priyanka Raina, Student Member, IEEE, Nathan Ickes, Member, IEEE, Srikanth V. Tenneti, Student Member, IEEE, and Anantha P. Chandrakasan, Fellow, IEEE

Abstract—This paper presents an on-chip implementation of a scalable reconfigurable bilateral filtering processor for computational photography applications such as HDR imaging, low-light enhancement, and glare reduction. Careful pipelining and scheduling has minimized the local storage requirement to tens of kB. The 40-nm CMOS test chip operates from 98 MHz at 0.9 V to 25 MHz at 0.5 V. The test chip processes 13 megapixels/s while consuming 17.8 mW at 98 MHz and 0.9 V, achieving significant energy reduction compared with software implementations on recent mobile processors.

Index Terms—Bilateral filtering, bilateral grid, computational photography, high-dynamic-range (HDR) imaging, low-power electronics, low-voltage operation, voltage scaling.

I. INTRODUCTION

C

OMPUTATIONAL photography is transforming digital photography by significantly enhancing and extending the capabilities of a digital camera. The field encompasses a wide range of techniques such as high-dynamic-range (HDR) imaging [1], low-light enhancement [2], [3], panorama stitching [4], image deblurring [5], and light field photography [6], which allow users not only to capture a scene flawlessly, but also to reveal details that could otherwise not be seen.

Recent research has focused on specialized image sensors to capture information that is not captured by a regular CMOS image sensor. An image sensor with multibucket pixels is proposed in [7] to enable time multiplexed exposure that improves the image dynamic range and detects structured light illumination. A back-illuminated stacked CMOS sensor is proposed in [8] that uses spatially varying pixel exposures to support HDR imaging. An approach to reduce the temporal readout noise in [9] to improve low-light-level imaging. However, computational photography applications using regular CMOS image sensors that are currently used in the commercial cameras have so far been software based.

Such CPU/GPU-based implementations lead to high energy consumption and typically do not support real-time processing.

Nonlinear filtering techniques like bilateral filtering [10] form a significant part of computational photography. These techniques have a wide range of applications, including HDR imaging, low-light enhancement, tone management [11], video enhancement [12], and optical flow estimation [13]. The high computational complexity of such multimedia processing applications necessitates fast hardware implementations [14], [15] to enable real-time processing. In addition, energy-efficient operation is a critical concern for portable multimedia applications. Voltage and frequency scaling is an important technique for reducing power consumption while achieving high peak computational performance [16]. The energy efficiency of digital circuits is maximized at low supply voltages [17], which makes the ability to operate at low voltage ($V_{DD} \sim 0.5$ V) a key component of achieving low-power operation.

This work implements a reconfigurable multi-application processor for computational photography by exploring power reduction techniques at various design stages, such as algorithms, architectures, and circuits. The algorithms are optimized to reduce the computational complexity and memory requirement. A parallel and pipelined architecture enables high throughput while operating at low frequencies, which allows real-time processing on HD images. Circuit design for low-voltage operation ensures reliable performance down to 0.5 V.

The reconfigurable hardware implementation performs HDR imaging, low-light-enhanced (LLE) imaging, and glare reduction, as shown in Fig. 1. The filtering engine can also be accessed from off-chip and used with other applications. The input images are preprocessed for the specific functions. The core of
the processing unit are two bilateral filter engines that operate in parallel and decompose an image into a low-frequency base layer and a high-frequency detail layer. The bilateral filtering is performed using a bilateral grid structure that converts an input image into a three-dimensional (3-D) data structure and filters it by convolving with a 3-D Gaussian kernel. Parallel processing allows enhanced throughput while operating at low frequency and low voltage. The bilateral filtered images are post-processed to generate the outputs for the specific functions.

This paper describes bilateral filtering and its efficient implementation using the bilateral grid [18]. A scalable hardware architecture for the bilateral filter engine is described. Implementation of HDR imaging, low-light enhancement, and glare reduction using bilateral filtering is discussed in Section IV. The challenges of low-voltage operation and approaches to address process variation are described in Section V. Section VI provides measurement results for the testchip.

II. BILINEAR FILTERING

Bilateral filtering is a nonlinear technique that takes into account the difference in the pixel intensities as well as the pixel locations while assigning weights, as opposed to linear Gaussian filtering that assigns filter weights based solely on the pixel locations. For an image $I$ at pixel position $p$, the bilateral filtered output $I_B$ is defined by

$$I_B(p) = \sum_{n=-N}^{N} G_S(n) \cdot G_I(I(p) - I(p-n)) \cdot I(p-n). \quad (1)$$

The output value at each pixel in the image is a weighted average of the values in a neighborhood, where the weight is the product of a Gaussian on the spatial distance ($G_S$) with standard deviation $\sigma_s$ and a Gaussian on the pixel intensity/range difference ($G_I$) with standard deviation $\sigma_r$. In linear Gaussian filtering, on the other hand, the weights are determined solely by the spatial term. In bilateral filtering, the range term $G_I(I(p) - I(p-n))$ ensures that only those pixels in the vicinity that have similar intensities contribute significantly towards filtering. This avoids blurring across edges and results in an output that effectively reduces noise while preserving the scene details. Fig. 2 compares Gaussian filtering and bilateral filtering in reducing image noise and preserving details. However, nonlinear filtering is inefficient and slow to implement because the filter kernel is spatially variant. A direct implementation of bilateral filtering can take on the order of several minutes to process HD images. Faster approaches for bilateral filtering have been proposed that reduce the processing time by filtering subsampled versions of the image with discrete intensity kernels and reconstructing the filtered results using linear interpolation [1], [19]. A fast approach to bilateral filtering based on a box spatial kernel, which can be iterated to yield smooth spatial falloff, is proposed in [20]. However, real-time processing of HD images requires further speed-up.

A. Bilateral Grid

A software-based bilateral grid structure is described in [18], which enables fast bilateral filtering but requires a large amount of storage (65 MB for a 10-megapixel image) for processing.

In this work, we implement bilateral filtering using a reconfigurable grid, which reduces the storage requirement to 21.5 kB by scheduling the filtering engine so that only two grid rows need to be stored at a time. The implementation is flexible to allow varying grid sizes for energy/resolution scalable image processing.

The bilateral grid structure used by this chip is constructed as follows. The input image is partitioned into blocks of size $\sigma_s \times \sigma_s$, and a histogram of pixel intensity values is generated for each block. Each histogram has $256/\sigma_s$ bins. This results in a 3-D representation of the 2-D image, as shown in Fig. 3, referred to as the bilateral grid. Each grid cell $(i, j, r)$ stores the number of pixels in a block corresponding to that intensity bin ($W_i^j$) and their summed intensity ($I_i^j$). The processor supports block sizes ranging from $16 \times 16$ to $128 \times 128$ pixels with 4 to 16 intensity bins in the histogram.

The bilateral grid has two key advantages, given here:

- **Aggressive down-sampling**: the size of the blocks ($\sigma_s \times \sigma_s$) used while creating the grid and the number of intensity bins ($256/\sigma_s$) determine the amount by which the image is down-sampled. The grid merges blocks of $16 \times 16$ to $128 \times 128$ pixels into 4 to 16 grid cells. This significantly reduces the number of computations required for processing as well as the amount of on-chip storage required.

- **Built-in edge awareness**: two pixels that are spatially adjacent but have very different intensities end up far apart in the grid in the intensity dimension. Filtering the grid level-by-level using a linear Gaussian kernel, only the intensity levels that are near each other influence the filtering and the levels that are far apart do not contribute in each others filtering. This is equivalent to performing bilateral filtering on the 2-D image.

III. BILINEAR FILTER ENGINE

The bilateral filter engine using the bilateral grid is implemented as shown in Fig. 4. It consists of three components—the
grid assignment engine, the grid filtering engine, and the grid interpolation engine.

The image is scanned pixel by pixel in a block-wise manner. The size of the block is scalable from 16 × 16 pixels to 128 × 128 pixels. Depending on the intensity of the input pixel, it is assigned to one of the intensity bins. The number of intensity bins is also scalable from 4 to 16.

### A. Grid Assignment

The pixels are assigned to the appropriate grid cells by the grid assignment engines. The hardware has 16 grid assignment (GA) engines that can operate in parallel to process 16 intensity levels in the grid. However, four or eight grid assignment engines could be activated if the grid uses fewer intensity levels. Fig. 5 shows the architecture of the grid assignment engine. For each pixel from each block, its intensity is compared with the boundaries of the intensity bins using digital comparators. If the pixel intensity is within the bin boundaries, it is assigned to that intensity bin. Intensities of all the pixels assigned to a bin are summed by an accumulator. A weight counter maintains the count of number of pixels assigned to the bin. Both the summed intensity and weight are stored for each bin in on-chip memory.

### B. Grid Filtering

The convolution (Conv) engine, shown in Fig. 6, convolves the grid intensities and weights with a 3 × 3 × 3 Gaussian kernel, which is equivalent to bilateral filtering in the image domain, and returns the normalized intensity. The convolution is performed by multiplying the 27 coefficients of the filter kernel with the 27 grid cells and adding them using a three-stage adder tree. The intensity and weight are convolved in parallel and the convolved intensity is normalized with the convolved weight by
using a fixed point divider to make sure that there is no intensity scaling during filtering. The hardware has 16 convolution engines that can operate in parallel to filter a grid with 16 intensity levels. However, four or eight of them can be activated if fewer intensity levels are used.

**C. Grid Interpolation**

The interpolation engine, shown in Fig. 7, reconstructs the filtered 2-D image from the filtered grid. The filtered intensity value at pixel \((x, y)\) is obtained by trilinear interpolation of \(2 \times 2 \times 2\) filtered grid values surrounding the location \((x/\sigma_x, y/\sigma_y, I_{xy}/\sigma_r)\). Trilinear interpolation is equivalent to performing linear interpolations independently across each of the three dimensions of the grid. To meet throughput requirements, the interpolation engine is implemented as three pipelined stages of linear interpolations. The output value \(I_{HF}(x, y)\) is calculated from filtered grid values \(F_{ij}^r\) using four parallel linear interpolations along the \(i\) dimension, given by

\[
\begin{align*}
F_j^r & = F_{i,j}^r \times w_1^i + F_{i+1,j}^r \times w_2^i \\
F_{j+1}^r & = F_{i,j+1}^r \times w_1^i + F_{i+1,j+1}^r \times w_2^i \\
F_{j+1}^r & = F_{i,j+1}^r \times w_1^i + F_{i+1,j+1}^r \times w_2^i \\
F_{j+1}^r & = F_{i,j+1}^r \times w_1^i + F_{i+1,j+1}^r \times w_2^i
\end{align*}
\] (2)

followed by two parallel linear interpolations along the \(j\) dimension, given by

\[
\begin{align*}
F^r & = F_j^r \times w_1^j + F_{j+1}^r \times w_2^j \\
F^{r+1} & = F_j^r \times w_1^j + F_{j+1}^r \times w_2^j
\end{align*}
\] (3)

followed by an interpolation along the \(r\) dimension, given by

\[
I_{HF}(x, y) = F^r \times w_1^r + F^{r+1} \times w_2^r.
\] (4)

The interpolation weights, given by

\[
\begin{align*}
w_1^i & = \frac{x}{\sigma_x} - i; \quad w_2^i = i + 1 - \frac{x}{\sigma_x} \\
w_1^j & = \frac{y}{\sigma_y} - j; \quad w_2^j = j + 1 - \frac{y}{\sigma_y} \\
w_1^r & = \frac{I_{xy}}{\sigma_r} - r; \quad w_2^r = r + 1 - \frac{I_{xy}}{\sigma_r}
\end{align*}
\] (5)

are computed based on the output pixel location \((x, y)\), the intensity of the original pixel in the input image \(I_{xy}\) at location \((x, y)\), and the grid cell index \((i, j, r)\). The pixel location \((x, y)\) and the grid cell index \((i, j, r)\) are maintained in internal counters. The original pixel intensity \(I_{xy}\) is read from the DRAM in chunks of 32 pixels per read request to fully utilize the memory bandwidth.

The assigned and filtered grid cells are stored in the on-chip memory. The last three assigned blocks are stored in a temporary buffer, and two previous rows of grid blocks are stored in the SRAM. The last two filtered blocks are stored in the temporary buffer, and one filtered grid row is stored in the SRAM.

**D. Memory Management**

The grid processing tasks are scheduled to minimize local storage requirements and memory traffic. Fig. 8 shows the memory management scheme by task scheduling. Grid processing is performed cell by cell in a row-wise manner. The last three blocks are stored in the temporary buffer and the last two rows are stored in the SRAM. Once a \(3 \times 3 \times 3\) block is available, the convolution engine begins filtering the grid. When block \(A\), shown in Fig. 8, is being assigned, the convolution engine is filtering block \(F\). As filtering proceeds to the next block in the row, the first assigned block, stored in the SRAM, becomes redundant and is replaced by the first assigned block in the temporary buffer. The last two filtered blocks are stored in the temporary buffer and the previous row of filtered blocks are stored in the SRAM. As \(2 \times 2 \times 2\) filtered blocks become available, the interpolation engine begins reconstructing the output 2-D image. When block \(F\), shown in Fig. 8, is being filtered, the interpolation engine is reconstructing the output 2-D image from block \(I\). As interpolation proceeds to the next block in the row, the first filtered block, stored in the SRAM, becomes redundant and is replaced by the first filtered block in the temporary buffer. Boundary rows and columns are replicated for processing boundary cells. This scheduling scheme allows processing without storing the entire grid. Only two assigned grid rows and one filtered grid row need to be stored locally at a time. Memory management reduces the memory requirement to 21.5 kB for processing a 10-megapixel image and allows processing grids of arbitrary height using the same amount of on-chip memory.
E. Scalable Grid

The size of the grid is determined by the image size and the downsampling factors. For an image of size $I_W \times I_H$ pixels with the spatial and intensity/range downsampling factors $\sigma_s$ and $\sigma_r$ respectively, the grid width ($G_W$) and height ($G_H$) are given by

$$G_W = \frac{I_W}{\sigma_s}, \quad G_H = \frac{I_H}{\sigma_r}$$  \hspace{1cm} (6)

and the number of grid cells ($N_G$) is given by

$$N_G = G_W \times G_H \times \left(\frac{256}{\sigma_r}\right).$$  \hspace{1cm} (7)

The number of computations as well as storage depends directly on the size of the grid. Selecting the downsampling factors the same as the standard deviations of the spatial and intensity/range Gaussians in the bilateral filter, given by (1), provides a good tradeoff between the output quality and processing complexity. The choice of downsampling factors is guided by the image content and the application. Most applications work well with a coarse grid resolution of the order of 32 pixels with 8 to 12 intensity bins. If the image has high spatial details, a smaller $\sigma_s$ would result in better preservation of those details in the output. Similarly, a smaller $\sigma_r$ would help preserve fine intensity details. The grid size is configurable by adjusting $\sigma_s$ from 16 to 128, which scales the block size from $16 \times 16$ to $128 \times 128$ pixels and $\sigma_r$ from 16 to 64, which scales the number of intensity levels from 16 to 4. For a 10-megapixel ($4096 \times 2592$) image, the number of grid cells scales from 663 552 ($\sigma_s = 16, \sigma_r = 16$) to 2592 ($\sigma_s = 128, \sigma_r = 64$). The architecture achieves energy scalability by activating only the required number of hardware units for a given grid resolution.

The 21.5 kB of on-chip SRAM used to store two rows of created grid cells and one row of filtered grid cells. The SRAM is implemented as eight banks supporting a maximum of 256 cells in each row of the grid with 16 intensity levels, corresponding to the worst case of $\sigma_s = 16, \sigma_r = 16$. Each bank is clock and input gated to save energy when a lower resolution grid is used. Only one bank is used when $\sigma_s = 128$, and all eight banks are used when $\sigma_s = 16$. The bilateral filter engine achieves scalability by activating only the required number of processing engines and SRAM banks for the desired grid resolution.

IV. APPLICATIONS

The testchip has two bilateral filter engines, each processing four pixels/cycle. The processor performs HDR imaging, LLE imaging, and glare reduction using the bilateral filter engines.

A. High-Dynamic-Range Imaging

HDR imaging is a technique for capturing a greater dynamic range between the brightest and darkest regions of an image than a traditional digital camera can. It is done by capturing multiple images of the same scene with varying exposure levels, such that the low-exposure images capture the bright regions of the scene well without loss of detail and the high-exposure images capture the dark regions of the scene. These differently exposed images are then combined together into an HDR image, which more faithfully represents the brightness values in the scene.

The first step in HDR imaging is to create a composite HDR image from multiple differently exposed images which represents the true scene radiance value at each pixel of the image [21]. The true scene radiance value at each pixel is recovered from the recorded intensity $I$ and the exposure time $\Delta t$ as follows. The exposure $E$ is defined as the product of sensor irradiance $R$ (which is the amount of light hitting the camera sensor and is proportional to the scene radiance) and the exposure time $\Delta t$. The intensity $I$ is a nonlinear function of the exposure $E$, given by

$$I = f(R \times \Delta t).$$  \hspace{1cm} (8)

We can then obtain the sensor irradiance as given by

$$\log(R'_I) = g(I) - \log(\Delta t)$$  \hspace{1cm} (9)

where $g = \log f^{-1}$.

The mapping $g$ is known as the camera curve [21]. Fig. 9 shows the camera curves for the RGB color channels of a typical camera sensor.

The HDR creation module, shown in Fig. 10, takes values of a pixel from three different exposures ($I_{E_1}, I_{E_2}, I_{E_3}$) and generates an output pixel which represents the true scene radiance value ($I_{HDR}$) at that location. Since we are working with a finite range of discrete pixel values (8 bits per color), the camera
curves are stored as combinational lookup tables (LUTs) to enable fast access. The true (log) exposure values are obtained from the pixel intensities using the camera curves, followed by exposure time correction to obtain (log) scene radiance. The three resulting (log) radiance values obtained from the three images represent the radiance values of the same location in the scene. A weighted average of these three values is taken to obtain the final (log) radiance value. The weighting function gives a higher weight to the exposures in which pixel value is closer to the middle of the response function (thus avoiding the high contributions from images where the pixel value is saturated). In the end an exponentiation is performed to get the final radiance value (16 bits per pixel per color). Processing in the log domain simplifies the computations to additions and subtractions instead of multiplications and divisions.

Displaying HDR images on LDR media (8 b/pixel) requires tone mapping that compresses image dynamic range by non-linear filtering [1]. The 16-b/pixel HDR image is split into intensity and color channels. A low-frequency base layer and a high-frequency detail layer are created by bilateral filtering the HDR intensity in the log domain. The dynamic range of the base layer is compressed by a scaling factor in the log domain. The detail layer is untouched to preserve the details and the colors are scaled linearly to 8 b/pixel. Merging the compressed base layer, the detail layer and the color channels results in a tone-mapped HDR image.

In HDR mode, both bilateral grids are configured to perform filtering in an interleaved manner, where each grid processes alternate blocks in parallel.

Fig. 11 shows a set of input low-dynamic-range exposures and the tone-mapped HDR output image.

**B. Glare Reduction**

Glare reduction is similar to performing single image HDR tone-mapping. The input image is split into intensity and color channels. A low-frequency base layer and a high-frequency detail layer are created by bilateral filtering the intensity. The contrast of the base layer is enhanced using the contrast adjustment module, shown in Fig. 12, which is also used in HDR tone-mapping. The contrast can be increased or decreased depending on the adjustment factor. The scaled color data are merged with the contrast-enhanced base layer and the detail layer to obtain a glare reduced output image.

Fig. 13 shows an input image with glare and the glare-reduced output image. Glare reduction recovers details that are white-washed in the original image and enhances the image colors and contrast.

**C. LLE Imaging**

LLE imaging is performed by merging two images captured in quick succession, one taken without flash \(I_{NF}\) and one with flash \(I_{F}\) [2], [3]. The bilateral grid is used to decompose both images into base and detail layers. The scene ambience is captured in the base layer of the nonflash image and details are captured in the detail layer of the flash image. In this mode, one grid
is configured to perform bilateral filtering on the flash image and the other to perform cross-bilateral filtering, given by

$$I_{CB}(p) = \sum_{n=1}^{N} G_S(n) G_I(p-n) I_I(p-n) I_{NF}(p-n)$$

(10)

on the nonflash image using the flash image. The location of the grid cell is determined by the flash image and the intensity value is determined by the nonflash image. The image taken with flash contains shadows that are not present in the nonflash image. A shadow correction module is implemented which merges the details from the flash image with base layer of the cross-bilateral filtered nonflash image and corrects for the flash shadows to avoid artifacts in the output image. A mask representing regions with high details in the filtered nonflash image is created, as shown in Fig. 14. Gradients are computed at each pixel for blocks of 4 x 4 pixels. If the gradient at a pixel is higher than the average gradient for that block, the pixel is assigned as an edge pixel. This results in a binary mask that highlights all of the strong edges in the scene but no false edges due to the flash shadows. The details from the flash image are added to the filtered nonflash image, as shown in Fig. 15, only in the regions represented by the mask. A linear filter is used to smooth the mask to ensure that the resulting image does not have discontinuities. This implementation of the shadow correction module handles shadows effectively to produce LLE images without artifacts.

Fig. 16 shows a set of input flash and nonflash images and the LLE output image. The enhanced output effectively reduces noise while preserving details and corrects for flash shadows without creating artifacts.

V. LOW-VOLTAGE OPERATION

The energy consumed by a digital circuit can be minimized by operating at the optimal $V_{DD}$, which requires the ability to operate at low voltage. Random dopant fluctuation (RDF) is a dominant source of local variation at low voltage, causing random, local threshold voltage shifts [22]. To maintain sufficient reliability and performance at low voltage, significant attention needs to be given to the effects of local variation.

Performance of logic circuits is highly sensitive to variation in $V_T$ in this region of operation and can also result in functional failures at the extremes of $V_T$ variation. To quantify the functionality of a combinational cell at low voltage, we use the standard cell characterization approach described in [23]. A subset of standard cells from the 40-nm CMOS logic library are analyzed to ensure functionality and quantify the performance at 0.5 V. Standard cells that fail the functionality or do not satisfy
the performance requirement are not used in the design. The functionality and setup/hold performance of flip-flops are also verified using the cell characterization approach.

At nominal voltage, local variations in $V_T$ may result in 5%–10% variation in the logic timing. However, at low voltage, these variations can result in timing path delays with standard deviation comparable to the global corner delay and must be accounted for during timing closure in order to ensure a robust, manufacturable design. The probability density function (PDF) of delay at 0.5 V for a representative path from the design is shown in Fig. 18. The global corner delay for this path is 21.9 ns but, after accounting for the local variations the $3\sigma$ delay, becomes 36.1 ns.

At nominal voltage, paths that fail the setup/hold requirement are determined using the corner-based analysis and timing closure is achieved by performing setup/hold fix on these paths. However, at low voltage, it is not possible to consider only the paths that fail the setup/hold requirement in the corner analysis and determine their $3\sigma$ setup/hold performance, since a path with larger corner delay need not have a larger stochastic variation. We use the nonlinear operating point analysis (OPA) approach [23] to perform timing analysis at $V_{DD} = 0.5$ V. The potentially critical paths in the design, in presence of variations, are determined by the approach described in [23] and $3\sigma$ setup and hold performance is computed at $V_{DD} = 0.5$ V using OPA.

The three-step approach is summarized below.

1) All paths are analyzed with traditional static timing analysis (STA). $+/−3\sigma$ delay (corner delay plus the stochastic $+/−3\sigma$ delay) is used for each cell in the path. This is a pessimistic analysis, so those paths that pass the setup/hold requirement can be removed from further consideration.

2) Paths that fail the first step are reanalyzed. OPA-based analysis is applied to the clock paths to determine their $3\sigma$ delay.

Fig. 16. Input images: (a) image with flash and (b) image without flash. Output image: (c) LLE image.

Fig. 17. Input images: (a) image with flash and (b) image without flash. Output images: (c) filtered nonflash image and (d) LLE image.

Fig. 18. Delay PDF of a representative timing path from the design at 0.5 V. The global corner delay is 21.9 ns, and the $3\sigma$ delay, after accounting for local variations, is 36.1 ns.
TABLE I
SETUP/HOLD TIMING ANALYSIS AT 0.5 V

| Phase | Data Path   | Clock Path | Paths Analyzed | Worst Slack (ns) | % Fail |
|-------|-------------|------------|----------------|------------------|--------|
|       | STA (+3σ)  | STA (+3σ)  | 95k            | -10.7            | 3.6%   |
|       | STA (+3σ)  | OPA        | 3.4k           | -2.9             | 1.5%   |
|       | OPA        | OPA        | 52             | -0.05            | 13.4%  |

Paths requiring fixing (before timing closure) 7

Hold Analysis

| Phase | Data Path   | Clock Path | Paths Analyzed | Worst Slack (ns) | % Fail |
|-------|-------------|------------|----------------|------------------|--------|
|       | STA (−3σ)  | STA (+3σ)  | 95k            | -8.2             | 2.8%   |
|       | STA (−3σ)  | OPA        | 2.7k           | -1.8             | 2.4%   |
|       | OPA        | OPA        | 65             | -0.13            | 13.8%  |

Paths requiring fixing (before timing closure) 9

The functionality and timing characterization for standard cells and the OPA analysis for timing paths ensure reliable functionality, despite statistical variations, with 3σ confidence at 0.5 V.

VI. MEASUREMENT RESULTS

The test chip, shown in Fig. 19, is implemented in 40-nm CMOS technology and verified to be operational from 25 MHz at 0.5 V to 98 MHz at 0.9 V with SRAMs operating at 0.9 V.
This chip is designed to function as an accelerator core as part of a larger microprocessor system, utilizing the system’s existing DRAM resources. For standalone testing of this chip, a 32-b-wide 266-MHz DDR2 memory controller was implemented using a Xilinx XC5VLX50 FPGA. The performance versus energy tradeoff of the test chip for a range of $V_{DD}$ is shown in Fig. 20. The processor is able to operate from 25 MHz at 0.5 V with 2.3-mW power consumption to 98 MHz at 0.9 V with 17.8-mW power consumption. The run-time scales linearly with the image size with 13-megapixel/s throughput.

Table II shows a comparison of the processor performance with implementations on other mobile processors. Software that replicates the functionality of the testchip and maintains identical image quality is implemented on the mobile processors. The implementations are optimized for multithreading and multicore processing. Processing runtime and power consumption during software execution are measured. The processor achieves more than 5.2× faster performance than the fastest software implementation and consumes less than 40× power compared with the most power efficient one, resulting in an energy reduction of more than 280× compared with software implementations on some of the recent mobile processors while maintaining the same output image quality. Flexible bit-width computations, along with a high amount of parallelism and pipelining, enable an optimized processor implementation that achieves higher performance at a lower frequency and significant improvement in energy efficiency compared with software implementations.

The processor is integrated, as shown in Fig. 21, with a camera and a display through a host PC using the USB interface. A software application, running on the host PC, is developed for processor configuration, image capture, activating processing, and result display. The system, shown in Fig. 22, provides a portable platform for live computational photography.

VII. CONCLUSION

We have described the development and implementation of a reconfigurable processor for computational photography using 40-nm CMOS technology. The processor performs HDR imaging, low-light enhancement, and glare reduction using a reconfigurable bilateral grid. Highly parallel architecture enables real-time processing of HD images while operating at less than 100 MHz. The ability to operate at low supply voltages is important for maximizing energy efficiency. Circuit design for low-voltage operation ensures reliable performance down to 0.5 V. The processor achieves 280× energy reduction compared with software implementations on recent mobile processors. The energy-scalable implementation proposed in this work enables efficient integration into portable multimedia devices for real-time computational photography.

ACKNOWLEDGMENT

The authors would like to thank the TSMC University Shuttle Program for chip fabrication and Prof. F. Durand and J. Regan-Kelley for valuable feedback and suggestions.

REFERENCES

[1] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dynamic-range images,” ACM Trans. Graphics, vol. 21, no. 3, pp. 257–266, Jul. 2002.
[2] G. Petschnigg, M. Agrawala, H. Hoppe, R. Szeliski, M. Cohen, and K. Toyama, “Digital photography with flash and no-flash image pairs,” *ACM Trans. Graphics*, vol. 23, no. 3, pp. 664–672, Aug. 2004.

[3] E. Eisemann and F. Durand, “Flash photography enhancement via intrinsic relighting,” *ACM Trans. Graphics*, vol. 23, no. 3, pp. 673–678, Aug. 2004.

[4] M. Brown and D. G. Lowe, “Automatic panoramic image stitching using invariant features,” *Int. J. Comput. Vis.*, vol. 74, no. 1, pp. 59–73, Aug. 2007.

[5] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, “Efficient marginal likelihood optimization in blind deconvolution,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 2657–2664.

[6] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan, “Light-field photography with a handheld plenoptic camera,” *Stanford Univ. Comput. Sci. Tech. Rep. CSTR 2005-02*, Apr. 2005.

[7] G. Wan, X. Li, G. Agranov, M. Levoy, and M. Horowitz, “CMOS image sensors with multi-bucket pixels for computational photography,” *IEEE J. Solid-State Circuits*, vol. 47, no. 4, pp. 1031–1042, Apr. 2012.

[8] S. Sukegawa, T. Umebayashi, T. Nakajima, H. Kawanobe, K. Koseki, I. Hirono, T. Haruta, M. Kasai, K. Fukumoto, T. Wakanos, K. Inoue, H. Takahashi, N. Yagami, T. Hira, and N. Watanabe, “A 1/4-inch 8 Mpixel back-illuminated stacked CMOS image sensor,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2013, pp. 484–485.

[9] Y. Chen, Y. Xu, Y. Chae, A. Mierop, X. Wang, and A. Theuwissen, “A 0.7 e-rms-temporal-readout-noise CMOS image sensor for low-light-level imaging,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2012, pp. 384–385.

[10] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in *Proc. IEEE Conf. Comput. Vis.*, Jan. 1998, pp. 839–846.

[11] S. Bae, S. Paris, and F. Durand, “Two-scale tone management for photographic look,” *ACM Trans. Graphics*, vol. 25, no. 3, pp. 637–645, Jul. 2006.

[12] E. Bennet and L. McMillan, “Video enhancement using per-pixel virtual exposures,” *ACM Trans. Graphics*, vol. 24, no. 3, pp. 845–852, Jul. 2005.

[13] J. Xiao, H. Cheng, H. Awbney, C. Rao, and M. Isordi, “Bilateral filtering based optical flow estimation with occlusion detection,” in *Proc. Eur. Conf. Comput. Vis.*, 2006, pp. 211–224.

[14] J.-H. Woo, J.-H. Sohn, H. Kim, and H.-J. Yoo, “A 195 mW, 9.1 MVertices/s fully programmable 3-D graphics processor for low-power mobile devices,” *IEEE J. Solid-State Circuits*, vol. 43, no. 11, pp. 2370–2380, Nov. 2008.

[15] F. Sheikh, S. K. Mathew, M. A. Anders, H. Kaul, S. K. Hsu, A. Agarwal, R. K. Krishnamurthy, and S. Borkar, “A 2.05 GVertices/s 151 mW lighting accelerator for 3D graphics vertex and pixel shading in 32 nm CMOS,” *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 128–139, Jan. 2013.

[16] T. Burd and R. Broderson, “Design issues for dynamic voltage scaling,” in *Proc. IEEE Symp. Low Power Electron. Design*, 2000, pp. 9–14.

[17] B. H. Calhoun and A. P. Chandrakasan, “Characterizing and modeling minimum energy operation for subthreshold circuits,” in *Proc. IEEE Int. Symp. Low Power Electron. Design*, 2004, pp. 90–95.

[18] J. Chen, S. Paris, and F. Durand, “Real-time edge-aware image processing with the bilateral grid,” *ACM Trans. Graphics*, vol. 26, no. 3, Jul. 2007, article 103.

[19] S. Paris and F. Durand, “A fast approximation of the bilateral filter using a signal processing approach,” *Int. J. Comput. Vis.*, vol. 81, no. 1, pp. 24–52, Jan. 2009.

[20] B. Weiss, “Fast median and bilateral filtering,” *ACM Trans. Graphics*, vol. 25, no. 3, pp. 519–526, Jul. 2006.

[21] P. E. Debevec and J. Malik, “Recovering high dynamic range radiance maps from photographs,” in *Proc. ACM Conf. Comput. Graphics and Interactive Tech.*, Aug. 1997, pp. 369–378.

[22] B. Zhai, S. Hanson, D. Blauuw, and D. Sylvester, “Analysis and mitigation of variability in subthreshold design,” in *Proc. IEEE Int. Symp. Low Power Electron. Design*, 2005, pp. 20–25.

[23] R. Rithe, S. Chao, J. Gu, A. Wang, S. Datla, G. Gammie, D. Buss, and A. Chandrakasan, “The effect of random dopant fluctuations on logic timing at low voltage,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 5, pp. 911–924, May 2012.

[24] Intel Atom Processor Z2760. [Online]. Available: http://www.intel.com/content/www/us/en-processors/atom/atom–z2760-datasheet.html

[25] DragonBoard Snapdragon S4 Plus APQ8060A Mobile Development Board. [Online]. Available: https://developer.qualcomm.com/mobile-development/devices/dragonboard

[26] Samsung Exynos 5 Dual Arndale Board. [Online]. Available: http://www.arduino-board.org/wiki/index.php/Main_Page

[27] PandaBoard: Open OMAP 4 Mobile Software Development Platform. [Online]. Available: http://pandaboard.org/content/platform

Rahul Rithi (S’08) received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 2008, and the S.M. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2010, where he is currently working toward the Ph.D. degree.

His research interests include low-power integrated circuits and energy-efficient systems for portable multimedia applications.

Mr. Rithi was the recipient of the President of India Gold Medal in 2008 and the MIT Presidential Fellowship in 2008.

Priyanka Raina (S’11) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 2011, and the S.M. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2013, where she is currently working toward the Ph.D. doctoral degree.

Her research interests include design of low-power circuits for computational photography applications.

Nathan Ickes (M’11) received the S.B., M.Eng., and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2001, 2002, and 2008, respectively.

He is currently a Research Scientist with the Digital Integrated Circuits and Systems Group, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include micropower digital circuits and system architectures, software support for power management, high-reliability and fault tolerance, and biomedical applications of micropower systems.

Srikanth V. Tenneti (S’13) received the B.Tech degree in electrical engineering from the Indian Institute of Technology, Bombay, India, in 2012. He is currently working toward the Ph.D. degree in electrical engineering at the California Institute of Technology, Pasadena, CA, USA.

His research interests include sub-Nyquist sampling schemes for parametric analog signals. He has worked with the Digital Integrated Circuits and Systems Group at the Massachusetts Institute of Technology, Cambridge, MA, USA, and the Signal Processing and Artificial Neural Networks (SPANN) Lab at the Indian Institute of Technology, Bombay, India, in the past.

Mr. Tenneti was the recipient of Best Undergraduate Thesis Award in Electrical Engineering from the Indian Institute of Technology, Bombay, in 2012.
Anantha P. Chandrakasan (F’04) received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, CA, USA, in 1989, 1990, and 1994, respectively.

Since September 1994, he has been with the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, where he is currently the Joseph F. and Nancy P. Keithley Professor of Electrical Engineering. He was the Director of the MIT Microsystems Technology Laboratories from 2006 to 2011. Since July 2011, he is the Head of the Electrical Engineering and Computer Science Department, MIT. He is a coauthor of Low Power Digital CMOS Design (Kluwer Academic, 1995), Digital Integrated Circuits (Pearson Prentice-Hall, 2003, 2nd ed.), and Sub-threshold Design for Ultra-Low Power Systems (Springer 2006). He is also a coeditor of Low Power CMOS Design (IEEE, 1998), Design of High-Performance Microprocessor Circuits (IEEE, 2000), and Leakage in Nanometer CMOS Technologies (Springer, 2005). His research interests include micro-power digital and mixed-signal integrated circuit design, wireless microsensor system design, portable multimedia devices, energy efficient radios and emerging technologies.

Dr. Chandrakasan was a coreipient of several awards, including the 1993 IEEE Communications Society’s Best Tutorial Paper Award, the IEEE Electron Devices Society’s (EDS) 1997 Paul Rappaport Award for the Best Paper in an EDS publication during 1997, the 1999 DAC Design Contest Award, the 2004 DAC/ISSCC Student Design Contest Award, the 2007 ISSCC Beatrice Winner Award for Editorial Excellence, and the ISSCC Jack Kilby Award for Outstanding Student Paper (2007, 2008, 2009). He received the 2009 Semiconductor Industry Association (SIA) University Researcher Award. He was the recipient of the 2013 IEEE Donald O. Pederson Award in Solid-State Circuits. He has served as a technical program co-chair for the 1997 International Symposium on Low Power Electronics and Design, VLSI Design’98, and the 1998 IEEE Workshop on Signal Processing Systems. He was the Signal Processing Sub-committee Chair for ISSCC 1999–2001, the Program Vice-Chair for ISSCC 2002, the Program Chair for ISSCC 2003, the Technology Directions Sub-Committee Chair for ISSCC 2004–2009, and the Conference Chair for ISSCC 2010–2012. He is the Conference Chair for ISSCC 2013. He was an associate editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS from 1998 to 2001. He served on the IEEE Solid-State Circuits Society AdCom from 2000 to 2007 and he was the meetings committee chair from 2004 to 2007.