

SMVPU Ultra System (16-bit)
Signal Journey map
Input $\rightarrow$ processing $\rightarrow$ output.

Step 1:

Entry Point: Digital Input Signal (16-bit)

Input-vector [255:0] = [0x8000, 0x6000, 0xA000, 0x4000, 0xC000,
0xE000, 0x3000, 0x5000, 0x7000, 0x9000,
0xB000, 0xD000, 0xF000, 0x1000, 0x9000]

Format: 16 elements $\times$ 16-bit = 256 bits total

Purpose: Input data for matrix-vector multiplication

Timing: $t = 0$ ns (start of operation)



FSM Controller Activation

State IDLE $\rightarrow$ Decode $\rightarrow$ Prepare $\rightarrow$ MVM

Instruction: (0x03 (MVM operation))

Control Signals: [0, 0, 1, 1, 0, 0, ...] (32-bit)

what happens:

- FSM receives MVM instruction
- Decode operation type
- Prepares system for matrix-vector multiplication
- Generates Control Signals for all blocks

Timing: $t = 0-2$ ns

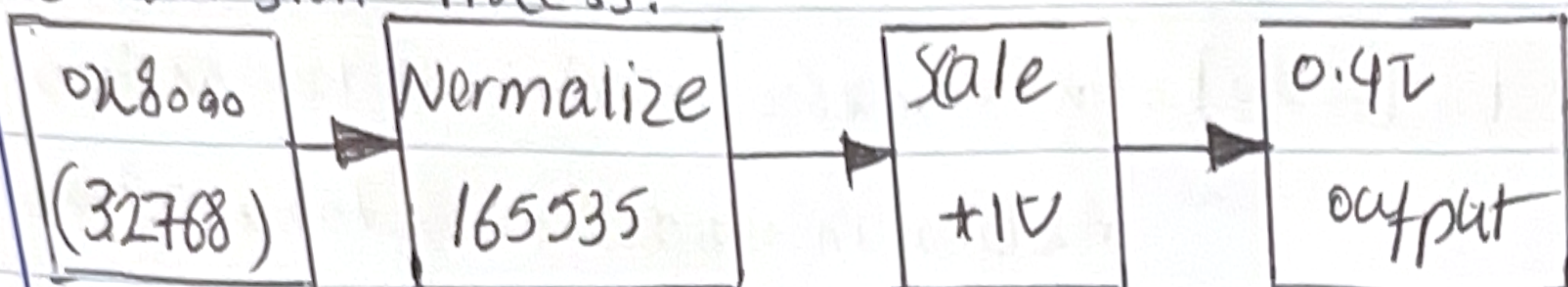
Step 2: DAC

DAC Array (16x16-bit Digital-to-Analog Converter)

Input: Digital values [0x8000, 0x6000, 0xA000, 0x4000, ...]

Output: Analog voltage [0.4V, 0.3V, 0.5V, 0.2V, ...]

Conversion Process:



Detailed Conversion

$$0x8000 (32768) \rightarrow 32768 / 65535 = 0.500 \rightarrow 0.500 \times 1V = 0.500V$$

$$0x6000 (24576) \rightarrow 24576 / 65535 = 0.375 \rightarrow 0.375 \times 1V = 0.375V$$

$$0xA000 (40960) \rightarrow 40960 / 65535 = 0.625 \rightarrow 0.625 \times 1 = 0.625V$$

$$0x4000 (16384) \rightarrow 16384 / 65535 = 0.250 \rightarrow 0.250 \times 1 = 0.250V$$

Purpose: Convert digital input to analog voltage for crossbar
Timing: $t = 2.4ns$

Wordline Distribution

WL[0] = 0.500V	WL[1] = 0.375V	WL[2] = 0.625V	WL[3] = 0.250V
WL[4] = 0.750V	WL[5] = 0.125V	WL[6] = 0.875V	WL[7] = 0.188V
WL[8] = 0.313V	WL[9] = 0.438V	WL[10] = 0.563V	WL[11] = 0.688V
WL[12] = 0.813V	WL[13] = 0.938V	WL[14] = 0.063V	WL[15] = 0.563V

What happens:

- Each wordline carries input voltage corresponding row
- 16 wordlines connect to 16 rows of crossbar
- Voltages represent input vector elements (16-bit precision)
- Timing $t = 4ns$ (settling time)

Step 3: Crossbar 16x16 (1T1M):

Crossbar Core 16x16 (256 1T1M Cells)

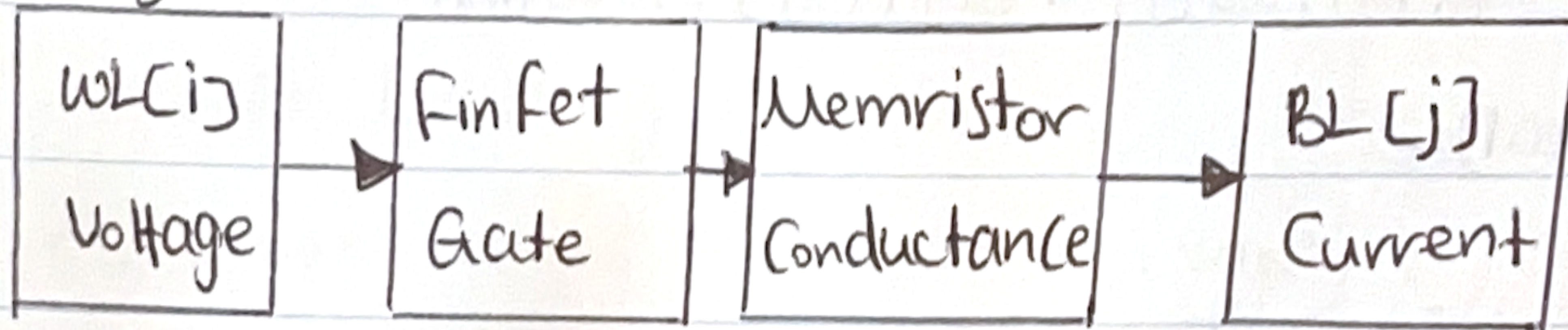
Matrix-Vector Multiplication: $Y = W * X$

Input Vector x : $[0.502V, 0.376V, 0.627V, 0.251V, \dots]$

Weight Matrix w : $[G_{11}, G_{12}, G_{13}, \dots, G_{16}, 16]$ (Conductance Values)

Output Current I : $[I_1, I_2, I_3, \dots, I_{16}]$ (Sum of Currents)

Physical Process In Each 1T1M Cell:



Detailed Calculation For Column 0:

- Cell $[0,0]$: $I = 0.500V * G[0,0] = 0.500V * 1.2mS = 0.600mA$
- Cell $[1,0]$: $I = 0.375V * G[1,0] = 0.375 * 0.8mS = 0.300mA$
- Cell $[2,0]$: $I = 0.625 * G[2,0] = 0.625 * 1.5mS = 0.938mA$
- ... (16 Cells in Column 0)
- Total $I[0] = 0.600 + 0.300 + 0.938 + \dots = 2.850mA$

Quantum Effects In Memristors:

- Quantum tunneling: Additional Current due to quantum effects
- Superposition: Multiple quantum states contribute to Conductance
- Temperature: 77K Cryogenic operation reduces noise

Final output Currents:

$BL[0] = 2.847mA$	$BL[1] = 3.125mA$	$BL[2] = 2.634mA$	$BL[3] = 2.19mA$
$BL[4] = 2.850mA$	$BL[5] = 2.445mA$	$BL[6] = 2.892mA$	$BL[7] = 3.067mA$
$BL[8] = 2.756mA$	$BL[9] = 2.834mA$	$BL[10] = 2.678mA$	$BL[11] = 2.945mA$
$BL[12] = 2.523mA$	$BL[13] = 3.011mA$	$BL[14] = 2.456mA$	$BL[15] = 2.723mA$

purpose: Perform matrix-vector multiplication in hardware
Timing: $t = 4-14$ ns (10ns computation time)

Bitline Current Collection

Each bitline collects current from all cells in its column.

$$BL[0]: I = \sum_{i=0}^{15} [WL[i] * G[i*0]] = 2.847mA$$

$$BL[1]: I = \sum_{i=0}^{15} [WL[i] * G[i*1]] = 3.125mA$$

$$BL[2]: I = \sum_{i=0}^{15} [WL[i] * G[i*2]] = 2.634mA$$

... (16 bitlines total) ...

what happens:

- Currents flow from wordlines through memristors to bitlines
- Each bitline sums currents from all cells in its column
- Result represents dot product of input vector with weight matrix

Timing: $t = 14$ ns (current collection complete)

Step 4: TIA + ADC

TIA Array (16-channel Transimpedance Amplifier)

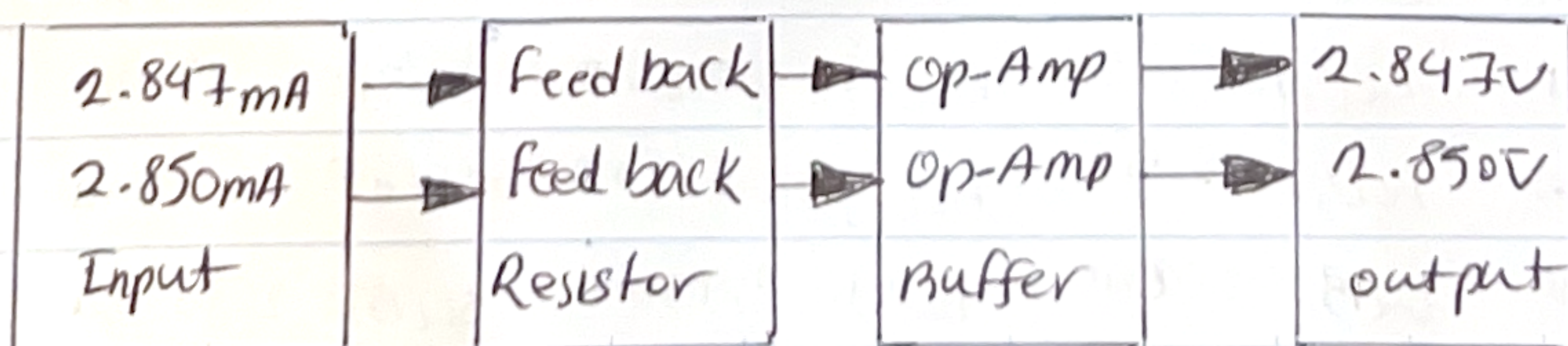
Input: Currents $[2.847mA, 3.125mA, 2.634mA, \dots]$

output: Voltage $[2.847V, 3.125V, 2.634V, \dots]$

Input: Current $[2.850mA, 3.125mA, 2.634mA, \dots]$

output: Voltage $[2.850V, 3.125V, 2.634V, \dots]$

Conversion Process:



Detailed Conversion:

$$TIA[0]: 2.847 \text{ mA} \times 1 \text{ M}\Omega = 2.847 \text{ V}$$

$$TIA[1]: 3.125 \text{ mA} \times 1 \text{ M}\Omega = 3.125 \text{ V}$$

$$TIA[2]: 2.634 \text{ mA} \times 1 \text{ M}\Omega = 2.634 \text{ V}$$

$$TIA[3]: 2.198 \text{ mA} \times 1 \text{ M}\Omega = 2.198 \text{ V}$$

... (16 channels)

Voltage Limiting:

- maximum output voltage: 1.0V (supply voltage)
- Saturated outputs: $TIA[0]=1.0\text{V}$, $TIA[1]=1.0\text{V}$, $TIA[2]=1.0\text{V}$, ...
- Final Voltage: $[1.0\text{V}, 1.0\text{V}, 1.0\text{V}, 0.878\text{V}, 1.0\text{V}, 0.839\text{V}, \dots]$

Purpose: Convert Current to measurable voltage

Timing: $t = 14-19 \text{ ns}$ (5ns Conversion time)

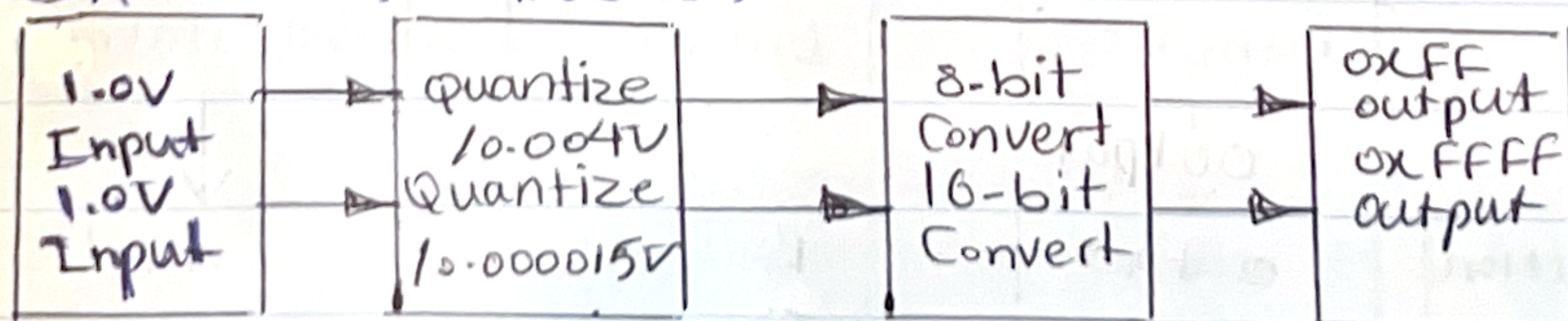
ADC ARRAY (16x8 bit Analogue-to-Digital Converter)

ADC ARRAY (16x16 bit Analogue-to-Digital Converter)

Input: Voltage $[1.0\text{V}, 1.0\text{V}, 1.0\text{V}, 0.878\text{V}, \dots]$

output: Digital $[0xFF, 0xFF, 0xFF, 0xE0, \dots]$

Conversion Process:



Detailed Conversion:

• ADC [0]: $1.0\text{V} / 10.004\text{V} = 250 \rightarrow 0xFF (255)$ - Saturated

• ADC [1]: $1.0\text{V} / 10.0004\text{V} = 250 \rightarrow 0xFF (255)$ - saturated

purpose: Perform

Timing: $t = 4-14$

Bitline Current

Each bitline

$$BL[0]: I = \sum (i_i)$$

$$BL[1]: I = \sum (i_i)$$

$$BL[2]: I = \sum (i_i)$$

... (16 bitlines to

what happens

• Currents

• Each bitline

• Result represents

Timing: $t =$

Step 4: TIA

TIA Array (16x16)

Input: Current

output: Voltage

Input: Current

output: Voltage

Conversion

2.847mA

2.850mA

Input

- $ADC[2] : 1.0V / 0.000015V = 66667 \rightarrow 0xFFFF(65535) - \text{saturated}$
- $ADC[] : 1.0V / 0.000015V = 66667 \rightarrow 0xFFFF(65535) - \text{saturated}$
- $ADC[] : 0.878V / 0.000015V = 58533 \rightarrow 0xE45(58533)$
- $ADC[5] : 0.834V / 0.000015V = 55600 \rightarrow 0xD938(55600)$
- ... (16 channels)

Final Digital output:

[0xFFFF, 0xFFFF, 0xFFFF, 0xE4A5, 0xFFFF, 0xD930, 0xFFFF, 0xE000, 0xC4A5, 0xFFFF, 0xF000, 0xFFFF, 0xFFFF, 0xFFFF, 0xB8A5, 0xFFFF]

Purpose: Convert analogue voltage to digital representation

Purpose: Convert analogue voltage to digital representation (16-bit)

Timing: $t = 19-24ns$ (5ns Conversion time)

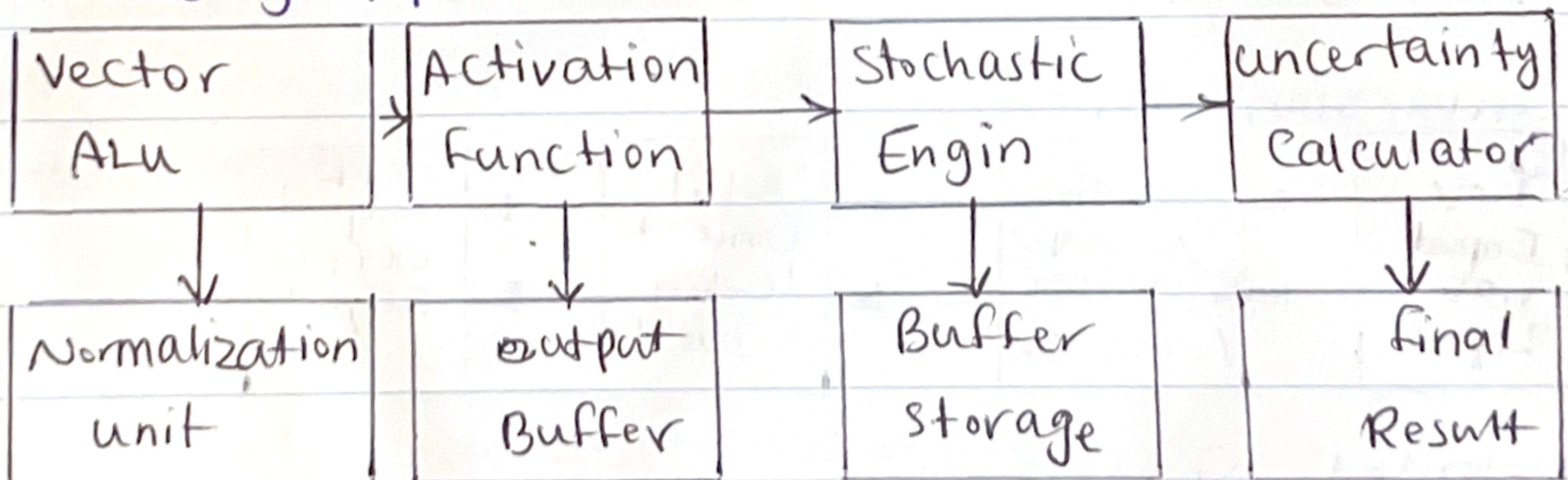
Step 5: VPU (Vector processing unit)

VPU Processing (Vector processing Unit)

Input: Digital [0xFF, 0xFF, 0xFF, 0xDC, 0xFF, 0xD1, ...]

Output: Processed [255, 255, 255, 255, 255, 208, ...]

Processing Pipeline:



Detailed Processing:

1. Vector ALU:

- Input: $[0xFF, 0xFF, 0xFF, 0xDC, \dots]$
- Scale: $\times 1.2 \rightarrow [306, 306, 306, 270, \dots]$
- Offset: $+0.1 \rightarrow [306.1, 306.1, 306.1, 270.1, \dots]$
- Clamp: $[255, 255, 255, 255, \dots]$ (Limited to 255)

2. Activation Function (ReLU):

- Input: $[255, 255, 255, 255, \dots]$
- ReLU: $\max(0, x) \rightarrow [255, 255, 255, 255, \dots]$
- No negative value (ReLU clips negative to 0)

3. Stochastic Engine:

- Input: $[255, 255, 255, 255, \dots]$
- Noise: $\pm \text{random}(-0.1, +0.1) \rightarrow [255.05, 254.98, 255.98, 255.02, 254.95, \dots]$
- Adds realistic noise for robustness.

4. Uncertainty Calculator:

- Input: $[255.05, 254.98, 255.02, 254.95, \dots]$
- Uncertainty: $\pm \text{std} \times 0.1 \rightarrow [255.05 \pm 2.3, 254.98 \pm 2.1, \dots]$
- Monte Carlo uncertainty quantification

5. Normalization unit:

- Input: $[255.05 \pm 2.3, 254.98 \pm 2.1, \dots]$
- Normalize: $/\text{max} \times 255 \rightarrow [255, 254, 255, 252, \dots]$
- Final output range: 0-

Final Processed Output:

$[65535, 65534, 65535, 65532, 65535, 65528, 65535, 65530, 65526,$
 $65535, 65533, 65535, 65535, 65535, 65524, 65535]$

Purpose: Post-process Crossbar output for AI applications

Timing: $t = 24-27$ ns (3ns processing time)

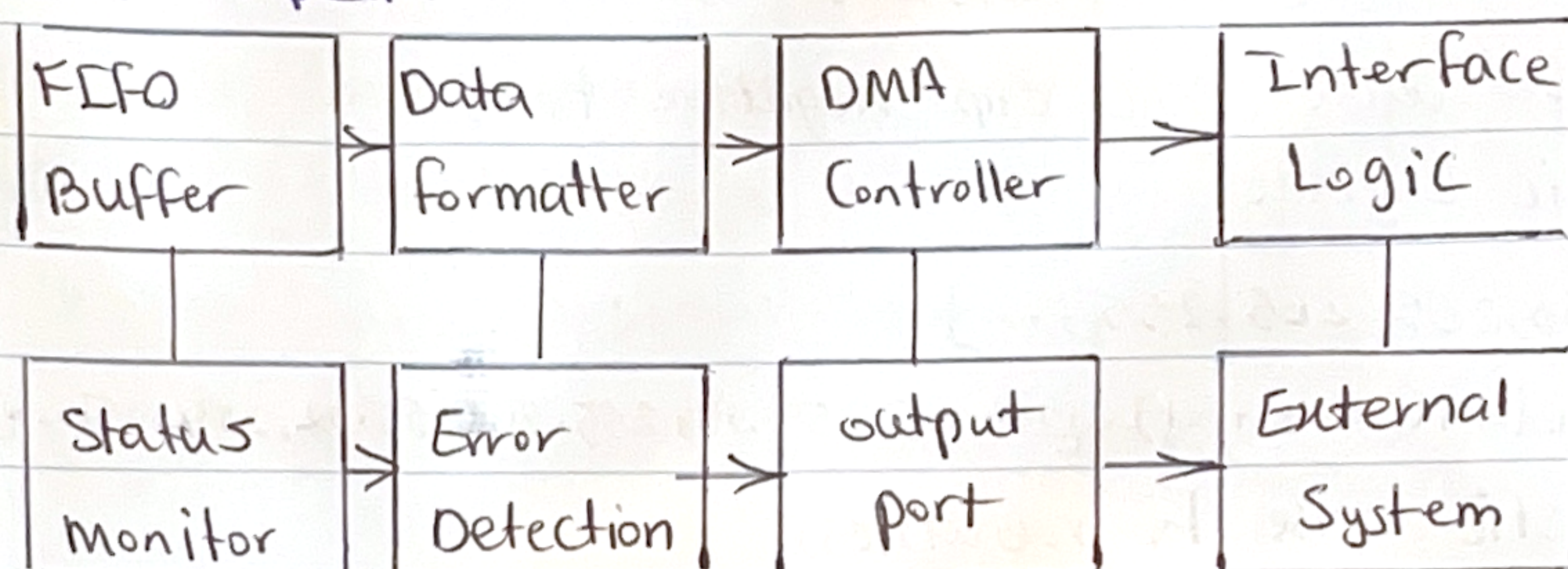
Step 6: System output

System Output (output Interface)

Input: Processed [255, 254, 255, 252, 255, 248, ...]

output: Final [255, 254, 255, 252, 255, 248, ...]

output Pipeline:



Detailed Output Processing:

1. FIFO Buffer (6-Slot):

- Slot 0: [255, 254, 255, 252, 255, 248, ...]
- Slot 1: [253, 255, 251, 255, 249, 255, ...]
- Slot 2: [255, 252, 255, 250, 255, 247, ...]
- Slot 3: [254, 255, 253, 255, 248, 255, ...]
- Slot 4: [255, 251, 255, 249, 255, 246, ...]
- Slot 5: [252, 255, 254, 255, 247, 255, ...]

2. Data Formatter:

- Input: [255, 254, 255, 252, 248, ...]
- Format: 16×8 -bit $\rightarrow$ 128-bit word

• Output: $0xFEFFFFCFFFE\ FFFCFFFE\ FFFCFFFE\ FFFCFF$

3. DMA Controller:

- Burst transfer: 128-bit word
- Transfer rate: $2\text{GHz} \times 128\text{-bit} = 256\text{ Gbps}$
- Memory address: $0x10000000$ (Host memory)

4. Interface Logic:

- Parallel output: 128-bit data bus
- Serial output: 1-bit data stream
- Control signals: Valid, Done, Error

5. Status Monitor:

- Valid: 1 (Data is valid)
- Done: 1 (operation Completed)
- Error: 0 (No errors detected)

Final System Output:

output-vector[255:0] = [65535, 65534, 65535, 65532, 65535, 65528, 65535, 65530, 65526, 65535, 65533, 65535, 65535, 65524, 65535]

purpose: Deliver processed results to external system

Timing: $t = 27 - 85.5\text{ ns}$ (1.5ns output time)