A Real Chip Evaluation of a CNN Accelerator SNACC

Ryohei Tomura, Takuya Kojima, Hideharu Amano
Dept. of Information and Computer Science, Keio University, Japan
Email: wasmii@am.ics.keio.ac.jp
Ryuichi Sakamoto, Masaaki Kondo
Graduate School of Information Science and Technology, The University of Tokyo, Japan

***Abstract*—SNACC (Scalable Neuro Accelerator Core with Cubic integration) is an accelerator for deep neural network, which can improve the performance by increasing the number of stacked chips with inductive coupling wireless through chip interface (TCI). The chip implementation and real chip evaluation of SNACC are introduced. It consists of four processing element cores which execute dedicated SIMD instructions, distributed memory modules for storing weight data, and TCI. The real chip evaluation by using Renesas Electronics' 65nm SOTB (Silicon On Thin Box) CMOS technology appears that a simple CNN LeNet works at 50MHz for all layers with 0.90V supply voltage. The power consumption is less than 12mW. The performance can be enhanced by the forward body biasing about 15% in exchange for about 2mW leakage increasing. Also, SNACC achieved more than 20 times high performance to a MIPS R3000 compatible embedded processor.**

I. Introduction

In order to cope with recent advances in deep learning technologies, accelerators for convolutional neural network (CNN) have been widely researched. Unlike the trainig phase typically done in the cloud, various types of dedicated accelerators have been developed for the interface in the edge devices which require high energy efficiency. Eyeriss[1] is a CNN accelerator for embedded systems especially focusing on optimized 2D-convolutional execution. For acceleration of fully connected layers, DaDianNao[2] leaves out the necessity of off-chip memory accesses by having large eDRAM modules on a chip. EIE[3] cuts unnecessary trained weight parameters and the bit-width of them resulting in the reduction of the data size of the parameters without sacrificing the accuracy of the recognition.

We also proposed and implemented an accelerator called SNACC (Scalable Neuro Accelerator Core with Cubic integration) to achieve scalable performance improvement by changing the number of stacked chips[4]. It was designed with the following strategies.

- A dedicated light-weight microcontroller whose instruction set architecture is optimized for CNN operation is adopted.
- An execution engine which consists of a SIMD multiply and accumulate (MAC) unit with a variable bit-width operand is provided for implementing fully connected layers.
- Data re-usability of both convolutional and fully connected layers is exploited with two read buffers, one for stream like data and the other for temporal data.
- By providing wireless inductive coupling through chip interface (TCI), it can be easily connected with the host CPU and other SNACC chips to improve the performance.

In this paper, some evaluation results of the real chip fabricated by Renesas 65nm CMOS SOTB process is presented with the results of simulation for multiple chips.

II. Design of SNACC

In this section, we briefly introduce the target application and design of SNACC. The detail is described in the paper[4].

A. CNN overview

Various types of convolutional neural networks (CNNs) have been proposed for deep learning applications. Some of them have a lot of layers with complicated structure[5][6]. However, most of them consist of multiple layers with the following three types: the convolutional layer (Eq.(1)(2)), the pooling layer (Eq.(3)) and the fully connected layer (Eq.(4)(5)), which are expressed in the corresponding equations.

$$a_{n_o}[i,j] = \sum_{n_i}\sum_{p}\sum_{q}\omega_{n_i,n_j}[i,j]x[i+p,j+1] + b_n \quad (1)$$

$$y_{n_o}[i,j] = f(a_{n_o}[i,j]) \quad (2)$$

$$y[i,j] = max_{p,q}(x[i+p,j+q]) \quad (3)$$

$$a[i] = \sum_{j}\omega[i,j]x[j] + b_i \quad (4)$$

$$y[i] = f(a[i]) \quad (5)$$

In the equations, f(•) denotes an activation function. ReLU function f(x) = max(0,x) and sigmoid function $f(x) = 1/(1+e^{-1})$ are representative ones. SNACC is designed so that these three layers can be executed efficiently.

A typical CNN consists of several pairs of convolutional and pooling layers, and then fully connected layers follow. Convolutional and fully connected layers dominate the execution time of the CNN. However, their execution characteristics are different, especially in the data access re-usability.

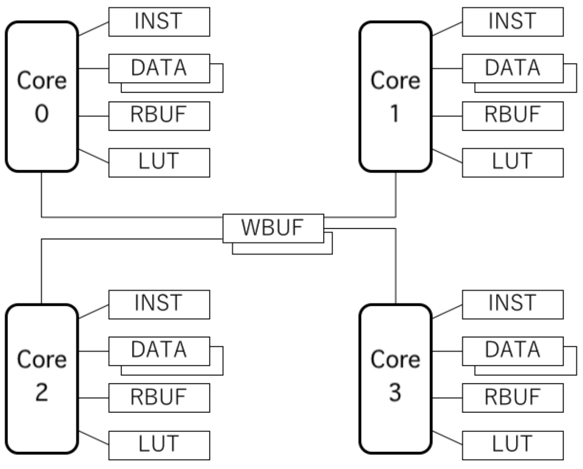

Fig. 1: The overview of SNACC with local memory modules

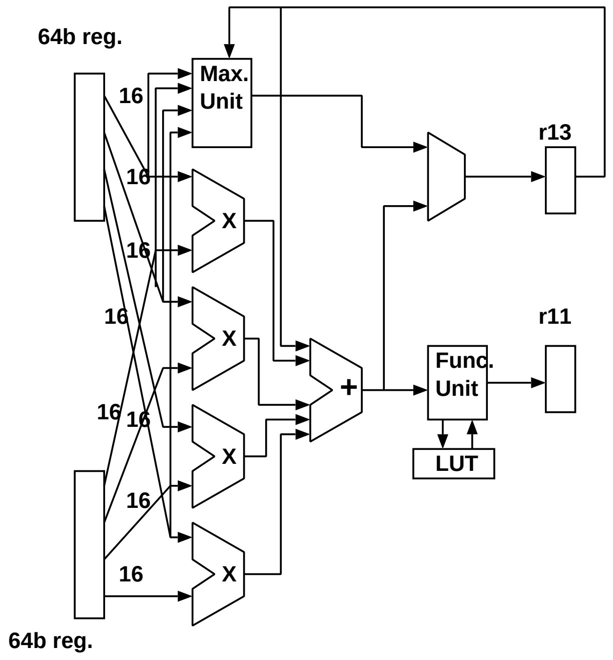

Fig. 2: SIMD unit for product-sum operation

B. Design of SNACC

SNACC consists of four SIMD cores, each of which implements its original instruction set and local memory designed for CNNs. The instruction bit width is 16-bit, and 16 general purpose registers are provided. Fig. 1 shows the schematic diagram of the local memory configuration of the SNACC. Each core has five memory modules, INST, DATA, RBUF, LUT, and WBUF, for instruction codes, input data, weight data, output data, and lookup table, respectively. DATA and WBUF are double-buffered so that the data transfer and processing can be overlapped. Each core including four local memories has an independent address space except WBUF. The address space of WBUF is shared with four cores. In this way, the computation results of each core can be shared.

Table I shows the examples of instructions supported by SNACC. The instruction set of the SNACC core consists of R-type (register-register) and I-type (register immediate) instructions. However, unlike the standard 32-bit RISC instruction set architecture, only two operands are specified. One of the most significant features of SNACC is SIMD instructions which perform the mad (multiply-add) instruction and the madlp (multiply - add with loop) instruction. Also, This instruction set has the dbchange(change double buffer) instruction and the dma(issue a DMA request) instruction. Using these instructions, we can reduce control overhead when calculating multiply-accumulation. Examples of control overhead include address calculation for accessing target data, loop control and processing conditional branches. In the case of SNACC, we implement these processes and the SIMD multiply-calculation operation sequence in the hardware and include them in the multi-cycle custom SIMD arithmetic instructions. As the result, it is more efficient than software implementation using universal instructions set in terms of accelerating of CNN detection and power consumption. Fig. 2 shows the SIMD unit for the mad instruction. The SIMD unit can handle fixed-point arithmetic four 16-bits data or eight 8-bits data. Each multiplier unit receives two input data from the DATA and RBUF memory, and then, these products are summed by an adder unit. The Max unit selects the maximum value from all the inputs. Output data from the adder unit and the max unit are chosen by a multiplexer and is stored in the register *r13*. In addition, an activation function defined by the lookup table is applied to output data from the adder, and the result is stored in the register *r11*. The madlp instruction iterates this mad instruction for a specified number of times. For controlling the SIMD instructions, each core provides eight 8-bits control registers, and four 32-bits SIMD registers.

C. TCI and the chip stack

SNACC provides an inductive coupling channels TCI[7] to form a link with stacked chip. Square coils implemented with arbitrary metal layers are used as data transceivers, and no special fabrication technology is needed. Data are transferred through magnetic field between two chips by overlapping a transmitter coil over a receiver coil that are placed in different chips. A pair of driver and inductor for sending data is called the TX channel, while a pair of receiver and inductor for receiving data is called the RX channel. An inductive coupling channel is usually formed by a coil for the high-frequency synchronization clock, and a coil for data transfer. A high-frequency clock (1 - 8 GHz) is generated by a ring oscillator, and the serialized data are transferred following the high-frequency clock directly through the driver. Since this paper focuses on the chip implementation and evaluation of SNACC as a single chip, the detail explanation on TCI is omitted. For more details on the TCI, please refer papers [7][8].

TABLE I: Typical instructions supported by the core

inst	description
loadi	load immediate
bneq	branch not equal
jump	jump (PC-relative)
mad	SIMD multiply-accumulate
madlp	SIMD multiply-accumulate for loop
setcr	set control register
addi	add immediate
subi	subtract immediate
sll	shift left logical
srl	shift right logical
sra	shift right arithmetic
add	add register
sub	subtract register
mul	multiple register
and	and register
or	or register
xor	xor register
readcr	read control register
dbchange	change double buffer
dma	issue DMA request
loadv	load vector
loadh	load 16-bits data
loadw	load 32-bits data
storeh	store 16-bits data to WBUF
storew	store 32-bits data to WBUF

TABLE II: Spec. of SNACC

Process	Renesas 65nm DLSOTB_V3 CMOS 7 Metal
Area	3mm $\times$ 6mm
Chip Thickness	80μm
Target Freq.	50MHz
CAD	Synopsys Design Compiler 2016.03-SP4 Synopsys IC Compiler 2016.03-SP4

D. Chip implementation

SNACC was implemented by using Renesas 65nm SOTB process[9]. SOTB (Silicon on Thin BOX) is a low power CMOS FD-SOI (Silicon on Insulator) process which can control the threshold of transistors with body biasing. The specification of the SNACC is shown in Table II.

As shown in Fig 3, it is implemented on the 3mm $\times$ 6mm die. Red frames show four cores, and blue one is the TCI IP.

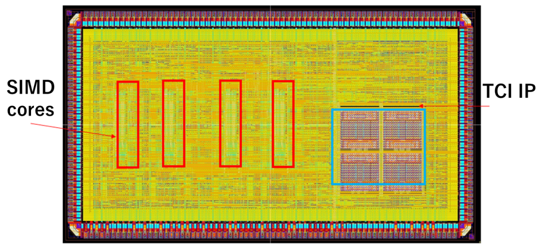

Fig. 3: The layout of SNACC

III. REAL CHIP EVALUATION

A. LeNet

In this paper, a simple CNN, LeNet, is used for evaluating SNACC. For implementing more recent sophisticated CNNs, we are now developing programming environment[10]. However, now we rely on an assembly language. LeNet is an old network proposed by Yann LeCun[11]. It consists of 6 layers: two convolution layers, two pooling layers and two fully connected layers (classifier) as shown in Figure 4. Here, we use "Max-pooling" for pooling layers. ReLU and softmax activation are introduced to generate the final results.

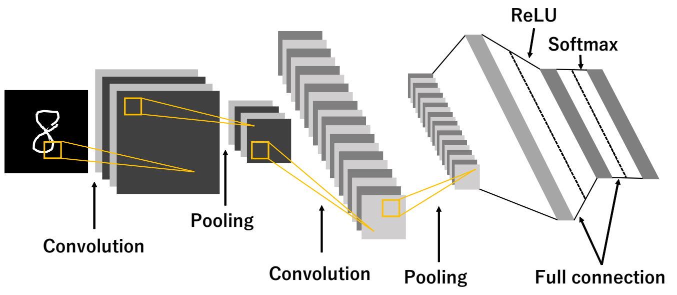

Fig. 4: The structure of LeNet

LeNet was proposed with the aim of the classification of hand-written numbers, so we use MNIST datasets as input image. The size of input images is 28×28. Figure 5 shows an input image and inference result. The output data are generated as a probability of possible candidates.

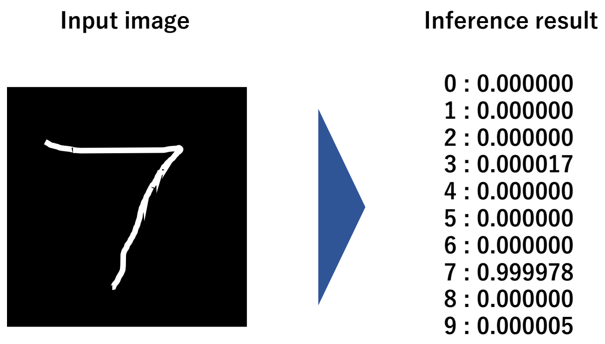

Fig. 5: An example of input image

Unlike the original LeNet, we used $20 \times 3 \times 3$ convolution kernel in the first convolution layer, and $50 \times 20 \times 3 \times 3$ convolution kernel in the second convolution layer. In both of max-pooling layers, we adopt 2×2 kernel and set $stride = 1$. We applied 500×1800 weights in the first fully connected layer, and 10×500 weights in the second fully connected layer.

To evaluate SNACC's power consumption, we implemented each layer with the assembly language for SNACC. Code1 shows an example assembly code for the first fully connected layers. The meaning for each instruction in Code 1 is described in Table I.

Code 1: Assembly Code for the 1st fully connected layer

```
setcr,r4,04,
setcr,r1,FF,
setcr,r2,08,
setcr,r3,08,
setcr,r6,00,
xor,rTR0,rTR0,
xor,r0,r0,
loadi,r0,01,
sll,r0,18,//set dmem memory address
xor,r1,r1,
loadi,r1,05,
sll,r1,18,//set rbuf memory address
xor,r2,r2,
loadi,r2,09,
sll,r2,18,
xor,r6,r6,
readcr,r6,r0,
xor,r7,r7,
loadi,r7,01,
sll,r7,09,
mul,r6,r7,
add,r2,r6,//set wbuf memory address
xor,r3,r3,
loadi,r3,C8,
loadv,r0,r1,
madlp,r3,00,//madlp calculation
storeh,rTR0,r2,//store value to wbuf
    memory
xor,rTR0,rTR0,
addi,r2,02,//move pointer to next wbuf
    address
loadi,r1,64,
sll,r1,04,
loadv,r0,r1,
madlp,r3,00,//madlp calculation
storeh,rTR0,r2,//store value to wbuf
    memory
halt,r0,r0,
```

B. Evaluation Environment

Figure 6 shows our evaluation environment in this work. We set target board which inculde SNACC and FPGA equipped with Artix-7 as host controler on the mother board.

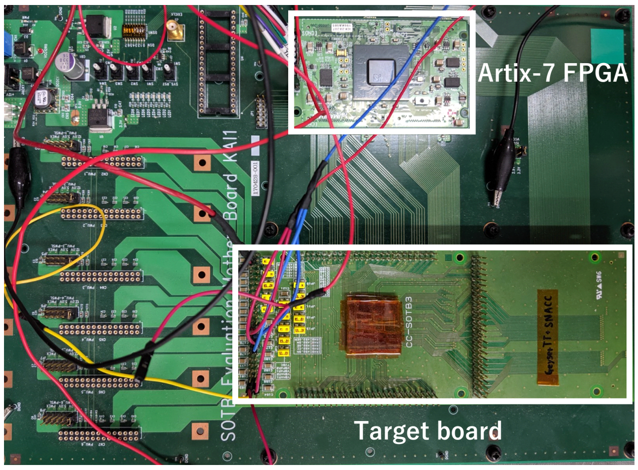

Fig. 6: Evaluation Environment

C. Operational Frequency

Renesas SOTB 65nm LSTP(Low Standby Power) process focuses on leakage reduction, and the operational frequency is not so high even with 0.75V standard power supply voltage(VDD). Figure 7 shows the required supply voltage for achieving the target operational frequency when using zero body biasing and 0.4V forward body biasing. For achieving higher operational frequency, we need to give large supply voltage. 50MHz was achieved within 0.90V for all layers. As shown in this figure, all layers required similar VDD to each operational frequency.

Figure 8 shows the power consumption when each program is working at the given operational frequency using zero body biasing and 0.4V forward body biasing. It shows that the power consumption for each applications is less than 12mW.

However, at each operational frequency, the convolution layer achieved less energy than other layers, even though it is said the convolution layer consumes larger energy than other layers in general. It was caused by the low utilization of SIMD units in this layer. Increasing it is one of our future work.

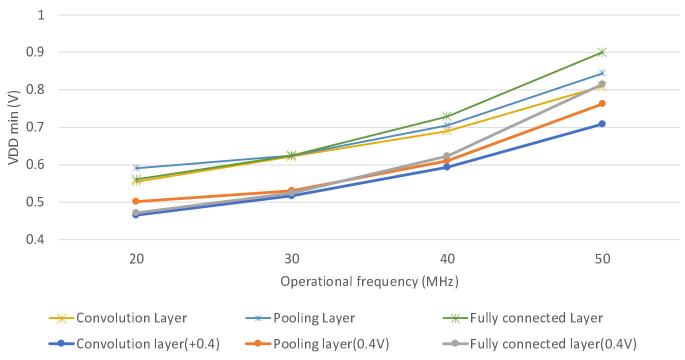

Fig. 7: The required VDD versus operational frequency when using zero and 0.4V forward body biasing

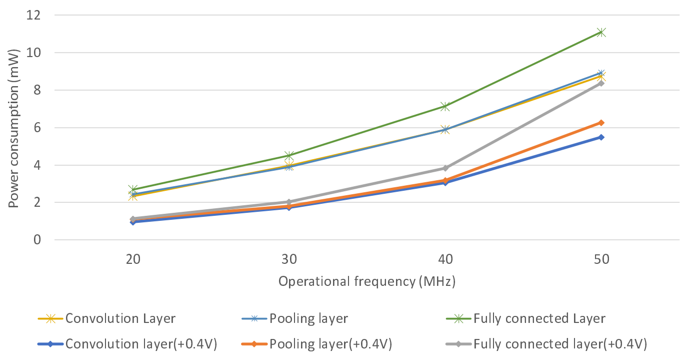

Fig. 8: The power consumption versus operational frequency when using zero and 0.4V forward body biasing

D. Leakage Current

The most important characteristics of the SOTB process is high controllability of body biasing. There are a lot of studies to make the use of body biasing for the optimization of power consumption[12]. However, recent SOTB (LSTP) process shifts to reducing leakage power by increasing the threshold of each transistor. Even with the zero bias, which gives the same voltage as the GND (VBN=0) to NMOS and VDD (VBP=VDD) to PMOS, the leakage current is quite small. Figure 9 shows the leakage current versus the body bias to NMOS (VBN). Here, we used the balanced bias, that is (VBP = VDD-VBN), so only values of VBN are shown. It is apparent that the leakage current is well suppressed even with the strong forward biasing.

We can improve the operational frequency by giving the strong forward biasing. From the evaluation, it appears that the fully forward biasing improved the operational frequency by about 15%. Considering the current increasing shown in Figure 8, using forward body biasing is a better solution for improving the energy.

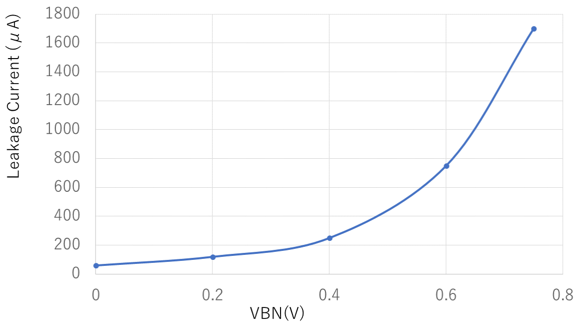

Fig. 9: Leakage current versus the body bias voltage

E. *Performance evaluation using the forward body biasing*

In this subsection, we evaluate SNACC with forward body bias control. We used 0.2V and 0.4V forward body biasing. From figure 7, in this case, we can see the required VDD is lower than the case using zero body biasing. The same frequency can be achieved less than 0.85V VDD with 0.2V forward body biasing.

We suggested using body biasing is better solution for improving the energy in the last subsection. It is certified as shown in Figure 8 which shows the power consumption with forward body biasing. In this case, 50MHz operation can be achieved within 9mW for all layers even with 0.2V forward body biasing.

F. *Performance Improvement*

Lastly, we evaluate the performance of SNACC from the viewpoint of the execution time. We compared the SNACC to the host processor, GeyserTT[13], which is a MIPS R3000 compatible embedded processor. Table III shows the execution time of the convolution computation for a certain amount of data. As shown in the table, SNACC achieved more than 20 times higher performance than GeyserTT. This is because of the four SIMD cores and special instructions of SNACC. From the result, the basic design concept of SNACC works efficiently. Since the power of GeyserTT is about 30mW at 50MHz, the energy efficiency of SNACC is more than 30 times.

TABLE III: Execution Time

SNACC[ns]	GeyserTT[ns]	GeyserTT/SNACC
58,999	1,588,575	26.925

IV. Conclusions

In this paper, the real chip evaluation results of the convolutional neural network accelerator, SNACC, were presented. The simple CNN LeNet runs at 50MHz for all layers with 0.90V supply voltage. The power consumption is less than 12mW. The performance can be enhanced by the forward body biasing about 15% in exchange for about 2mW leakage increase. And we achieved making LeNet run at 50MHz for all layers using under 9mW power consumption. SNACC achieved more than 20 times higher performance to a MIPS R3000 compatible processor GeyserTT.

This evaluation focuses on the single chip SNACC and ignored the communication between host processor though the TCI. Now, a chip stack with a host CPU of GeyserTT and a SNACC chip is available. The whole system evaluation and comparison with other CNN accelerators are our future work. Developing programming environment is also our important future work.

Acknowledgment

This work is supported by VLSI Design and Education Center(VDEC), the University of Tokyo in collaboration with Synopsys, Inc and Cadence Design Systems, Inc. Also, this work is partly supported by JSPS Kakenhi (B) 18960959 Chip stacking methods for Building Block Computing Systems.

References

[1] Y.-H. Chen, T. Krishna, J. S.Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, 2017.

[2] Y. Chen and et.al, “Dadiannao: A machine-learning supercomputer,” *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014.

[3] S.Han, X.Liu, H.Mao, J.Pu, and A.Pedram, “Eie: Efficient inference engine on compressed deep neural network,” *43rd Intenernational Symposium on Computer Architecture*, 2016.

[4] R.Sakamoto, R.Takata, J.Ishii, M.Kondo, H.Nakamura, T.Ohkubo, T.Kojima, and H.Amano, “The design and implementation of scalable deep neural network accelerator cores,” in *Proc. of IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC-17)*, Sep. 2017.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

[6] C. Szegedy and et.al, “Going deeper with convolutions,” *2015 IEEE Conference on Computer Vison and Pattern Recognition*, 2015.

[7] Y. Take and et.al., “3-D NoC with Inductive-Coupling Links for Building-Block SiPs,” *IEEE Transactions on Computers (TC)*, vol. 63, no. 3, pp. 748–763, Mar. 2014.

[8] N. Miura and et al, “A Scalable 3D Heterogeneous Multicore with an Inductive ThruChip Interface,” in *IEEE Micro, Vol.33, No.6*, 2013, pp. 6–15.

[9] Takashi Ishigaki, et al., “Ultralow-power LSI Technology with Silicon on Thin Buried Oxide (SOTB) CMOSFET,” *Solid State Circuits Technologies, Jacobus W. Swart (Ed.), ISBN: 978-953-307-045-2, InTech*, pp. 146–156, 2010.

[10] T.Okubo, M.Sit, H.Amano, R.Takata, R.Sakamoto, and M.Kondo, “A software development environment for a multi-chip convolutional network accelerator,” *International Journal of Computer Application*, June 2017.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. of the IEEE*, Nov. 1998.

[12] H. Okuhara, A. Ben Ahmed, and H. Amano, “Digitally assisted on-chip body bias tuning scheme for ultra low-power vlsi systems,” *IEEE Transactions on Circuits and Systems I: Regular Papers.*, 2018.

[13] S.Hamada, A.Koshiba, M.Namiki, and H.Amano, “Building block operating system for 3d stacked computer systems with inductive coupling interconnect,” *ISOCC*, Dec. 2017.