# An SRAM-Based Accelerator for Solving Partial Differential Equations

Thomas Chen, Jacob Botimer, Teyuh Chou, and Zhengya Zhang
University of Michigan, Ann Arbor, MI, USA

***Abstract*—Accurate numerical solutions of partial differential equations (PDE) require high-precision fine-grid Jacobi iterations that are demanding in both computation and memory. To reduce the precision and memory, we reformulate the multi-grid Jacobi method in a residual form to enable the mapping of a high-precision PDE solver on SRAMs that perform low-precision parallel multiply-accumulates (MAC) in memory, reducing both energy and area. To improve performance, we employ a DLL to generate well-controlled unit pulses for driving word lines and a dual-ramp single-slope ADC to convert bit line outputs. The design is prototyped in a 1.87mm$^2$ 180nm test chip made of four 320×64 MAC-SRAMs, each supporting 128× parallel 5b×5b MACs with 32 5b output ADCs and consuming 16.6mW at 200MHz. The test chip is demonstrated to reach an error tolerance of 10$^{-8}$ in solving PDEs at 56.9GOPS.**

## I. Introduction

Many physical phenomena, such as heat and fluid dynamics, are described by partial differential equations (PDE). Most PDEs are solved numerically, by first quantizing the solution space in a grid and then iteratively refining the solution using the Jacobi method.

High-precision PDE solutions require fine grids and high numerical precision, leading to a significant amount of data that need to be processed, moved and stored. Moreover, a PDE solver commonly requires tens of thousands of iterations to converge. Digital PDE solvers have been proposed [1], [2], but scaling up the solvers to support practical problems is a challenge, due to the difficulties of providing a high compute density and a high memory bandwidth.

To enable a more efficient and practical PDE accelerator design, we apply two algorithm approaches: 1) we adopt a multi-grid method that divides the PDE solver to a fine-grid compute and a coarse-grid compute and iterates between the two to accelerator convergence by 5 to 10× on average; 2) we transform both fine-grid compute and coarse-grid compute to a residual form, lowering the precision to 5b for a low error tolerance below 10$^{-8}$.

Even with a faster convergence and a much-reduced precision, the implementation cost can still be high using a conventional digital approach. Recently, process in memory (PIM) has been proposed as a new technique that computes directly on a large array of data in place, within memory [3], [4]. SRAM and RRAM are the popular choices for PIM. In particular, SRAM-based PIM designs are closer to reality than RRAM. It relies on level- and/or width-modulating word lines (WL) of the SRAM array to encode multipliers, and activating multiple WLs in parallel [3]–[6]. The SRAM cells' currents in discharging the bit lines (BL) represent the products, and the current on each BL represents the sum of products. Alternatively, BLs can be modulated to encode multipliers, and BLs are joined to produce the sum of products [7].

Current SRAM-based PIM designs are limited by SRAM's binary storage and the overhead of multi-bit A/D conversion. Some designs support only binary multiplicands [3], [6], [7]; and some choose binary outputs [3], [6]. To reduce the number of ADCs, some designs are tailored to computations in a cone structure that require only one or a small number of ADCs at the final output [4], [5]. These approaches are not applicable to a PDE solver that requires iterative multi-bit operations and solutions to be updated in every iteration.

We present a MAC-SRAM that supports 5b×5b MACs with full-bandwidth 5b outputs to support a PDE solver. The PDE solver is successfully mapped to the MAC-SRAMs to take advantage of the high compute density and energy efficiency. We design a DLL-based 5b driver that produces WL pulses down to $\frac{1}{8}$ of a clock period with PVT tolerance. The WL pulses are level-modulated to match 5b multiplicands stored in SRAM. We employ a dual-ramp single slope ADC [8] that employs a coarse ramp followed by a fine ramp to increase conversion speed and minimize area. A 1.87mm$^2$ 180nm chip consisting of four 320×64 MAC-SRAMs is demonstrated at 200MHz, each providing 1.42G MAC/s and 32 5b ADCs at a power consumption of 16.6mW.

## II. Numerical PDE Solver Design

We use the solution to 2D Poisson's equation, shown in (1), to explain the PDE solver design. Poisson's equation is widely used in practical applications.

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = b, \tag{1}$$

where $b(x, y)$ is given and $u(x, y)$ is sought. The equation can be solved by the finite difference method that approximates $u$ and $b$ in a $M \times N$ grid of step size $\Delta x$ and $\Delta y$ along $x$ and $y$. The approximation results in a system of $MN$ equations:

$$\frac{u_{i-1,j} + u_{i+1,j} - 2u_{i,j}}{\Delta x^2} + \frac{u_{i,j-1} + u_{i,j+1} - 2u_{i,j}}{\Delta y^2} = b_{i,j}, \tag{2}$$
$$2 \le i \le M-1, 2 \le j \le N-1,$$

where $u_{i-1,j}$ is the value of $u$ at grid position $(i-1, j)$, $u_{i+1,j}$ is the value of $u$ at $(i+1, j)$, and so on. A 5-point stencil is assumed. In the matrix form, the $u$ values are put in a $MN \times 1$ vector $\mathbf{u}$, and similarly the $b$ values are put in a

* 978-1-5386-9395-7/19/$31.00 ©2019 IEEE

Authorized licensed use limited to: University of Michigan Library. Downloaded on March 21,2021 at 16:10:48 UTC from IEEE Xplore. Restrictions apply.

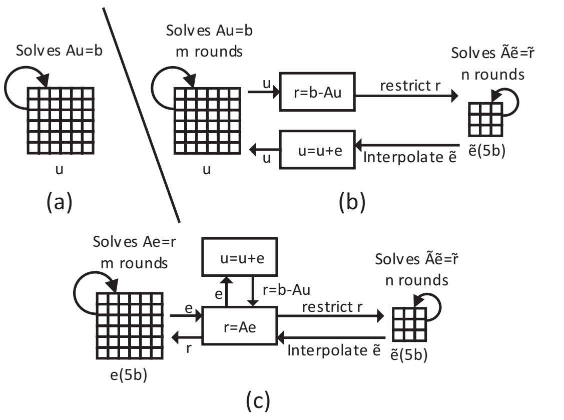

Fig. 1. PDE mapping approaches: (a) fine-grid only; (b) multi-grid with residuals on coarse grid; (c) multi-grid with residuals on both grids.

$MN \times 1$ vector $\mathbf{b}$. The coefficients of $u$ in $MN$ equations are put in a $MN \times MN$ matrix $\mathbf{A}$, where each row corresponds to one equation in (2). Solving the PDE is transformed to solving $\mathbf{Au} = \mathbf{b}$. In general, a $p$-point stencil, the density of nonzero values in $\mathbf{A}$ is approximately $\frac{p}{MN}$. Since $M$ and $N$ are usually large, $\mathbf{A}$ is highly sparse. Higher-order PDEs use higher-order grids, and the solutions follow the same formulation.

## A. Low-Precision Residual Approach

To reach an accurate solution, a fine grid of fine step sizes is required, as shown in Fig. 1(a). However, a fine grid results in a large volume of data and a relatively slow convergence. To speed up convergence, the multi-grid method introduces a $\tilde{M} \times \tilde{N}$ ($\tilde{M} < M$, $\tilde{N} < N$) coarse grid in addition to the fine grid, as shown in Fig. 1(b). By interleaving coarse-grid iterations with fine-grid iterations, low-frequency errors are reduced to accelerate convergence.

As graphically illustrated in Fig. 1(b), to transition from fine grid to coarse grid, restriction is applied to project the residual $\mathbf{r}$ to the coarse grid. To transition from coarse grid back to fine grid, interpolation is applied to project $\tilde{\mathbf{e}}$ to the fine grid. Because coarse-grid compute operates on errors, the bitwidth is reduced. Realizing the potential benefit of the residual approach, we extend it to fine-grid compute, as shown in Fig. 1(c). The approach is described as follows.

$$\begin{aligned}
\texttt{Residual:}\ & \mathbf{r} = \mathbf{b} - \mathbf{Au}^{(n)}, \qquad (3)\\
\texttt{Fine grid:}\ & \texttt{solve}\ \mathbf{Ae} = \mathbf{r},\\
& \mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \mathbf{e},\\
\texttt{Coarse grid:}\ & \texttt{solve}\ \tilde{\mathbf{A}}\tilde{\mathbf{e}} = \tilde{\mathbf{r}},\ \tilde{\mathbf{r}} = \texttt{restrict}(\mathbf{r}),\\
& \mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \texttt{interpolate}(\tilde{\mathbf{e}}),
\end{aligned}$$

where $\tilde{\mathbf{A}}$ is the $\tilde{M}\tilde{N} \times \tilde{M}\tilde{N}$ coefficient matrix for the coarse grid, and $\tilde{\mathbf{e}}$ and $\tilde{\mathbf{r}}$ are both $\tilde{M}\tilde{N} \times 1$ vectors.

To solve $\mathbf{Ae} = \mathbf{r}$, $\mathbf{A}$ is separated to a diagonal term $\mathbf{D}$ and an off-diagonal term $\mathbf{R}$, i.e., $\mathbf{A} = \mathbf{D} + \mathbf{R}$, and then $\mathbf{e}$ is solved iteratively by the Jacobi method as in (4).

$$\mathbf{e}^{(n+1)} = \mathbf{D}^{-1}(\mathbf{r} - \mathbf{Re}^{(n)}) = \mathbf{c} - \mathbf{Se}^{(n)}. \qquad (4)$$

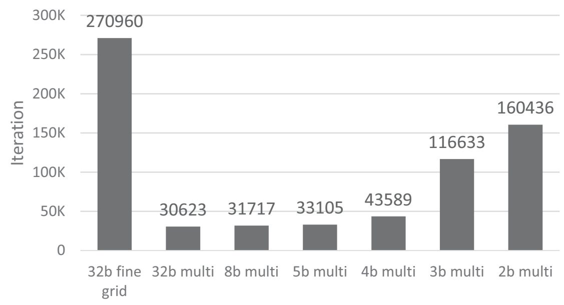

Fig. 2. Latency of solving 2D Poisson's equation to reach an error tolerance of $10^{-7}$ using a 249×249 single-grid and 32b, 8b, 5b, 4b, 3b, and 2b quantized multi-grids (a 249×249 fine grid combined with a 125×125 coarse grid).

Since $\mathbf{R}$ and $\mathbf{D}^{-1}$ are known or can be pre-computed and $\mathbf{r}$ does not change each iteration, $\mathbf{c}$ and $\mathbf{S}$ can be pre-computed. Consequently, the core computation is the matrix-vector product $\mathbf{Se}^{(n)}$. Since $\mathbf{A}$ is highly sparse, the stencil matrix $\mathbf{S}$ is also highly sparse. $\tilde{\mathbf{A}}\tilde{\mathbf{e}} = \tilde{\mathbf{r}}$ is solved iteratively using the same method.

In Fig. 2, we compare the latency of solving the 2D Poisson's equation to reach an error tolerance of $10^{-7}$ using a 32b single fine-grid and 32b, 8b, 5b, 3b and 2b quantized multi-grids. Moving from the 32b single-grid to the 32b multi-grid reduces the latency by almost 9×. As the multi-grid bitwidth is reduced, the latency increases. The optimal quantization centers around 5b or 4b, where the latency increases by only 8% and 42%, respectively. Although only one example is shown, the observations are generally applicable.

## B. Mapping of PDE Solver on MAC-SRAM

We adopt the low-precision residual approach to accelerate the PDE solver by mapping the 5b core computation $\mathbf{Se}^{(n)}$ to MAC-SRAMs. The stencil matrix $\mathbf{S}$ is highly sparse, making it inefficient to be stored in SRAM. Instead, we store the errors $\mathbf{e}$ in SRAM with each 5b value stored in 5 cells in consecutive rows, and 5b stencil weights are applied as WL pulses to the SRAM. The MAC outputs are converted to 5b digital values.

In a typical PIM design, all rows of the memory are activated at the same time to unleash the full parallelism [3], [6]. However, in a PDE solver, $\mathbf{S}$ is sparse, so we apply only a set of non-zero stencil entries at a time. For example, in solving the 2D Poisson's equation in (2) using a 5-point stencil, 4 stencil entries (diagonal term is excluded) are applied at the same time. Activating a subset of rows of the memory also reduces the precision of the BL outputs and simplifies the ADC design.

## III. Prototype Architecture

A prototype PDE solver chip is designed in 180nm CMOS based on 4 MAC-SRAMs, and each is a 320×64 8T SRAM array that is completed with peripherals. The 4 MAC-SRAMs can be used to compute 4 independent grids of up to 64×64 (5b grid values), or they can be joined to support a grid of up to 127×127 (5b grid values). The precision is configurable

Authorized licensed use limited to: University of Michigan Library. Downloaded on March 21,2021 at 16:10:48 UTC from IEEE Xplore. Restrictions apply.

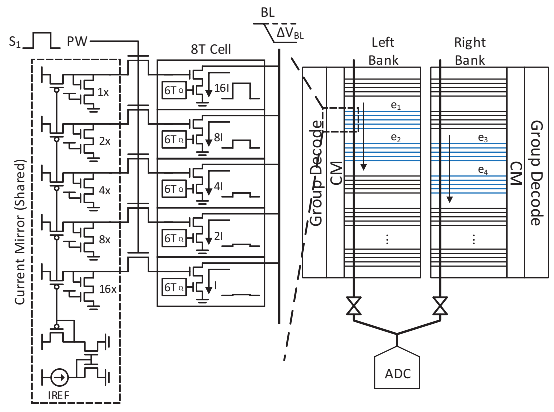

Fig. 3. Illustration of group read for MAC operations.

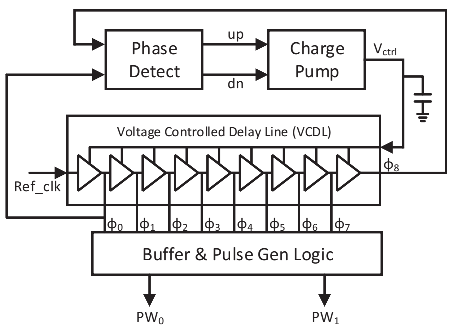

Fig. 4. DLL design for WL pulse generation.

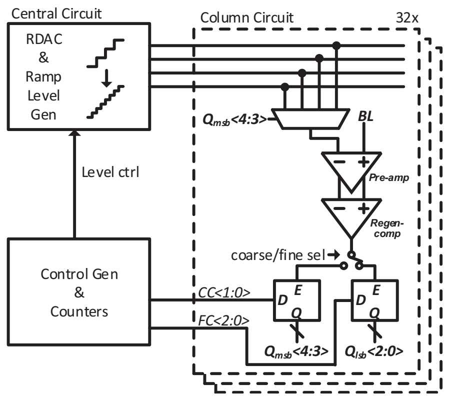

Fig. 5. 5-bit dual-ramp single-slope ADC design for BL readout.

from 1b to 5b. A buffer is added to each MAC-SRAM to store grid values between iterations. Offset subtraction is performed at the output of each MAC-SRAM, and a separate memory is used to store the offsets **c**.

A 320×64 MAC-SRAM is made of two 320×32 SRAM banks in a folded structure as shown in Fig. 3. The MAC-SRAM occupies 0.467mm$^2$ in 180nm CMOS and is clocked at 200MHz. It supports two modes: a single read/write mode for normal memory access, and a group read mode for MAC operations. In the group read mode, up to 20 rows (i.e., four 5b groups) are selected in parallel by the group decoder. 5b width-modulation of WL is controlled by a DLL, and 5b level-modulation is done via current mirrors. Merge and select muxes allow analog summation of partial sums. The 32 merged BLs are digitized by 32 5b ADCs.

## IV. Group Read and Word Line Pulse Generation

The group read mode is illustrated in Fig. 3, showing 4 stencil entries $(S_1, S_2, S_3, S_4)$ applied to 4 error vectors $(e_1, e_2, e_3, e_4)$ stored in 20 rows of the SRAM banks for MAC operations. A 5b error value is stored across 5 SRAM cells in a group. The MAC operations are conducted in groups in the following manner: 1) the group decoder turns on the access to a group of 5 SRAM rows; 2) the WL pulse width (PW) is selected by a 5b stencil entry; 3) current mirrors generate the WL voltage needed to provide 1×, 2×, 4×, 8×, and 16× unit cell current for the analog readout of 5b error values; and 4) the products between the stencil entry and the error values are accumulated on the BLs. Up to 4 groups are activated at the same time to enable 128 5b×5b MACs in parallel in the MAC-SRAM.

To generate fine and well-controlled WL pulses, we design a DLL to generate 8 phases of the 5ns clock period using an 8-stage voltage-controlled delay line in a control loop shown in Fig. 4. The phases are continuously adjusted by tracking the 200MHz reference clock using the phase detector, and errors are corrected by the control voltage of the delay line. The DLL occupies 1,500$\mu$m$^2$ in 180nm and consumes 950$\mu$W. It provides the unit PW with a maximum tracking error of 12ps under PVT variations, which is more robust than an open-loop approach [7].

## V. Bit Line Readout

We adopt a compact dual-ramp single-slope (DRSS) ADC architecture [8] that applies a 2b coarse-ramp comparison followed by a 3b fine-ramp comparison, as shown in Fig. 5. The BL voltage is first compared with the 2b coarse ramp to obtain the 2b MSB, which then selects one of four 3b fine ramps for comparison to obtain the 3b LSB. The dual-ramp approach performs 5b comparison in $2^2 + 2^3 = 12$ time steps, faster than a serial conversion architecture [7].

In implementing DRSS ADCs, a central circuit is shared by 32 columns and it generates two ramps by a resistive DAC. A compact column circuit consists of a pre-amplifier followed by a regenerative comparator and latches. The 32 ADCs in a MAC-SRAM occupy 0.044mm$^2$ and the conversion costs 8.91mW at 200MHz. Under PVT variations, the DNL of the ADCs is kept below 0.5b to ensure accurate 5b conversion even without any calibration.

## VI. Results and Comparison

A 180nm 11.0mm$^2$ PDE solver test chip was fabricated and tested. The chip consists of a PDE solver and BIST circuits, as shown in Fig. 6. The 4 MAC-SRAMs in the PDE solver core each takes 570$\mu$m×820$\mu$m and dissipates

Authorized licensed use limited to: University of Michigan Library. Downloaded on March 21,2021 at 16:10:48 UTC from IEEE Xplore. Restrictions apply.

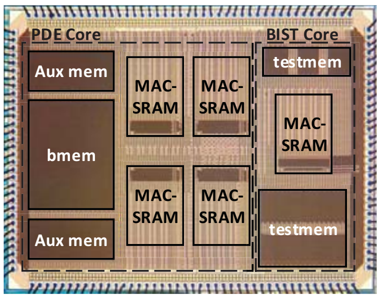

Fig. 6. Microphotograph of PDE solver test chip.

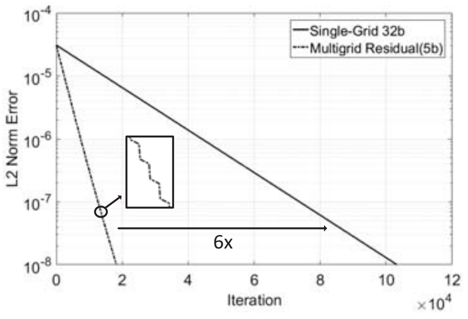

Fig. 7. Convergence of 5b multi-grid (5b fine-grid: 127×127, 5b coarse-grid: 64×64) compared to single-grid baseline implementation (32b fine-grid: 127×127). Results are based on solving 2D Poisson's equation.

16.6mW when performing group read at 200MHz and room temperature. When running Jacobi iterations, the 5b multi-grid PDE solver reaches an error tolerance of $10^{-8}$ while speeding up convergence by 6× over the baseline 32b single-grid implementation, as shown in Fig. 7.

The 200MHz MAC-SRAM completes 128 5b×5b MAC operations in 18 clock cycles (4-cycle WL pulse, 1-cycle BL propagation, 12-cycle ADC and 1-cycle latching). With 4 MAC-SRAMs, the PDE solver chip performs 512 5b×5b MAC operations every 18 clock cycles. Following [7] that counts an operation at each active SRAM cell as 2 OPs, the performance and energy of each MAC-SRAM are 14.2GOPS and 857GOPS/W, respectively. At a lower precision, the performance and energy efficiency can be more than doubled, as shown in Fig. 8. We synthesized a comparable digital ASIC

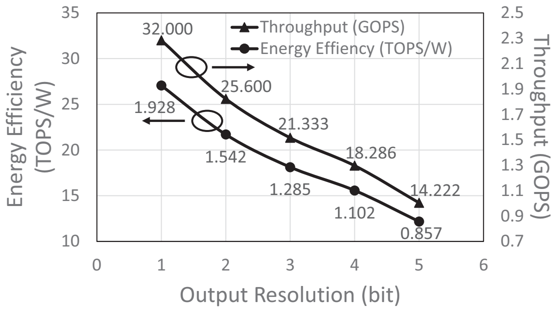

Fig. 8. Measured performance and energy.

TABLE I
COMPARISON WITH STATE-OF-THE-ART PIM USING SRAM (UNNORMALIZED)

| | IMCORE [5] | Conv-RAM [7] | This work |
|---|---|---|---|
| Application | SVM | CNN | PDE solver |
| Technology | 65nm | 65nm | 180nm |
| Core voltage | 0.925V | 1.2V | 1.8V |
| Core area (mm$^2$) | 0.52 | 0.067 | 1.868 |
| Memory size | 512×256 | 16×16×64 | 4×320×64 |
| PIM kernel precision | 4b×8b | 7b×1b | **5b×5b** |
| ADC resolution | 4b | 7b | 5b |
| Number of ADCs | 1 | 16 | **128** |
| Activated cells | 4×256 | 16×1×64 | 4×20×32 |
| Latency (ns) | 31 | 150 | 90 |
| Performance (GOPS[a]) | 66.2[b] | 10.7[c] | 56.8 |
| Power (mW) | 3.17 | 0.381 | 66.4 |
| Efficiency (TOPS/W) | 20.9 | 28.1 | 0.857 |
| Density (GOPS/mm$^2$) | 127 | 160 | 30.5 |

[a]An actived cell is counted as 2 OPs [7]. [b]In test mode [5].
[c]Activates 50×16 cells and uses 4b output for testing [7].

in 180nm CMOS, but its compute density is 40× lower than this work.

This design is the first PIM that targets solving PDEs. Prior PIM designs do not meet the requirements of the PDE solver due to limited multiplicand precision [3], [6], [7], limited ADC resolution [3], [6], or limited number of ADCs [4], [5]. In Table I, we attempt to compare the PDE solver chip with two 65nm PIM designs, IMCORE [5] and Conv-RAM [7]. Our MAC-SRAM provides a higher multiplicand precision than Conv-RAM and substantially more ADCs than IMCORE and Conv-RAM to support iterative solution updates. As a result, the power efficiency measured in TOPS/W is lower. Note that the work was prototyped in a 180nm technology and the results in Table I are not normalized. We expect that the energy efficiency and compute density of this design to be substantially improved in newer technologies.

## ACKNOWLEDGMENT

This work was supported in part by DARPA through award HR0011-17-2-0018 and NSF GRFP DGE 1256260.

## REFERENCES

[1] Y. Huang *et al.*, "Evaluation of an analog accelerator for linear algebra," in *ISCA*, vol. 44, no. 3, 2016, pp. 570–582.
[2] J. Kung *et al.*, "A programmable hardware accelerator for simulating dynamical systems," in *ISCA*, 2017, pp. 403–415.
[3] J. Zhang *et al.*, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array," *JSSC*, vol. 52, no. 4, pp. 915–924, 2017.
[4] M. Kang *et al.*, "A 481pJ/decision 3.4 M decision/s multifunctional deep in-memory inference processor using standard 6T SRAM array," *arXiv preprint arXiv:1610.07501*, 2016.
[5] S. K. Gonugondla *et al.*, "A 42pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training," in *ISSCC*, 2018, pp. 490–492.
[6] W.-S. Khwa *et al.*, "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary dnn edge processors," in *ISSCC*, 2018, pp. 496–498.
[7] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *ISSCC*, 2018, pp. 488–490.
[8] M. F. Snoeij *et al.*, "A CMOS image sensor with a column-level multiple-ramp single-slope ADC," in *ISSCC*, 2007, pp. 506–618.

Authorized licensed use limited to: University of Michigan Library. Downloaded on March 21,2021 at 16:10:48 UTC from IEEE Xplore. Restrictions apply.