Group 6: In-memory classifier based on dot-product calculation in 4 × 8 8T-SRAM Array

Jiajun Cao, Rui Sun, Wenbo Duan, Guoqin Yin, Tingjie Yu

Abstract

Our project has achieved an in-memory classifier based on dot-product calculation with 2-bit weight precision on a 4 × 8 8T SRAM Array. The motivation is to realize dot-product calculation of high computation throughput which is required for many applications, especially in machine learning [1]. With the design of 8T-SRAM array, DACs and ADCs, we expect to implement vector-matrix dot product with 2-bit precision of the weights which is indicated by analog voltage outputs. Furthermore, with different analog output voltages, a classifier of 6 classes is achieved.

Introduction

We use regular 8T SRAM in our design and connect the RWL to the input voltage which is from DAC (as shown in figure.1(a)). By sizing each cell from large to small, multibit weight can be stored in SRAM array (as shown in figure.1(b)).In our design, we group every 2 SRAM bit-cell as weight(2 bits) storage by differently sizing the read port transistors. In classifier mode, DAC will convert digital data vector to an analog voltage *Vin* which would be applied to SRAM RWL and the precharge pmos will pull up the output node. According to the stored weight and different *Vin*, the precharged read bitline will be discharged to a certain fixed value of voltage

due to the competitive relationship between precharge PMOS and pull down NMOS as shown in figure.1(c). In addition, for the different sizing of read ports in the 2-bit SRAM cell, larger transistor contributes larger current which further pull down the RBL voltage. Thus, with the same inputs *Vin* being applied to each row, the RBL voltage will also be different according to the weight stored in SRAM bit-cell. Finally, we use latched comparators with different reference voltages to classify different analog RBL voltages. For each column, there would be about 34 kinds of possible conditions according to the dot-product. However, in our simulation, we can classify the first 30 different results and the last four results are not stable. Therefore, we divide those 30 analog voltages into 6 groups and generate 3-bit digital data.

Design and implementation

DAC

As shown in figure2.(a), we implement a 2 bit DAC[2]. With different data input, there will be different currents flow down from the pull-up network. The self-bias nmos in pull-down network will generate a stable voltage level, which is connected to RWL of SRAM array, due to different current. We size the pmos in pull-up network and adjust bias voltage *Vbias* to generate a corresponding voltage signal of the 2-bit input data. Thus, it functions as a simple 2-bit DAC. *classen* is the enable signal for classify mode. When *classen*=0 and *WLReset=1*, DAC doesn't work, SRAM array performs its normal role as data storage (SRAM mode).

SRAM Array

Considering a 2-bit weight stored in 2 grouped SRAM bit-cell as shown in figure2.(c) (2 grouped SRAM bit-cell means one SRAM's RBL is connected to the other and each bitline has a control transistor to switch between classify mode and SRAM mode), we size the MSB SRAM read port and LSB SRAM read port 860nm and 280nm respectively. As shown in figure2.(c), when the cell stores 00, NMOSs(gate is the weight data stored in SRAM) are off, the voltage of RBL could not discharge, so the voltage remains to be *Vdd*. When the cell stores 10, 01 or 11, the precharged line RBL is discharged through the path where both NMOSs(one's gate is controlled by *Vin*, the other's gate is controlled by weight bit) are on. So due to different weight values and *Vin*, we can get different currents flowing through bitline RBL which corresponds to different analog voltages.

The timing is typically like this: When Classen changes from 0 to 1, WLReset changes from 1 to 0, DAC converts digital data to analog voltages, the input analog voltage and the weights lead to different voltages on the RBL which can reflect a result of dot product between input 2-bit data vector and 2-bit 4×4 weight matrix. Then, the RBL voltages are sampled by ADCs.

ADC

The ADC in our design consists of voltage references and latched comparators (shown in figure2.(b)). When *latch* signal is high, comparators compare the input RBL voltage with different reference voltages. When the *latch* signal is 0, the comparators are pre-charged and do not consume dynamic power consumption. The latch signal will only be on when CLK is at the negative edge, because when CLK is at the positive edge the output voltage is not stable and sampled signal might be wrong. Figure 3(a) shows our full schematic. We added some input and output buffers to minimize delay and strengthen the signal.

layout

As shown in figure 3.(b), the layout arrangement requires ADC, DAC, voltage reference and SRAM blocks. Due to the asymmetry of our sized 2-bits SRAM, we did not follow conventional ways to design the layout of SRAM. With some adjustments, we would eventually integrate our classifier into our final RISC processor.

Results and conclusion

We implemented a 4 × 8 8T-SRAM array which can be both used as a classifier and a normal SRAM. The inner product classifier supports 30 different values, and an ADC divides the analog voltages into 6 classes. The classification results are shown in Figure 4(a) and the Monte Carlo simulations are shown in Figure 5. As the calculation results increase, the voltage drops gradually (from 1.16 V to 516 mV roughly). Our best Monte Carlo result of boundary is 14/20 (calculation results is 15) and worst simulation result is 7/20 (calculation results is 27). The simulation result is acceptable. Also we did 6-sigma Monte Carlo simulation when the classifier is on SRAM mode. All results within 100 simulations are correct.

8	**10/20**	**9**	**13/20**
15	**14/20**	**16**	**9/20**
18	**10/20**	**19**	**13/20**
24	**12/20**	**25**	**13/20**
27	**7/20**	**28**	**10/20**

Table1: Monte Carlo simulation of classification boundaries.

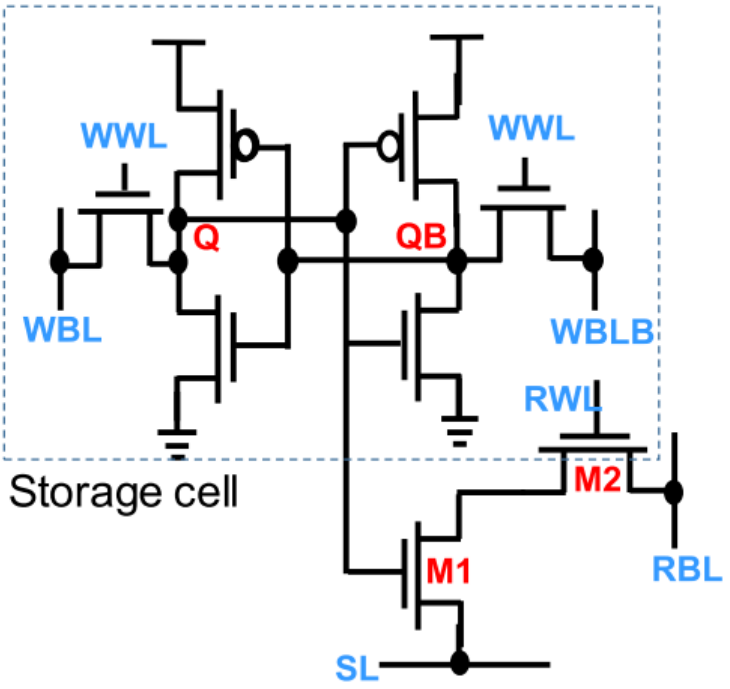
The design can be further improved. The first is to improve the stability and precision of comparators. ADC with higher resolution can increase the quantities of classification. The second is to guarantee the linearity of the analog voltage on RBL corresponding to the digital result. Under ideal conditions, we assume there is no current flowing between RBLs in one cell. However, in 2-bit weight cells, the current is unavoidable. With the application of well-adjusted operational amplifiers or 1-bit SRAM instead of 2-bit SRAM. The linear relationship may be improved. The third one is to rise VDD which can give us a wider voltage range to classify.

References：

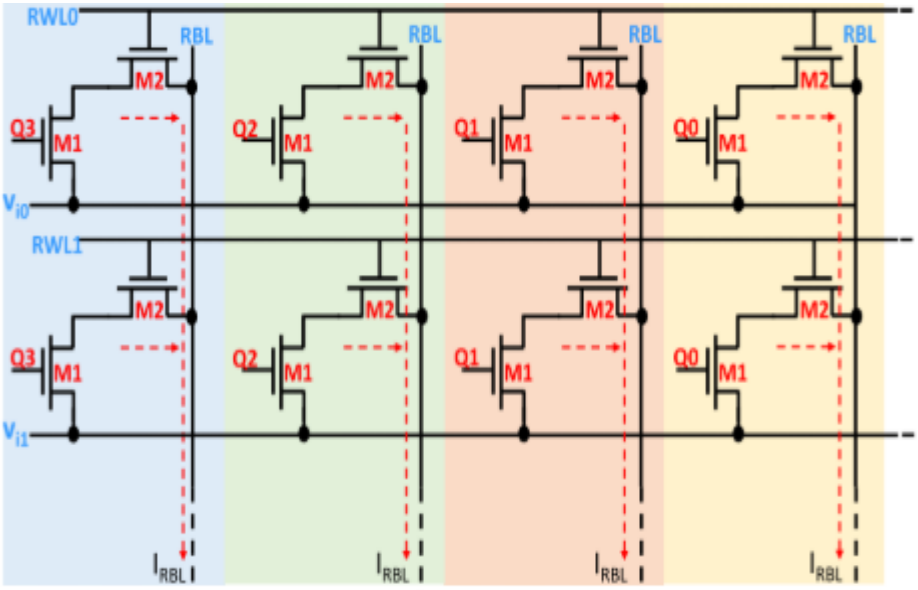
[1] Akhilesh Jaiswal , Indranil Chakraborty , Amogh Agrawal , and Kaushik Roy, "8T SRAM Cell as a Multibit Dot-Product Engine for Beyond Von Neumann Computing，" *IEEE Transactions On Very Large Scale Integration (VLSI) Systems, vol.* 27, no. 11, November 2019

[2] Jintao Zhang, "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array," in *IEEE Journal Of Solid-State Circuits*. 2017

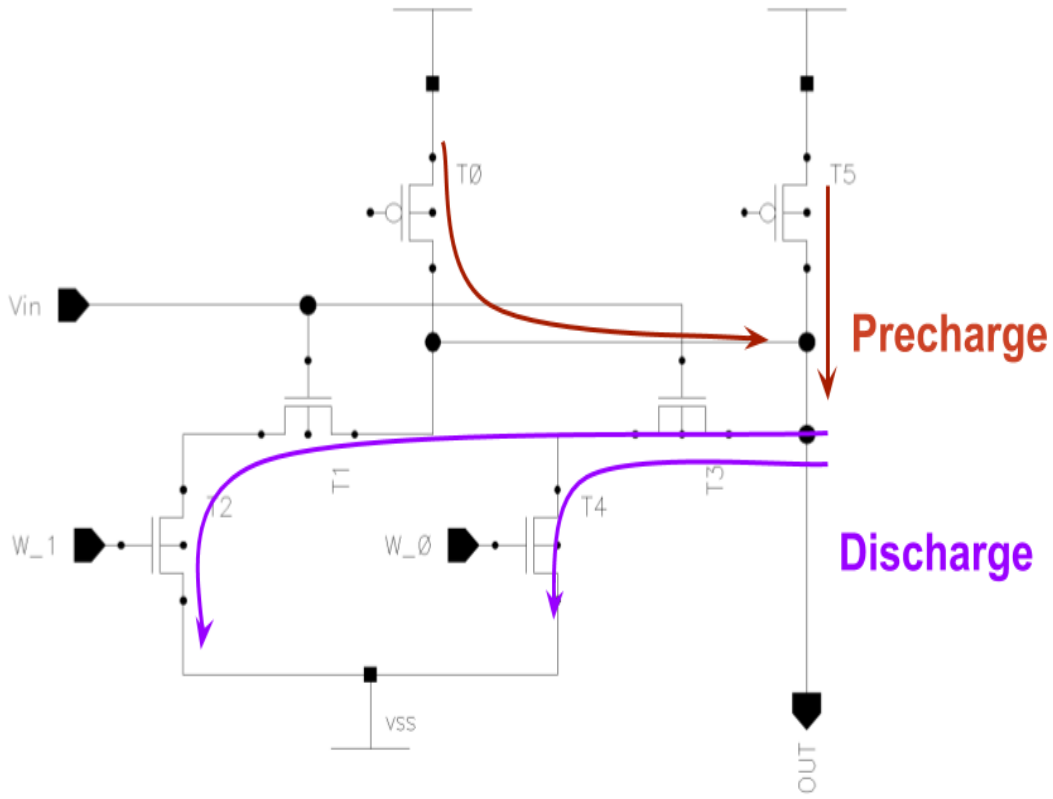
[Figure 1]：



(a).8T SRAM array

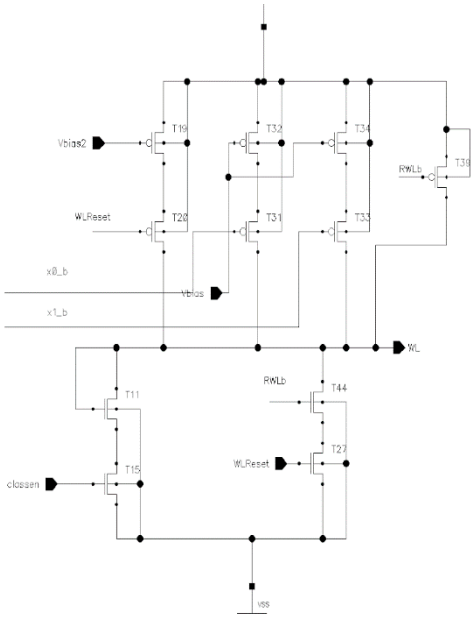


(b).Multibit weight storage in SRAM array

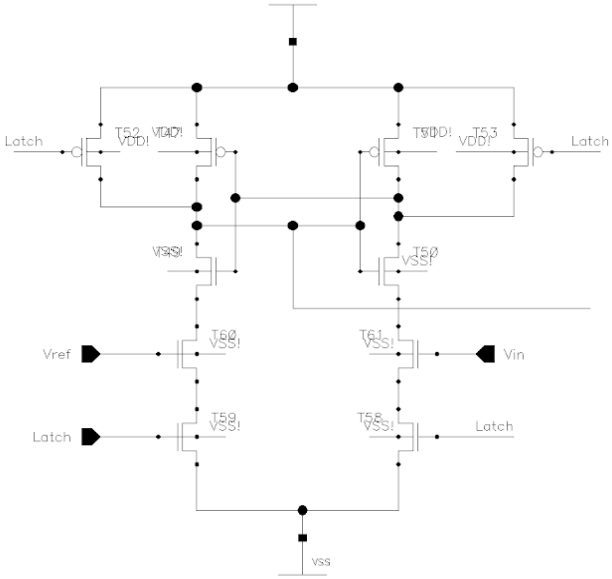


(c).Stablize output node voltage due to the competitive relationship of precharge pmos and pull down nmos.

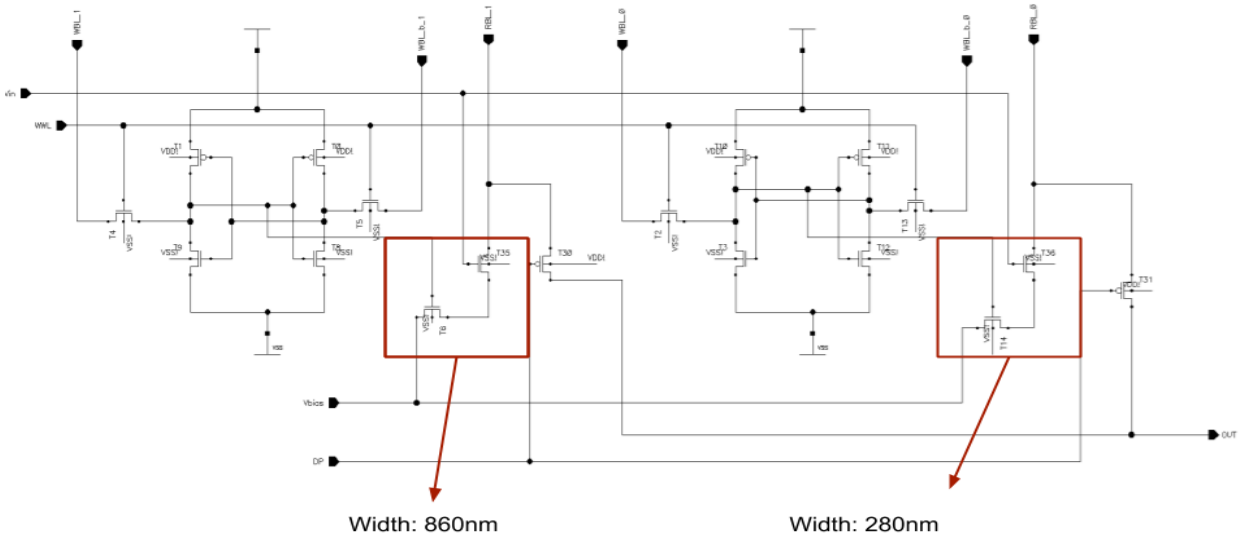
[Figure 2]



(a). DAC

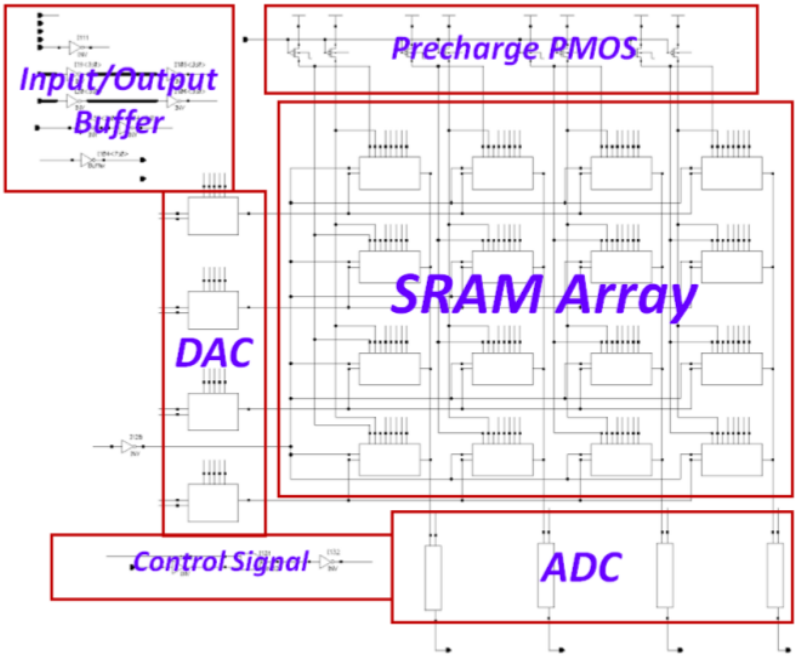


(b).Comparator

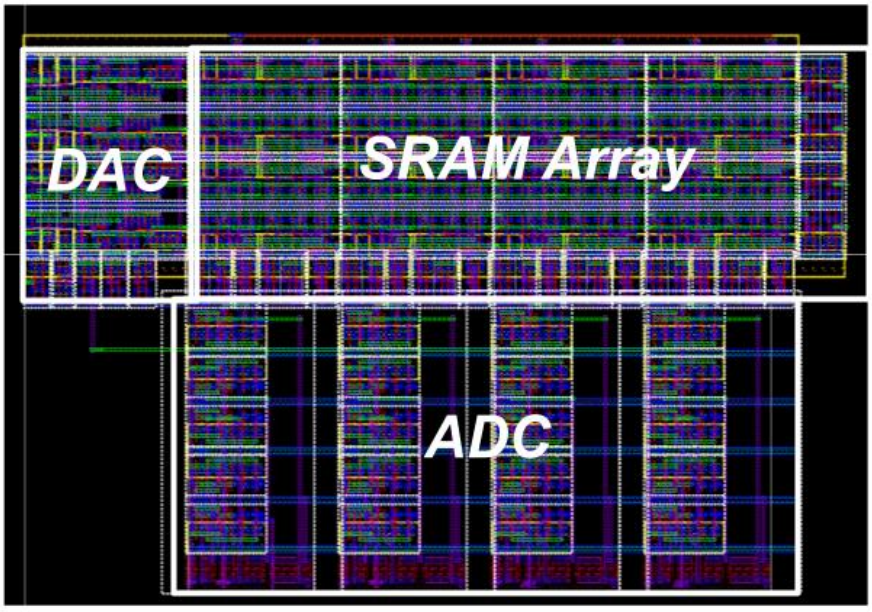


(c) Grouped SRAM bit cell(stand for one 2-bit weight)

[Figure 3]

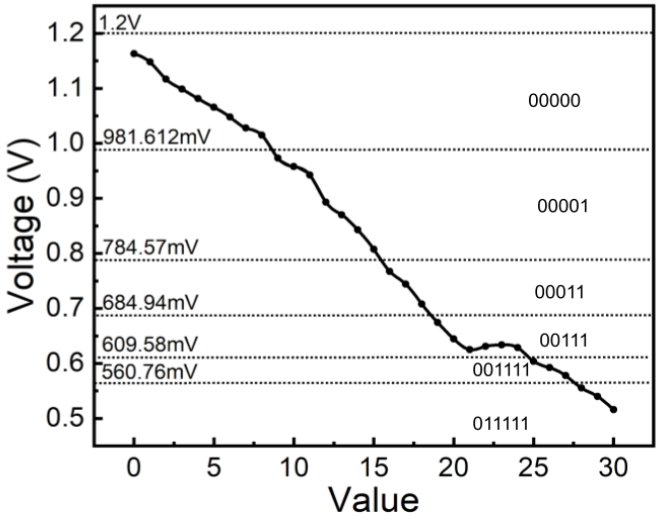


(a). Full schematic

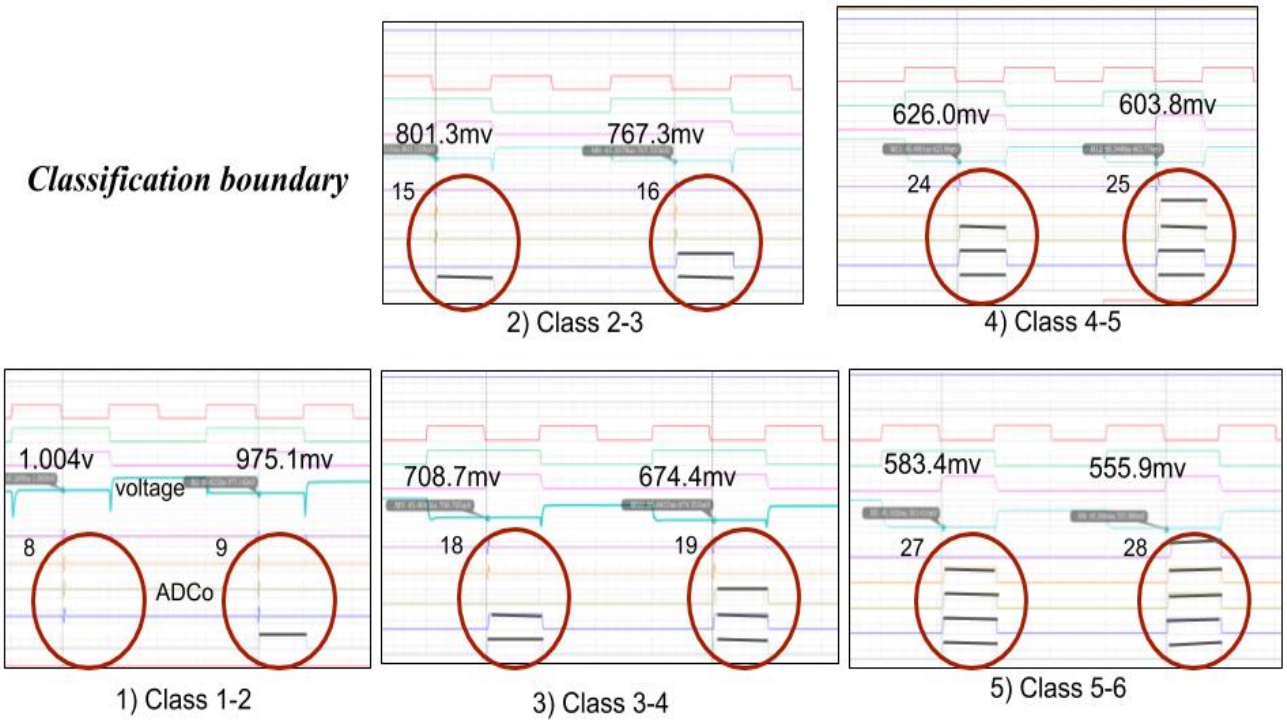


(b). Full layout

[Figure 4]

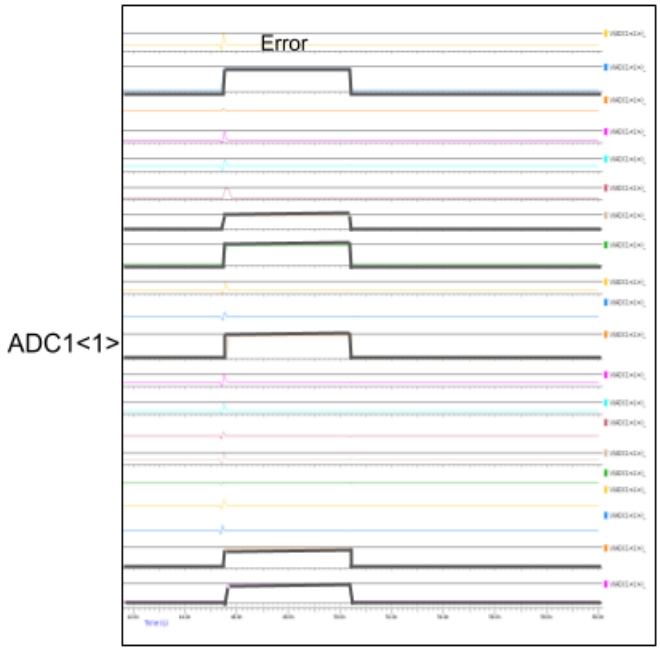


(a). Relationship between calculation results and output voltage

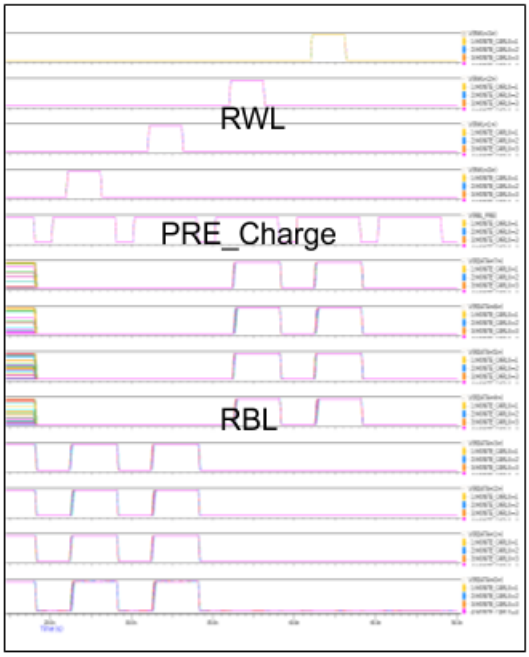


(b). Simulation results of classify boundary

[Figure 5]



Classifier result: 15(1*3+1*3+1*3+2*3)
Monte Carlo result: 14/20



SRAM Monte Carlo simulation with 6-sigma variation

Monte Carlo simulation results:

[Figure 6]

Full layout after integration

a: DMEM. ***b***: Controller. ***c***: SRAM classifier. ***d***: PC. ***e***: IMEM. ***f***: Datapath