A 1000 frames/s Vision Chip Using Scalable Pixel-Neighborhood-Level Parallel Processing

Joseph A. Schmitz, Mahir K. Gharzai, Sina Balkır, Michael W. Hoffman, Daniel J. White, *Member, IEEE*, and Nathan Schemm

***Abstract*— This paper presents a novel vision chip architecture based on pixel-neighborhood-level parallel processing. In the architecture, an 8-b RISC processing core is embedded in an 8×8 array of digital pixel sensors on the same focal plane. These neighborhood processors (NPs) are tiled in a 2-D array to form the final imager resolution. Program execution is carried out in parallel across the array of pixel-neighborhood processing cores, allowing for direct scalability in terms of resolution, without reduction in processing speed or frame rate. To accomplish this, a compact, low-complexity NP architecture along with a general-purpose, 8-b instruction set has been designed and implemented. A prototype vision chip containing an 8×10 array of NPs with a 64×80 resolution has been designed and fabricated in a 0.13-μm 1P8M CMOS fabrication process. The system is reprogrammable and can perform a wide range of image and video processing tasks. Several example algorithms are implemented and tested on the single-chip vision system to demonstrate the functionality of pixel-neighborhood-level parallelism, including 1000-frames/s object tracking.**

***Index Terms*— CMOS image sensor, image processing, neighborhood processor (NP), object tracking, programmable, single instruction multiple data, vision chip.**

I. Introduction

INTEGRATION of sensing and processing in the context of CMOS image processing systems can generally be categorized into three levels: chip-level, column-level, and pixel-level processing. Each of these systems introduces processing elements (PEs) at various locations relative to the imager array, providing distinct levels of parallel processing and programmability to meet application requirements.

Chip-level processing has the advantage of providing the highest imaging quality and fill factor, high programming flexibility, but minimal parallelism [1], [2]. Previous work integrates a processing unit at the row or column level [3]–[8]. Pixel-level single instruction multiple data processing embeds compact PEs at each photodiode at the expense of decreased fill factor and resolution, but provides the greatest degree of parallelism and speed [9]–[15].

A further advance in pixel-level processors has been to design PEs that behave like general-purpose processors, having the programmability and flexibility to meet application requirements as well as their own memory and arithmetic capabilities [16]–[20]. Another category of vision chips has been developed that implement hybrid architectures, which combine two or more levels of processing, that aim to increase programmability and processing speed [21]–[35].

The authors present a novel vision chip architecture that demonstrates a balance between chip-, row/column-, and pixel-level image processing paradigms implemented in single-chip vision systems. The processing architecture is composed of neighborhoods of 8×8 digital pixel sensors (DPSs), where each neighborhood contains a physically embedded, general-purpose processing core. It is the first published example of this architecture, to the best of our knowledge.

Compared with existing methods, neighborhood-level processing is the only one that features performance that scales with resolution and is able to execute a wide range of algorithms. In contrast, row/column-level processors are responsible for increasing numbers of pixels under increasing resolution, while pixel-level processors are restricted to simple algorithms due to limited pixel memory. In neighborhood processing, by using the shared area between photodiodes, more complex PEs can be implemented with increased memory and functionality compared with pixel-level processors, while maintaining pixel pitch and speed.

Program execution is carried out in parallel on a 2-D array of pixel-neighborhood processors (NPs), allowing direct scalability in terms of resolution. A single-chip vision system formed from an array of these NPs can be programmed to perform a variety of image and video processing tasks. Fig. 1 compares the physical relationship between neighborhood-level processing architecture and other vision chip processing methods.

In this paper, a prototype vision chip with the aforementioned pixel-neighborhood-level parallel processing is presented. A 64×80 resolution imager formed from an 8×10 array of neighborhoods in a 0.13-μm fabrication process has been designed, fabricated, and tested. The architecture has individual register banks and memory for each NP core and does not rely on shared, global memory.

The choice of the neighborhood size is driven by two different but important considerations: 1) the spatial image

Manuscript received April 4, 2016; revised August 10, 2016 and September 9, 2016; accepted September 12, 2016. Date of publication October 12, 2016; date of current version January 30, 2017. This paper was approved by Associate Editor Dejan Marković. This work was supported in part by the MOSIS High Education Program and in part by the NASA Nebraska Space Grant Consortium.

J. A. Schmitz, M. K. Gharzai, S. Balkır, and M. W. Hoffman are with the Department of Electrical and Computer Engineering, University of Nebraska–Lincoln, Lincoln, NE 68588-0511 USA (e-mail: josephschmitz@unl.edu; kabeero@unl.edu; sbalkir@unl.edu; mhoffman1@unl.edu).

D. J. White is with the Gellersen Center, Valparaiso University, Valparaiso, IN 46383 USA (e-mail: dan.white@valpo.edu).

N. Schemm is with Texas Instruments Inc., Dallas, TX 75243 USA (e-mail: nathan.schemm@ti.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2016.2613094

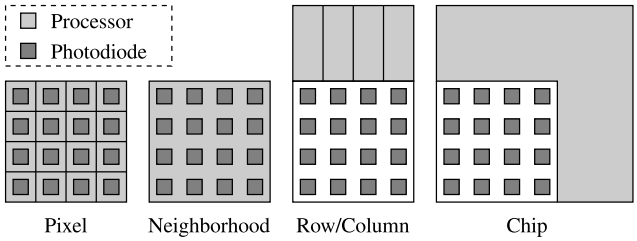

Fig. 1. Comparison of neighborhood-level processing to existing processing architectures for a 4 × 4 pixel array. From left to right: pixel-level, neighborhood-level, row/column-level, and chip-level processing.

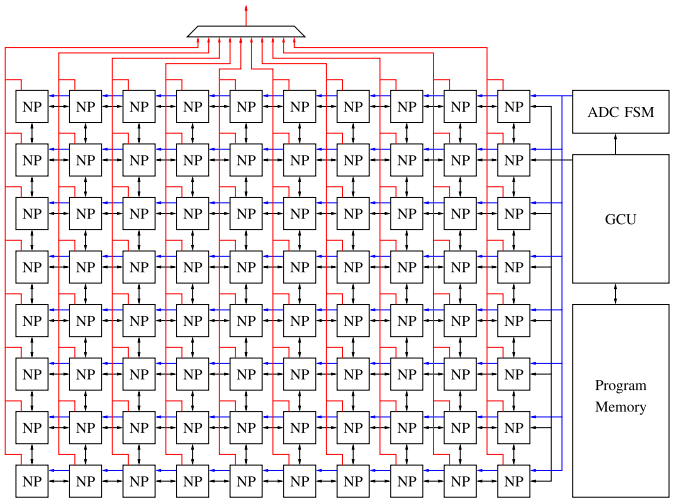

Fig. 2. Block diagram showing data movement between NPs, global program flow control signals, and global ADC control signals.

correlation properties typically encountered in image processing applications and 2) the necessary footprint for supporting the functionality of a general-purpose digital processor without impacting overall imaging performance and scalability.

This paper is organized as follows. Section II discusses the architecture of pixel-neighborhood-level processing. Hardware description, design, and instruction set are presented in Section III. CMOS implementation and layout are presented in Section IV. Sections V and VI showcase the fabricated hardware and test platform with results. Finally, Sections VII and VIII discuss these results and conclude this paper.

II. Vision Chip Architecture

A. Architecture Overview

Fig. 2 shows an overview of the presented vision chip architecture. It consists of a 2-D NP array controlled by a single global control unit (GCU), which executes instructions from a program memory, and an analog-to-digital converter finite-state machine (ADC FSM) that drives the DPS ADCs. Fig. 3 shows the contents of a single NP. It contains an 8 × 8 DPS array that captures raw image data directly into the NP's memory. Digital logic is embedded physically between the DPS structures on the same focal plane. The logic implements a programmable, 8-b processor with local memory, registers, and ALU, and is capable of performing low-, mid-, and high-level algorithms on image data. NPs are able to directly

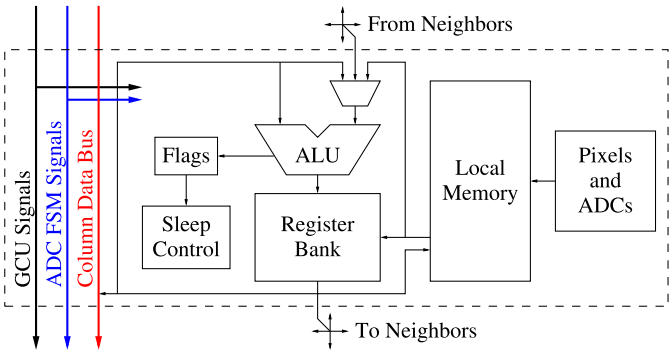

Fig. 3. Block diagram showing the contents of an NP.

exchange data between adjacent NPs and also have access to a column-level data bus that can be used to output data, in a single clock cycle, to an external resource.

The statistical properties of the input data are an important consideration when selecting the neighborhood size. For imaging natural scenes, pixel correlation drops off outside of an 8 × 8 window, and as a result, it is often selected as the block size in image compression algorithms [36]. The vision chip in this paper targets imaging applications and uses a neighborhood size of 8 × 8, and the NPs are interfaced to DPSs to form a visible-spectrum imager.

NPs may be programmed to operate in groups to support larger imaging algorithm block sizes. For example, a 2 × 2 group of NPs can operate on 16 × 16 pixel blocks. To accomplish this, adjacent NPs exchange data between each other, with registers being immediately available and local memory requiring and additional clock cycle to access.

B. Architectural Design Requirements

The neighborhood-level processing architecture was implemented to satisfy multiple design requirements related to scalability, programmability, and physical layout. Comparisons between neighborhood-parallelism and other existing PE architectures, including pixel processors (PP), row/column processors (RCP), and chip processors (CP), are drawn as well in this section.

1) Resolution Scaling: In designing a vision chip, it is desirable to be able to easily scale the resolution of the imager. Similar to PPs, NPs are designed to be tileable and can be abutted to form arbitrary resolutions. For algorithms where every pixel must be processed, increasing resolution requires each RCP or CP to perform more computation, slowing its maximum throughput. NPs and PPs process a fixed amount of pixel data regardless of resolution and do not lose throughput at higher resolutions. The amount of pixel data each type of processor must compute, per frame, across scaling resolution, is shown in Table I.

In order for the NP architecture to be scalable to higher resolutions, adding additional rows and columns of NPs should not significantly slow down the operating frequency of the array. Global control and ADC signals are driven from the GCU to the first NP in every column using large buffers that introduce minimal delay. The signals are then propagated

TABLE I
PIXELS PER PROCESSOR UNDER SCALING IMAGER RESOLUTION FOR A PIXEL PROCESSOR (PP), NP, ROW/COLUMN PROCESSOR (RCP), AND CHIP PROCESSOR (CP)

Resolution	PP	NP	RCP	CP
8×8	1	64	8	64
16×16	1	64	16	256
32×32	1	64	32	1024
64×64	1	64	64	4096
128×128	1	64	128	16 384
256×256	1	64	256	65 536

TABLE II
CRITICAL PATH TIMING FOR MULTIPLE NP ARRAY SIZES

Imager resolution	64×80	128×128
Off-Chip Memory Access	22.3 ns	22.3 ns
Testability Multiplexers	1.1 ns	1.1 ns
GCU Instruction Decode	0.6 ns	0.8 ns
Control Signal Propagation	0.9 ns	2.2 ns
NP Logic and Memory Access	2.5 ns	2.4 ns
Column Data Bus Propagation	1.7 ns	3.3 ns
Total	29.2 ns	32.2 ns

down the columns and buffered by each NP to the next. This leads to additional columns of NPs adding negligible delay compared with additional rows of NPs. Table II shows that increasing the resolution from 64×80 to 128×128 introduces 3 ns of additional delay, or approximately 360 ps per additional row. A strong external buffer drives the ADC ramp, allowing it to propagate faster than the digital signals for all array sizes. The GCU is designed to handle up to a 16×16 array of NPs, corresponding to a resolution of 128×128. The GCU's maximum resolution can be increased by updating an RTL constant, adding only the additional row propagation delays.

The operating frequency of the vision chip is 20 MHz, limited by critical timing paths where an NP's local memory is read and output through the column data bus. The primary source of delay is related to the off-chip program memory, chosen to simplify chip testing, which accounts for 22.3 ns of the 50-ns clock period. For comparison, an on-chip SRAM would reduce this to a 1.25-ns delay. In addition, test structures that allow critical signals to be bypassed externally add 1.1 ns of delay. Significant timing slack is left to provide necessary flexibility for the digital place and route tools to generate a valid layout in a reasonable number of iterations.

2) Algorithm Support: The historical popularity of 8-b processors has led to the development of many optimized low-, mid-, and high-level 8-b algorithms that may be easily ported to run on the 8-b NP architecture. Alternative processor architectures use serial, variable-bit-width designs that increase flexibility by allowing the algorithm to tradeoff precision for speed and storage [4], [19], [21], [32], [33], [35]. However, they require a higher clock rate to achieve the same performance as an 8-b architecture. Hybrid architectures combine both parallel and serial computation, which requires more sophisticated software tools to efficiently map algorithms to both architecture styles [33].

Low-level algorithms, such as edge detection and simple filters, are executed for each pixel in full parallel. Assuming a square resolution of $N \times N$, these algorithms can be executed in $O(1)$ time by PPs, $O(N)$ time by RCPs, and $O(N^2)$ time by nonparallel CPs. Since NPs are responsible for a fixed number of pixels independent of resolution, they also execute low-level algorithms in constant time, albeit at a fixed fraction of the speed of a PP, since it needs to process more pixel data each frame.

Mid-level algorithms require data from multiple pixels to compute, such as gathering image statistics and block transforms, such as the DFT and DCT. PPs have difficulty with this type of algorithm, and RCPs are better suited, since they contain more memory and have fast access to nearby and distant pixel data located in its row/column. NPs are also well suited for this class of algorithms, since they have significant amounts of memory and immediate access to all pixels in their neighborhood. The advantage of NPs over RCPs is that NPs leverage the natural correlation seen in nearby pixels used in imaging applications, where RCP's image data span a long slice of the focal plane, which tends to be less correlated.

High-level algorithms involve processing that requires random access to data across the entire image and is inherently not parallelizable. NPs require several clock cycles to move data across the NP array and operate slower with this class of algorithms, but in a similar fashion as PPs and RCPs. Chip-level processors are best suited for this type of computation and can be paired, on-chip, with NPs to complement their feature set.

3) Interpixel Processor Layout: The digital logic for the processor is implemented physically between the photodiodes on the focal plane, which yields several benefits.

1) Increased operating frequency, since the DPS ADCs and memory are located close together, minimizing wiring delays experienced by RCPs that utilize row/column drivers.
2) Images are stored directly to memory, removing delays associated with loading data externally as seen in architectures with separated imagers and processors [32], [33], [35].
3) DPSs facilitate the entire image to be captured simultaneously across the array, which requires less time than architectures that convert only one row/column at a time.

4) Area and Fill Factor: The ALU, registers, and memory within an NP consume additional area beyond the photodiodes and ADCs. Integrating these structures between pixels requires that pixel pitch be increased or photodiode areas decreased, both leading to reduced fill factor. RCPs and CPs are implemented outside of the focal plane, so the fill factors in these architectures are not similarly constrained. In this paper, a 12% fill factor has been targeted. This effectively limits the maximum memory within an NP, since memory occupies the majority of the area.

C. Performance of Modeled Processing Architectures

In order to draw a general comparison between the presented neighborhood-parallel architecture and existing pixel-parallel and row/column-parallel architectures, each was modeled to estimate its relative performance for different algorithms. Execution speed is strongly related to how easily

TABLE III
ASSUMED PE CLOCK CYCLES FOR 8-b OPERATIONS

Operation	NP	RCP	PP
Load image data to PE	0	1	0
Move data between PEs	1	1	8
ALU Operation	2	1	8
Store ALU result to memory	1	1	0

TABLE IV
ESTIMATED PERFORMANCE OF PE ARCHITECTURES FOR SEVERAL ALGORITHMS

	Clock Cycles			Time (μs)		
	NP	RCP	PP	NP	RCP	PP
Edge Detection	1200	1900	100	60	19	1
Histogram	9300	35 000	32 000	465	350	320
8×8 DCT	3300	18 000	5000	165	180	50

a PE can move data within itself and between adjacent PEs. This varies significantly among implementations and greatly affects the results, so several simplifying assumptions were used, summarized in Table III.

1) NPs and RCPs use single-cycle, 8-b ALUs that save the result to a register, and PPs use serial ALUs that load, compute, and save to the same memory in one operation.
2) Since RCPs are implemented outside of the focal plane, it is assumed they require an additional clock cycle to make this data available to the ALU, compared with NPs and PPs.
3) All PEs can communicate only with the PE adjacent to them, so data must be shifted across the array.
4) PPs are able to immediately use data from adjacent PEs, where NPs and RCPs must first load it into their own memory.
5) PPs have sufficient local memory to hold necessary intermediate values, without which the histogram and DCT algorithms would not be possible to implement.
6) Program memory is infinite, meaning loops may be unrolled, and pixel addresses may be hard-coded in the program.

Based on the number of ALU operations and memory accesses combined with the assumptions in Table III, the performance of each PE architecture was modeled. Table IV shows the resulting estimated number of clock cycles needed to perform each algorithm on a 128×128 imager. The expected execution time is also modeled, assuming the NPs operate at 20 MHz and the RCPs and PPs at 100 MHz. Despite the lower clock frequency, NPs and RCPs show similar execution times. PPs are consistently the fastest of the three, but many PP implementations have too little local memory to realize these algorithms in practice.

III. HARDWARE DESCRIPTION OF VISION CHIP ARCHITECTURE

The presented vision chip is comprised of an array of NPs that are controlled by a single GCU and ADC FSM.

A. Neighborhood Processor

The NP is the core PE of the presented neighborhood-parallel vision chip architecture. Fig. 4 shows a detailed block diagram of the contents of a single NP. It implements a programmable, 8-b RISC processor complete with an ALU, register bank, and local memory. The architecture is designed to allow creation of arbitrary resolutions by abutting NPs against one another. Each NP has four bidirectional data buses connecting it to adjacent NPs, allowing 8-b data to be shifted one NP over in the array every clock cycle. All NPs must transfer data in the same direction, which is determined by the data source used in the program instruction. In addition, each column of NPs share an output data bus that allows an external device, such as a host processor, to read 8-b data from any NP in the array in one clock cycle. The features of the NP are detailed next.

1) Local Memory: Each NP contains 200 B of local memory. It is divided into three 64-B pages, designated by X, Y, and Z, and the fourth page of 8 B, designated by V. X memory consists of 64 B that are directly coupled to the DPS array. During ADC conversions, the DPSs assume control of this memory and are able to write image data directly into it in parallel. When not imaging and during the DPS integration time, X memory is available to the NP and may be used in algorithms. Y and Z memory are each 64 B and allow for up to two additional frames of image data to be stored in the NP. V memory contains 8 B and provides variable storage for algorithms that require all of X, Y, and Z memories to store image data.

2) Registers: The NP contains four 8-b registers. While local memory may serve as one input to the ALU, the second input must be a register. In addition, ALU results may only be stored in registers and cannot be saved directly to local memory. Load and store instructions allow data to be moved between registers and local memory.

3) ALU: The ALU performs 8-b addition, subtraction, logical, and shift operations in a single cycle. It generates carry, sign, zero, and overflow flags used for conditional instruction execution and branching. It also contains a single-cycle, hardware gray code to binary converter, which is necessary to convert the DPS ADC output into a binary value. It is able to source input from any of the registers, local memory, adjacent NPs, status register (SR), or row-column register, which contains a unique identifier assigned to each NP in the array.

4) Sleep Controller: The sleep controller gates the NP clock, preventing the NP from executing the current program instruction, allowing for code to be run conditionally by the NP. This simplifies conditional if–else code blocks where only certain NPs should execute a portion of program code. The NP remains disabled until a wake instruction is encountered during the program execution. Sleeping NPs use less power, reducing power consumption up to 5% if all NPs are asleep.

5) Digital Pixel Sensors: The NP contains 64 DPSs arranged in an 8 × 8 grid. Each DPS contains a photodiode, analog buffer, and comparator that, together with a global counter, implement a single-slope ADC. A single,

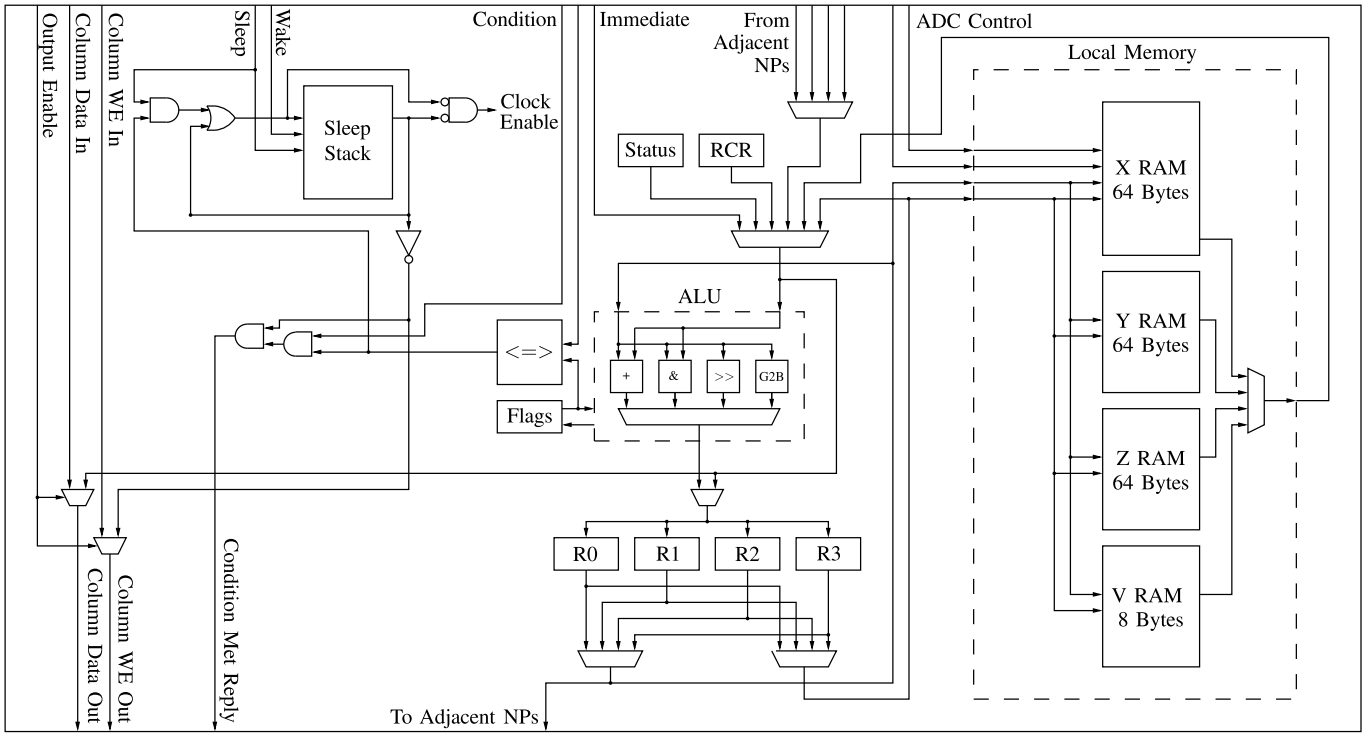

Fig. 4. Detailed NP diagram showing all aspects of the architecture.

global ADC FSM controller drives all DPSs during conversions and controls reset and integration timing. Digital correlated double sampling (CDS) is performed by first converting and storing the reset voltage of each DPS, which is later subtracted from the actual pixel values by the NP's ALU.

B. Global Control Unit

The GCU reads program instructions from the program memory and decodes them into a number of global control signals, which are distributed to all the NPs in the array.

Since all NPs execute the same instructions in parallel, conditional branching is problematic if not all NPs satisfy the required condition. This is handled by placing any NPs that do not satisfy the branch condition to sleep, preserving their state, while those that do execute the required instructions. Once the conditional instructions have been executed, WAK (wake) is executed to wake any disabled NPs.

The GCU is designed to consume little area, since it should not significantly increase the chip dimensions beyond what the NP array requires. The full RISC instruction set is shown in Table V. The available data sources are shown in Table VI, which include direct and indirect local memory access, constant values, the SR, and row–column address. The GCU is implemented along the outside edge of the NP array, and therefore does not share the same layout restrictions as the NP.

TABLE V
INSTRUCTION SET

Category	Instruction							
Arithmetic	ADD	ADC	SUB	SBC	CMP			
Logic	AND	ORR	EOR					
Shift	ASR	LSL	LSR					
Memory	LDR	STR						
Subroutine	BL	BX						
Branch	BEQ	BNE	BHS	BLO	BMI	BPL	BVS	BVC
	BHI	BLS	BGE	BLT	BGT	BLE	BAL	
Sleep	ZEQ	ZNE	ZHS	ZLO	ZMI	ZPL	ZVS	ZVC
	ZHI	ZLS	ZGE	ZLT	ZGT	ZLE	ZAL	

TABLE VI
DATA SOURCES

Category	Description
Memory	X Y Z V
Registers	R_0 R_1 R_2 R_3
Adjacent NPs	N S W E
Indirect Memory	X[R_i] Y[R_i] Z[R_i] V[R_i]
Special	SR RCR IMM

C. ADC FSM

The ADC FSM generates global control signals needed for the DPSs to perform conversions. It has programmable reset and integration timing to allow for dynamic contrast adjustment and multiple-exposure imaging in varied lighting conditions. It is physically integrated with the GCU and triggered through dedicated ADC instructions.

IV. CMOS HARDWARE IMPLEMENTATION

The vision chip presented in this paper is designed around an 8×8 pixel neighborhood and contains an 8×10 array

TABLE VII
DIGITAL INSTANCE COUNTS FOR A SINGLE NP, GCU, AND ADC FSM

Type	NP		GCU and ADC FSM	
	Instances	Area (μm^2)	Instances	Area (μm^2)
Sequential	1639	40 100	158	4304
Inverter	59	200	76	466
Clock Gating	206	5900	16	462
Logic	1843	15 400	333	3050
Total	3747	61 600	583	8282

of NPs forming an imager resolution of 64×80 pixels. It is implemented using a 1P8M 0.13-μm CMOS process and built from a combination of digital standard cells and full-custom DPSs. The GCU and ADC FSM are implemented on-chip along the side of the NP array, while the program ROM and clock generator exist off-chip. Analog control and bias voltages, including the ramp voltage used in the DPS ADCs, are driven externally for convenient testing and to allow exploration of nonlinear ramp waveforms.

A. Gate-Level Hardware Design

A core design challenge in implementing NPs is the task of integrating the digital logic within the array of DPSs. The logic must be placed and routed to leave large gaps in the layout for DPS structures. In this paper, the digital portions of the NPs are implemented using standard cells. This allows for faster implementation on silicon, since they may be flowed around the DPS structures in a nonrectangular manner. Standard cells also allow use of robust software tools to close timing by sizing buffer drives strengths, preventing setup, and hold violations in the NP array. The final standard cell area and counts for the NP and GCU are summarized in Table VII.

B. Digital Pixel Sensor

The DPS schematic is shown in Fig. 5. Level shift circuitry allows for a direct connection of the DPS cell to the digital logic cells, converting 2.5-V analog to 1.2-V digital and vice versa. Control lines, such as bias enable and photodiode reset, are level shifted to the analog supply voltages.

The photodiode is reset when the photodiode reset line is held high and is pulled up by a p-type transistor, to allow a full supply rail reset voltage. The reset voltage is provided off-chip by a DAC and routed over a global analog net to all photodiodes in the array, allowing an arbitrary reset voltage to be used for dynamic range and contrast control. The photodiode voltage is buffered by a source follower before reaching the comparator to prevent leakage while the ramp is increasing and the comparator bias is enabled.

The comparator is biased in subthreshold for maximum gain for minimum transistor W/L ratio and can be controlled by the global bias voltage. When the comparator detects a photodiode voltage that has crossed the global ramp compare line, it latches the value with a common source amplifier followed by a final level shift to the digital supply range.

The DPS design uses one layer of metal for all internal routing and two layers for digital and analog shields. This was crucial in allowing the top-level NP design to be realized. By minimizing the number of metals used in the DPS structures, routing area available for NP logic and memory was greatly increased. In addition, minimum area transistors are used to make room for larger photodiodes, increasing the imager fill factor, while CDS mitigates the increase in pixel-to-pixel sensitivity variations caused by these transistors.

C. Integration on Silicon

In order to exploit the repetitive configuration of the NPs in the design, hierarchical implementation was used. NPs underwent place and route first, then were imported into a chip-level design hierarchy where they were placed in an 8×10 array.

Successful NP layout required two challenges to be overcome. First, gaps had to be left for the DPSs in the digital logic. Second, no routing could be present over the photodiode, which would block light from reaching the DPS. This led to minimal routing area available to connect standard cells, creating significant routing congestion. As much as area constraints, routing congestion introduced significant challenges during the layout process. This is due to having only three metal layers suitable for routing in the implementation technology and limited space available between pixels to place wires without blocking photodiodes. In order to maximize open area for digital cell placement, the ADC within each DPS is oriented in an alternating pattern. This leaves larger open regions for standard cell placement and routing, as shown in Fig. 6.

After the individual NP was implemented, the full chip was placed and routed. Fig. 7 shows a 7.2-μm-wide channel between adjacent NPs where top-level clock tree buffers are inserted. All digital routing is completed on the bottom three metals and metal 4 is used for digital power routing. Metal 5 is used as shielding between the analog and digital sections, and metals 6–8 are used for analog power and global signal routing to all of the DPSs across the chip. The GCU and ADC FSM were routed in a small region adjacent to the NP array and consume much less area than a single NP.

The DPS cells were iteratively designed with two optimization goals: 1) maximizing fill factor of the photodiode while minimizing overall cell area and 2) maintaining routability of the surrounding digital standard cells.

The first requirement forces the overall shape of the cell to fit into a multiple of the standard cell height, eliminating collisions and voids when abutting the custom shaped cell with the contiguous rows of standard cells. Minimal widths and lengths were chosen to reduce layout area, with CDS compensating for the resulting increase in ADC comparator offset variation. Leakage of the photodiode voltage was reduced by using thick-oxide, high-V_t transistors throughout the analog stages. Imaging quality and latching speed are dependent on the gain and current draw of the stages, and with the required minimum transistor $W \cdot L$ products, subthreshold biasing was required for the source follower, comparator, and common source amplifier. The biasing voltages were consolidated to

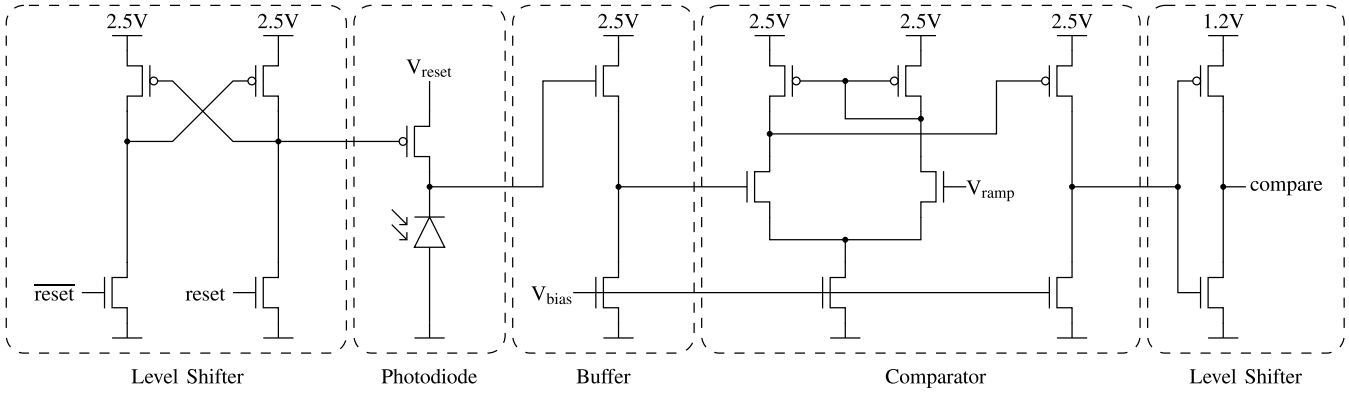

Fig. 5. Schematic of the DPS cell consisting of level shifters, photodiode with reset circuitry, buffer, comparator, and amplifier.

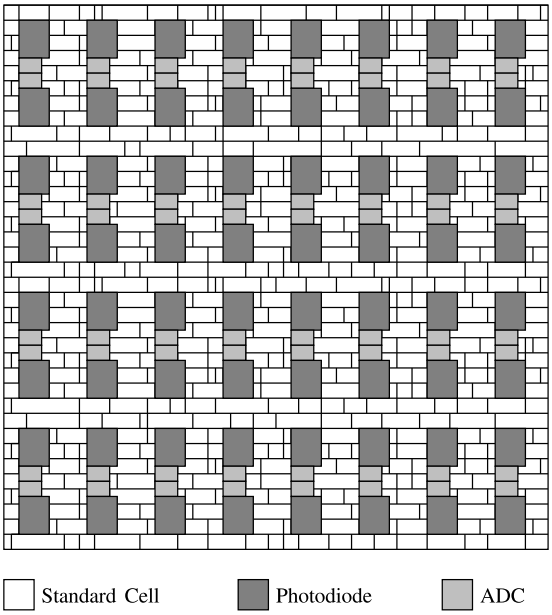

Fig. 6. Floorplan of a single NP consisting of digital standard cells and an 8×8 array of photodiodes and ADCs.

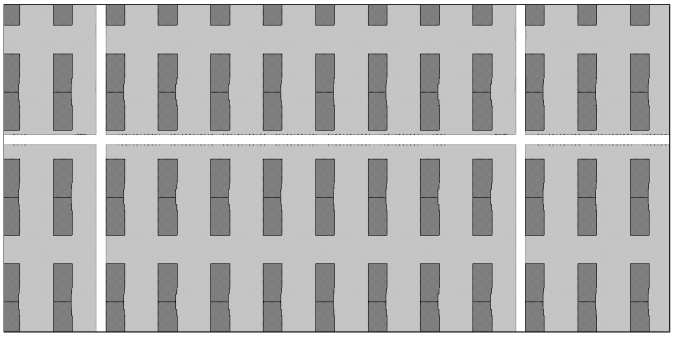

Fig. 7. Routing channel is left between adjacent NPs for clock tree buffer insertion in order to satisfy timing constraints.

use the same value, greatly reducing routing complexity for this critical analog route.

The second requirement leads to only the lowest metal layer being used for routing within the DPS cell. In particular,

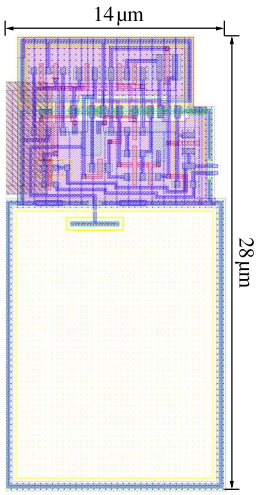

Fig. 8. DPS cell consisting of a photodiode paired with shielded analog circuitry. Analog routes inside the cell use the lowest layer of metal with digital and analog pins brought up to their respective top layer metals.

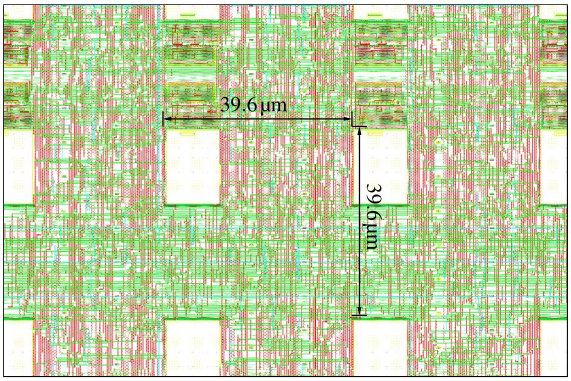

Fig. 9. DPSs after placement within digital cells. Routing channels traverse the analog shields without obstruction allowing a compact and high density layout with 39.6-μm pixel pitch.

pins were brought up to a single routing row on metal 3 for digital signals and metals 6–8 for analog signals. Metals 2 and 4 were used for shielding the DPS from digital and analog signals, respectively. The final DPS design is shown in Fig. 8. Fig. 9 shows the placed DPS cells with surrounding digital routing.

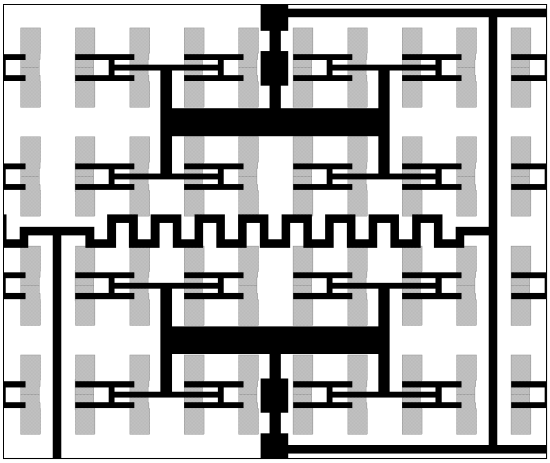

Fig. 10. H-tree wiring topology used to distribute DPS ramp voltage across the NP array with minimal skew between endpoints.

TABLE VIII
CHIP SPECIFICATIONS

Technology	0.13 µm 1P8M CMOS
NP Dimensions	317 µm×317 µm
Pixel Pitch	39.6 µm
Photodiode Dimensions	11.34 µm×15.44 µm
Fill Factor	12%
Fixed Pattern Noise	3.5%
Fixed Pattern Noise (CDS)	1.1%
Resolution	64×80
ADC	8-bit, single-slope
Total Transistors	$\approx 2.3 \times 10^6$
Die Size	5 mm×3.1 mm
Vision Core Size	3.1 mm×2.5 mm
Supply Voltage	1.2 V, 2.5 V
Clock Frequency	20 MHz

The routing of the ramp voltage for the ADC ensured that all ADCs received the same ramp voltage and that there was no systematic ramp timing skew introduced into the array due to distance from the ramp generator. To ensure equal propagation delay to all ADCs, the ramp is routed as an H-tree network, as shown in Fig. 10, so that the distance from signal source to termination is the same for every DPS. The network has a propagation delay of 14 ps and a timing skew of under 1 ps at the endpoints.

A die photo of the fabricated vision chip is shown in Fig. 11 with the NP array highlighted along with the GCU and ADC FSM layout near the top-right of the array. A single NP is highlighted from the array and zoomed to a layout view showing more detail of the metal routing. Some of the power and shielding layers were omitted to reveal more detail of the layout. Table VIII summarizes the vision chip specifications.

V. HARDWARE TEST BED

A. Support Circuitry

A picture of the hardware test bed is shown in Fig. 12. This board was interfaced with a commercial development board containing an FPGA with USB 2.0 connectivity. A lens mount was 3-D printed to align the PGA chip package with a CS-mount CCTV lens with F1.0 to maximize captured light.

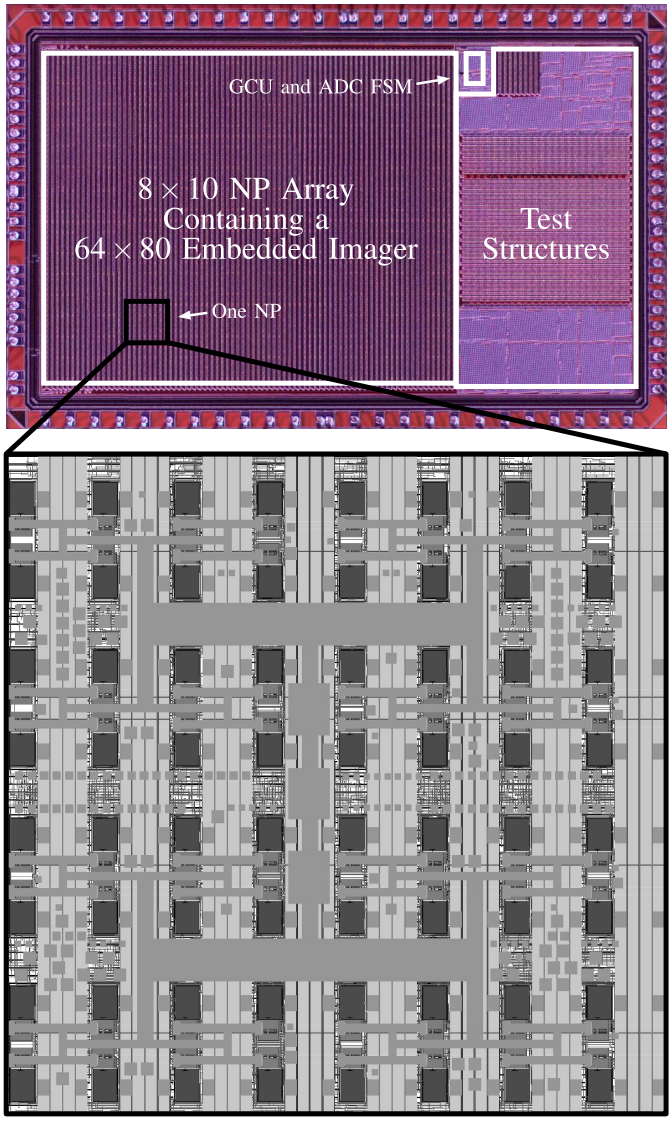

Fig. 11. Die photo of presented vision chip with an 8 × 10 array of NPs forming a resolution of 64 × 80. The layout for a single NP is enlarged that shows the 8 × 8 array of photodiodes, and analog and digital routing.

Initial testing used an asynchronous FIFO USB transfer mode and achieved streaming rates of up to 550 frames/s. Later revisions of the codebase and an updated USB interface allowed use of a synchronous FIFO that reached frame rates in an excess of 1500 frames/s.

B. Program, Control, and Recording Software

A software application suite was written to configure imaging parameters, capture images, and to perform video streaming and recording. The application can upload program code to the chip from a list of compiled assembly programs.

State machines were written on the FPGA to read out from the vision chip's data bus and buffer one frame of data on the FPGA's RAM to maximize data throughput. The application can modify imaging parameters during runtime, including integration time, reset voltages and time, and USB parameters, such as latency and packet size. A picture of the GUI in streaming mode is shown in Fig. 13.

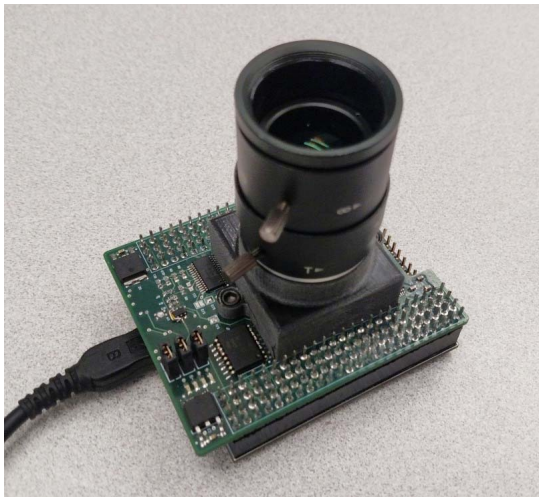

Fig. 12. Hardware test platform, which consists of the vision chip, analog board, and FPGA development board connected to PC via USB cable.

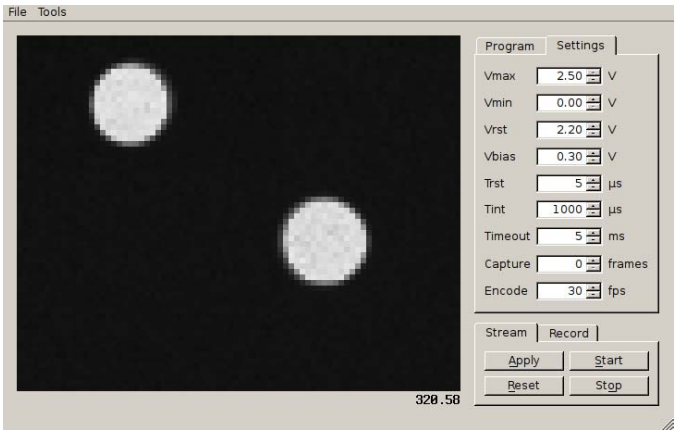

Fig. 13. Application GUI showing configuration settings and live video stream from the vision chip.

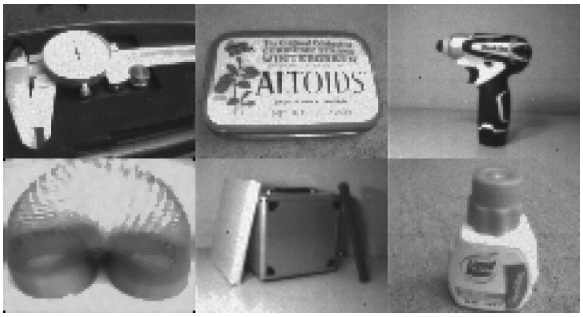

Fig. 14. Imaging using CDS.

VI. Test Results

A variety of processing tasks have been implemented to demonstrate the reprogrammable operation of the vision chip. Static imaging, edge detection, image filtering, statistics gathering, and object tracking results are presented in this section.

A. Imaging Quality

The vision chip's imaging quality is shown through several examples in Fig. 14, where CDS is used during capture for various objects. Object details can be seen clearly, highlighting the performance of the imager in capturing scenes with indoor lighting conditions and yielding a fixed-pattern noise of 1.1%. Fig. 15 shows artificial lighting scenes for high-speed recording using the asynchronous FIFO data mode.

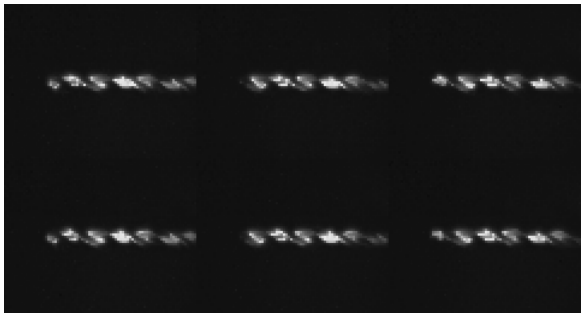

Fig. 15. Recording: 1300 RPM drill. 330 frames/s, every fifth frame shown.

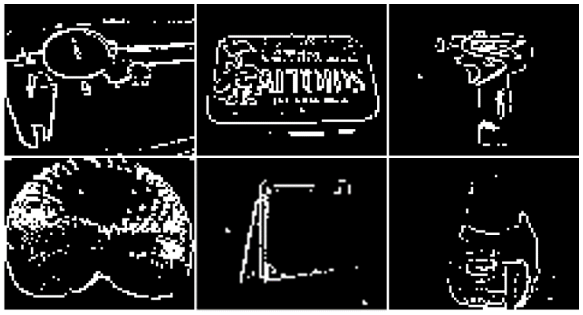

Fig. 16. Edge detection processed images.

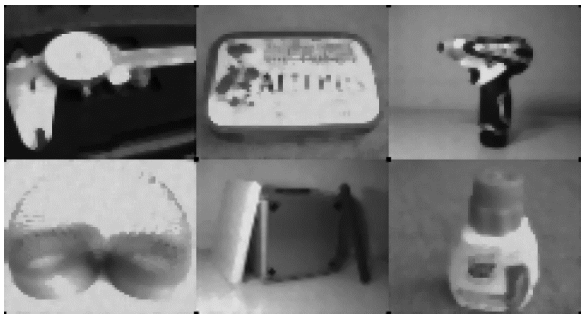

Fig. 17. 3×3 median filtered images.

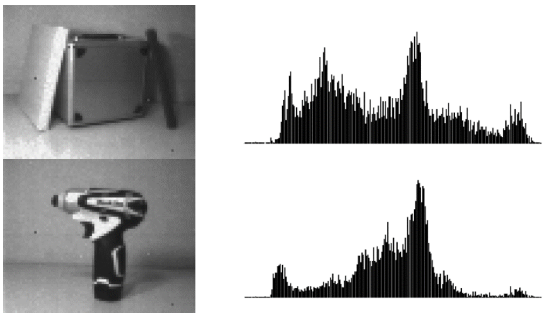

Fig. 18. Histogram calculation on acquired images.

B. Edge Detection

Fig. 16 shows the vision chip performing a threshold-based edge detection algorithm on the scenes in Fig. 14. The software

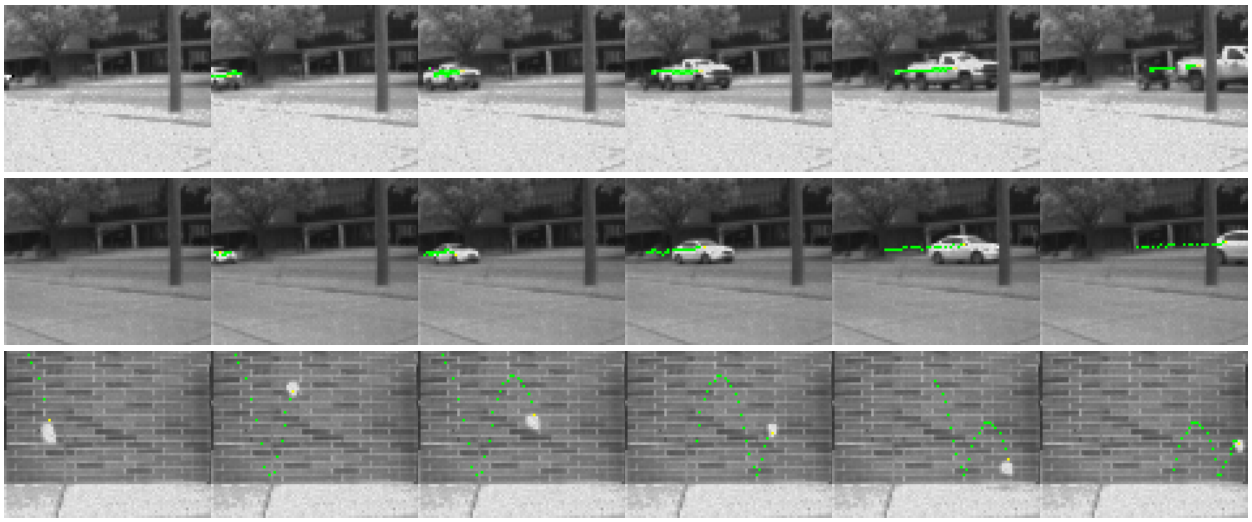

Fig. 19. Tracking. Top row: moving truck with trailer. 45 frames/s, every 15th frame shown. Middle row: moving car. 45 frames/s, every 15th frame shown. Bottom row: bouncing ball. 30 frames/s, every 10th frame shown.

application allows the edge detection threshold value to be changed to fit the scene's needs.

C. Median Filtering

In the median filter algorithm, the vision chip applies a 3×3 median filter to the original scenes. The resulting images are shown in Fig. 17.

D. Image Histogram

Fig. 18 shows the original image and its histogram calculated by the vision chip. In this algorithm, 16-b arithmetic is used in order to handle the potentially large histogram bin counts. The memory requirements for 256 16-b bins exceed that of a single NP, so 2×2 clusters of NPs work together to store the histogram bins. Once the histogram for each cluster is calculated, the histogram data from each are shifted across the entire array and summed within a single cluster, resulting in the final histogram.

E. Tracking

For the tracking algorithm, the vision chip calculates the differences between the current frame and last frame to determine the bounding box of movement. The center of this bounding box is then calculated and embedded in the image data sent back, allowing the user to save, analyze, and overlay this information. In the following recordings, a yellow dot is placed at the current center of the bounding box, while green dots outline the previous path traversed by the center coordinate data.

1) Tracking (Outdoor Scenes, Natural Lighting): The test setup was taken outdoors to capture real-life scenes, demonstrating tracking of irregular objects in ordinary conditions during an overcast day. In addition to lower ambient light levels, moving stimuli and/or subtle contrast differences occur in these outdoor scenes, requiring a longer integration time to extract an accurate path. Fig. 19 shows outdoor tracking scenes. The first and second rows show a truck with trailer and a car being tracked at 45 frames/s, and the third row shows a bouncing ball being tracked at 30 frames/s.

TABLE IX
SAMPLE NP PROGRAMS

Program	Program Length	Execution Time (μs)	Frame Rate (fps)	Power (mW)
CDS Imaging	58	103	906	25
8×8 DCT	347	380	724	36
Edge Detection	128	248	801	31
3×3 Median Filter	361	734	576	38
Histogram	257	832	545	31
Tracking	269	237	808	29

2) Tracking (Indoor Scenes, Artificial Lighting): For these results, a 150-W halogen light source was used for illumination with the vision chip. Fig. 20 shows indoor tracking scenes utilizing the synchronous FIFO USB transfer mode, demonstrating the upper limit of performance of the test setup. The first row shows tracking of a light source behind rotating fan blades and is captured at 921 frames/s. The second and third rows show the halogen light moving across the field of view and are captured at 1128 and 1015 frames/s, respectively.

VII. DISCUSSION OF RESULTS

A summary of the performance of the NPs executing several algorithms is presented in Table IX. The execution time is related to the complexity of the algorithm as well as the amount of data movement operations required. Algorithms that require few calculations, such as edge detection, execute in less time than more math-intensive ones, such as the DCT and median filter. Calculating the whole-image histogram is computationally simple, but involves shifting pixel data across the full imaging array, leading to increased run time. Frame rate was calculated using an integration time of 1 ms, which was realistically achieved in an outdoor sunny scene or indoors using a halogen light source. Power was measured with each

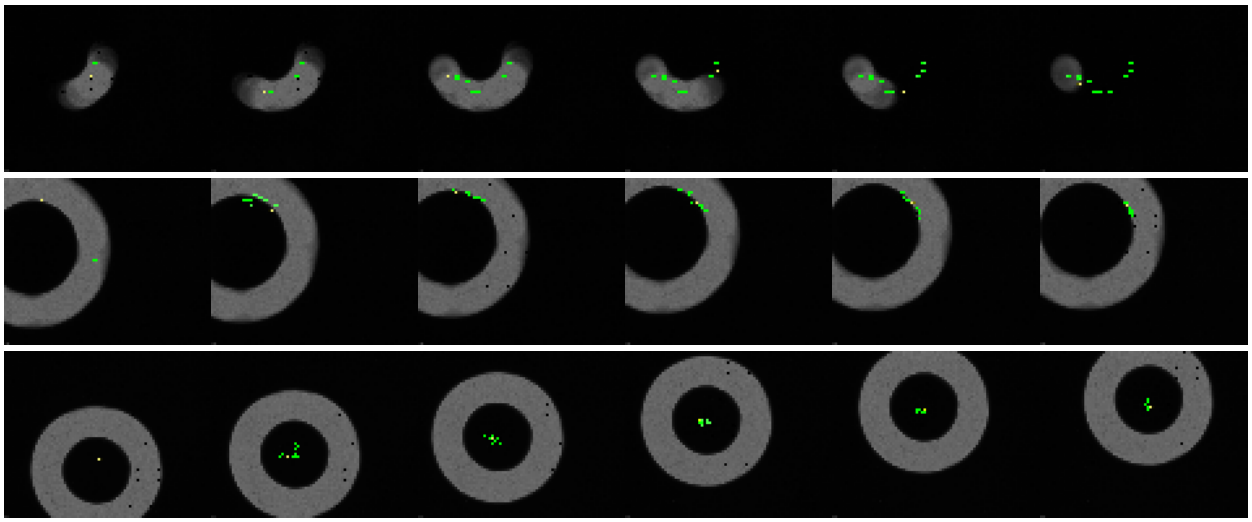

Fig. 20. Tracking. Top row: light behind a fan. 921 frames/s, sequential frames shown. Middle row: leading edge of a light ring. 1128 frames/s, every 20th frame shown. Bottom row: center of a light ring. 1015 frames/s, every 50th frame shown.

TABLE X
COMPARISON OF RECENT VISION CHIP IMPLEMENTATIONS

	This Work 2016	Shi 2014 [32]	Cottini 2013 [37]	Lopich 2013 [19]	Oliveira 2013 [23]	Lopich 2011 [21]	Zhang 2011 [33]	Ginhac 2010 [15]	Komuro 2009 [4]	Lin 2009 [35]
Parallelism	NP	Hybrid	PP	PP	Hybrid	Hybrid	Hybrid	PP	RCP	Hybrid
Technology	0.13 µm	0.18 µm	0.35 µm	0.18 µm	0.35 µm	0.35 µm	0.18 µm	0.35 µm	0.35 µm	0.18 µm
Resolution	80×64	256×256	64×64	160×80	32×32	19×22	128×128	64×64	320×240	64×64
Die area (mm^2)	16.0	82.3	-	50.0	2.0	9.0	13.5	13.8	78.5	5.2
PE dimensions (µm×µm)	317×317	-	26×26	54×51	37.5×37.5	100×117	65×25†	35×35	20×20	23×29†
PE array	10×8	64×64	64×64	160×80	8×8	19×22	32×128	64×64	320×240	32×32
Clock speed (MHz)	20	50	-	100	1	75	100	-	50	40
Frame rate (FPS)	1000	1340	13	-	125	-	1000	10 000	1000	1000
Fill factor (%)	12	60†	12	5.6	7	2	58†	25	-	58†
Sensor Type	DPS	APS	APS	DPS	APS	DPS	APS	APS	APS	APS
Grayscale (bits)	8	10	NA	8	NA	8	8	NA	-	6
Memory (bits/pixel)	24	64	NA	72	NA	64	72	NA	-	8
ALU bit width	8	1, 8	NA	1	NA	1	1, 8	NA	1	1, 11
Programming flexibility	H	H	L	H	L	H	H	L	M	H
Algorithm support	HML	HML	L	ML	M	ML	HML	L	HML	HML
Power (mW)	36	630	0.033	-	37	26.4	450	-	41.6	82.5
GOPS	1.6	12	-	23	-	1	44	-	-	-
GOPS/mm^2	0.2	-	4	0.65	-	0.2	3.4	-	-	-
GOPS/W	44.4	19	42	46	-	38	97.8	-	-	-

† The imager is implemented outside of the processing element array.

algorithm running in a loop at 20 MHz with no imaging being performed.

It is not straightforward to draw one-to-one comparisons between vision chips, since standard metrics, including power and GOPS, can vary depending on what algorithm the vision chip implements and the frame rate at which it is performed. In addition, implemented designs use CMOS technologies that are typically in the 0.13–0.35-μm range, which exhibit unique advantages and disadvantages in terms of photodiode performance and digital speed and density. Table X compares recent examples of vision chips against the architecture presented in this paper. The dimensions of a single neighborhood-level processor appear much larger than the PE in other works, since it contains more pixels than a pixel-level processor. Dividing this area by the 8×8 neighborhood size gives dimensions that are comparable to other programmable architectures that support low-level (L), medium-level (M), and high-level (H) algorithms. The NPs operate at 20 MHz, which, while relatively slow, performs 8-b operations in a single cycle that would require multiple cycles for designs with serial ALUs to complete [4], [19], [21], [32], [33], [35]. However, serial ALUs require less area and, when used in PP-based architectures, have higher GOPS than the NP architecture, since they have more ALUs for the same resolution [19], [21]. The presented vision chip has a lower fill factor than most single-purpose designs as well as those which separate the photodiodes and ADCs from the digital PEs [32], [33], [35]. However, it compares well with architectures that integrate

processing in the focal plane, with similar levels of programmability and scalability through tiling [19], [21].

VIII. Conclusion

This paper presents a novel, neighborhood-parallel vision chip architecture that possesses characteristics of existing pixel-parallel and row/column-parallel architectures. The PE consists of an 8×8 array of DPSs and digital logic, that implement an 8-b processor, physically integrated between the sensors on the same focal plane. A key characteristic of the described architecture is scalability. Each NP propagates global signals to adjacent NPs, which allows NPs to be tiled into larger arrays to directly increase the imager resolution. Implementing the processor within the pixel neighborhood requires a wider pixel pitch than row/column-level and chip-level processors where the imager and processors are physically distinct. However, unlike these architectures, increasing the resolution does not increase the number of pixels each NP must handle, so processing algorithms run at the same speed regardless of resolution.

The single-chip vision system is programmable using a RISC instruction set, and is able to perform low-, mid-, and high-level image processing tasks. Several example algorithms were demonstrated covering all levels of image processing tasks, including real-time target tracking at greater than 1000 frames/s.

References

[1] B. Zhang, K. Mei, and N. Zheng, "Reconfigurable processor for binary image processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 5, pp. 823–831, May 2013.

[2] A. Abbo *et al.*, "XETAL-II: A 107 GOPS, 600mW massively-parallel processor for video scene analysis," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2007, pp. 270–271 and 602.

[3] M. Sakakibara *et al.*, "A high-sensitivity CMOS image sensor with gain-adaptive column amplifiers," *IEEE J. Solid-State Circuits*, vol. 40, no. 5, pp. 1147–1156, May 2005.

[4] T. Komuro, A. Iwashita, and M. Ishikawa, "A QVGA-size pixel-parallel image processor for 1,000-fps vision," *IEEE Micro*, vol. 29, no. 6, pp. 58–67, Nov. 2009.

[5] Q. Lin, W. Miao, and N. Wu, "A high-speed target tracking CMOS image sensor," in *Proc. IEEE Asian Solid-State Circuits Conf.*, Nov. 2006, pp. 139–142.

[6] A. Elouardi, S. Bouaziz, A. Dupret, L. Lacassagne, J. O. Klein, and R. Reynaud, "A smart sensor for image processing: Towards a system on chip," in *Proc. IEEE Int. Symp. Ind. Electron.*, vol. 4. Jul. 2006, pp. 2857–2862.

[7] Y. Sugiyama *et al.*, "A high-speed CMOS image sensor with profile data acquiring function," *IEEE J. Solid-State Circuits*, vol. 40, no. 12, pp. 2816–2823, Dec. 2005.

[8] W. D. León-Salas, S. Balkir, K. Sayood, N. Schemm, and M. W. Hoffman, "A CMOS imager with focal plane compression using predictive coding," *IEEE J. Solid-State Circuits*, vol. 42, no. 11, pp. 2555–2572, Nov. 2007.

[9] M. S. Noohi, S. M. Sayedi, and A. Jalili, "A high-speed low-power multitask digital vision chip," in *Proc. 2nd RSI/ISM Int. Conf. Robot. Mechatronics*, Oct. 2014, pp. 161–165.

[10] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii, "A CMOS vision chip with SIMD processing element array for 1 ms image processing," in *IEEE Int. Conf. Solid-State Circuits (ISSCC) Dig. Tech. Papers*, Feb. 1999, pp. 206–207.

[11] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *IEEE Trans. Electron Devices*, vol. 50, no. 1, pp. 191–199, Jan. 2003.

[12] Y. M. Chi, U. Mallik, M. A. Clapp, E. Choi, G. Cauwenberghs, and R. Etienne-Cummings, "CMOS camera with in-pixel temporal change detection and ADC," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2187–2196, Oct. 2007.

[13] D. Kim, J. Cho, S. Lim, D. Lee, and G. Han, "A 5000S/s single-chip smart eye-tracking sensor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 46–47 and 594.

[14] S. Espejo, R. Carmona, R. Dominguez-Castro, and A. Rodriguez-Vazquez, "A 0.8 μm CMOS programmable analog-array-processing vision-chip with local logic and image-memory," in *Proc. 22nd Eur. Solid-State Circuits Conf.*, Sep. 1996, pp. 280–283.

[15] D. Ginhac, J. Dubois, B. Heyrman, and M. Paindavoine, "A high speed programmable focal-plane SIMD vision chip," *Anal. Integr. Circuits Signal Process.*, vol. 65, no. 3, pp. 389–398, Dec. 2010.

[16] R. Carmona, F. Jimenez-Garrido, R. Dominguez-Castro, S. Espejo, and A. Rodriguez-Vazquez, "A CMOS analog parallel array processor chip with programmable dynamics for early vision tasks," in *Proc. 28nd Eur. Solid-State Circuits Conf.*, Sep. 2002, pp. 371–374.

[17] T. Komuro, S. Kagami, and M. Ishikawa, "A new architecture of programmable digital vision chip," in *Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2002, pp. 266–269.

[18] M. Gottardi, N. Massari, and A. Simoni, "Programmable vision sensors with on-chip real-time image processing," in *Proc. IEEE Int. Workshop Imag. Syst. Techn.*, Apr. 2006, pp. 7–10.

[19] A. Lopich and P. Dudek, "A general-purpose vision processor with 160×80 pixel-parallel SIMD processor array," in *Proc. IEEE Custom Integ. Circuits Conf.*, Sep. 2013, pp. 1–4.

[20] P. Dudek, "A 39×48 general-purpose focal-plane processor array integrated circuit," in *Proc. Int. Symp. Circuits Syst.*, vol. 5. May 2004, pp. V-448–V-452.

[21] A. Lopich and P. Dudek, "A SIMD cellular processor array vision chip with asynchronous processing capabilities," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2420–2431, Oct. 2011.

[22] M. Suarez *et al.*, "CMOS-3D smart imager architectures for feature detection," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 4, pp. 723–736, Dec. 2012.

[23] F. D. V. R. Oliveira, H. L. Haas, J. G. R. C. Gomes, and A. Petraglia, "CMOS imager with focal-plane analog image compression combining DPCM and VQ," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 5, pp. 1331–1344, May 2013.

[24] J.-E. Eklund, C. Svensson, and A. Astrom, "VLSI implementation of a focal plane image processor-a realization of the near-sensor image processing concept," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 4, no. 3, pp. 322–335, Sep. 1996.

[25] N. Takahashi, K. Fujita, and T. Shibata, "A pixel-parallel self-similitude processing for multiple-resolution edge-filtering analog image sensors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 11, pp. 2384–2392, Nov. 2009.

[26] T. Komuro, S. Kagami, and M. Ishikawa, "A dynamically reconfigurable SIMD processor for a vision chip," *IEEE J. Solid-State Circuits*, vol. 39, no. 1, pp. 265–268, Jan. 2004.

[27] W. Miao, Q. Lin, W. Zhang, and N. J. Wu, "A programmable SIMD vision chip for real-time vision applications," *IEEE J. Solid-State Circuits*, vol. 43, no. 6, pp. 1470–1479, Jun. 2008.

[28] A. Elouardi, S. Bouaziz, A. Dupret, J. O. Klein, and R. Reynaud, "Image processing vision system implementing a smart sensor," in *Proc. 21st IEEE Instrum. Meas. Technol. Conf.*, vol. 1. May 2004, pp. 445–450.

[29] G. L. Cembrano, A. Rodriguez-Vazquez, R. C. Galan, F. Jimenez-Garrido, S. Espejo, and R. Dominguez-Castro, "A 1000 FPS at 128×128 vision processor with 8-bit digitized I/O," *IEEE J. Solid-State Circuits*, vol. 39, no. 7, pp. 1044–1055, Jul. 2004.

[30] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno, "1 ms column parallel vision system and its application of high speed target tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1. Apr. 2000, pp. 650–655.

[31] L. J. Kozlowski *et al.*, "Pixel noise suppression via SoC management of tapered reset in a 1920×1080 CMOS image sensor," *IEEE J. Solid-State Circuits*, vol. 40, no. 12, pp. 2766–2776, Dec. 2005.

[32] C. Shi *et al.*, "A 1000 fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array processor and self-organizing map neural network," *IEEE J. Solid-State Circuits*, vol. 49, no. 9, pp. 2067–2082, Sep. 2014.

[33] W. Zhang, Q. Fu, and N.-J. Wu, "A programmable vision chip based on multiple levels of parallel processors," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, Sep. 2011.

[34] H. Zhu and T. Shibata, "A real-time motion-feature-extraction image processor employing digital-pixel-sensor-based parallel architecture," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2012, pp. 1612–1615.

[35] Q. Lin, W. Miao, W. Zhang, Q. Fu, and N. Wu, "A 1,000 frames/s programmable vision chip with variable resolution and row-pixel-mixed

parallel image processors," *Sensors*, vol. 9, no. 8, pp. 5933–5951, 2009. [Online]. Available: http://www.mdpi.com/1424-8220/9/8/5933

[36] K. Sayood, *Introduction to Data Compression*. San Francisco, CA, USA: Elsevier, 2005.

[37] N. Cottini, M. Gottardi, N. Massari, R. Passerone, and Z. Smilansky, "A 33 μW 64×64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 850–863, Mar. 2013.

Joseph A. Schmitz received the B.S. degree in electrical engineering from the California Institute of Technology, Pasadena, CA, USA, in 2011, and the M.S. degree in electrical engineering from the University of Nebraska–Lincoln, Lincoln, NE, USA, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include low-power radiation detection and vision chip design.

Mahir K. Gharzai received the B.S. and M.S. degrees from the University of Nebraska–Lincoln, Lincoln, NE, USA, in 2010 and 2013, respectively, where he is currently pursuing the Ph.D. degree, with a focus in programmable parallel processing vision chips and with interests in high frame rate video applications, low latency VLSI design, and embedded device programming.

He was a Programmer with the National Security Agency, Fort Meade, MD, USA, and a United States Air Force Contractor in Kabul, Afghanistan.

Sina Balkır received the B.S. degree from Boğaziçi University, Istanbul, Turkey, in 1987, and the M.S. and Ph.D. degrees from Northwestern University, Evanston, IL, USA, in 1989 and 1992, respectively, all in electrical engineering.

From 1992 to 1998, he was an Assistant and Associate Professor with the Department of Electrical and Electronics Engineering, Boğaziçi University. He is currently a Professor with the Department of Electrical Engineering, University of Nebraska–Lincoln, Lincoln, NE, USA. His current research interests include CAD of VLSI systems, mixed-signal and analog VLSI design of sensory information processing systems, and focal plane imager arrays for smart camera applications.

Michael W. Hoffman received the B.S. degree from Rice University, Houston, TX, USA, the M.S. degree from the University of Southern California, Los Angeles, CA, USA, and the Ph.D. degree from the University of Minnesota, Minneapolis, MN, USA, all in electrical engineering.

From 1985 to 1988, he was a Signal Processing System Engineer with the Space Communications Division, TRW Inc., Livonia, MI, USA. In 1993, he joined the University of Nebraska–Lincoln, Lincoln, NE, USA, where he is currently a Professor. His current research interests include data compression, joint source channel coding, low-power mixed-signal processing applications, and sensor processing.

Daniel J. White (S'07–M'14) received the B.S., M.S., and Ph.D. degrees from the University of Nebraska–Lincoln, Lincoln, NE, USA, in 2005, 2006, and 2013, respectively, all in electrical engineering.

Since 2006, he has been the owner of White Audio, Valparaiso, IN, USA, an audio and electronics consultancy. He is currently an Assistant Professor of Electrical and Computer Engineering with the College of Engineering, Valparaiso University, Valparaiso, IN, USA. His current research interests include low-power electronics, mixed-signal processing, and federated satellite ground station networks.

Dr. White is a member of the American Society for Engineering Education and the Audio Engineering Society.

Nathan Schemm received the B.S. and Ph.D. degrees in electrical engineering from the University of Nebraska–Lincoln, Lincoln, NE, USA, in 2006 and 2010, respectively.

In 2010, he joined Texas Instruments Inc., Dallas, TX, USA, where he is currently a Senior Circuit Design Engineer. His current research interests include VLSI integration of wireless sensor networks, data converters, radiation detection applications, CMOS imagers, cellular neural networks, and ultrawideband technology.