Reconfigurable 8x4 Virtualized Fabrication Design

Table of Contents

Design Explanation:

The provided design implements an 8x4 Computational Unit (CU) fabric using virtualization with a 4x4 physical fabric. The design consists of a top-level module that interconnects multiple CUs in a grid-like structure, with each CU capable of performing various operations on 4-bit data inputs.

Key Components:

1. **Computational Units (CUs)**: Each CU is a 4-bit processing element that can perform various operations based on the selected operation code.
2. **Multiplexers**: 4-to-1 multiplexers (MUX4X1) are used to select input sources for each CU.
3. **Registers**: Several types of registers are used to store data (REG_D):
    - Input registers for A and B
    - Operation code registers
    - Output registers

Interconnection Structure:

- The first row can receive data from external inputs or the last row's output.
- CUs in columns 1-3 can receive data from horizontal connections to the first column of their respective rows.
- From row 1 onwards, CUs can also receive data from the previous row in the same column.
- Rows 2 and 3 have additional multi-level connections from rows 0 and 1, respectively.

Data Flow:

- Input data and operations are first sent to storage units (registers).
- Multiplexers select the appropriate input sources for each CU based on control signals (SELMA and SELMB).
- The selected inputs are processed by the CUs according to the stored operation codes.
- Results are stored in output registers and can be used as inputs for subsequent operations or as final outputs.

Virtualization:

The design implements an 8x4 logical structure using a 4x4 physical fabric. This is achieved through the interconnection scheme and the ability to feed outputs back as inputs, allowing for multiple passes through the fabric to complete larger computations. This architecture allows for flexible and efficient computation, enabling complex operations to be broken down and executed across multiple CUs and potentially multiple passes through the fabric

Subcomponents

1. Computational Unit (CU_6)

Description

The CU_6 entity is the core processing element. It takes two 4-bit inputs (A and B) and a 5-bit select signal (SEL) as inputs. Based on the SEL value, it performs one of several operations and outputs the result on the 4-bit output port (O).

Input/Output Ports

Port	Direction	Width	Description
A	Input	dw (Generic, default 4)	First input operand
B	Input	dw (Generic, default 4)	Second input operand
O	Output	dw (Generic, default 4)	Result of the selected operation
SEL	Input	dws (Generic, default 5)	Operation select code

Operations

SEL (Binary)	Operation	Description
00000	NO OP	Output is zero
00001	A AND B	Bitwise AND
00010	A OR B	Bitwise OR
00011	A NAND B	Bitwise NAND
00100	A NOR B	Bitwise NOR
00101	A XOR B	Bitwise XOR
00110	A XNOR B	Bitwise XNOR
00111	A + B	Addition
01000	A - B	Subtraction
01001	A * B	Multiplication
01010	A > B	Greater Than (Output is all '1's if true, all '0's if false)
01011	A < B	Less Than (Output is all '1's if true, all '0's if false)
01100	A = B	Equal To (Output is all '1's if true, all '0's if false)
01101	A >= B	Greater Than or Equal To (Output is all '1's if true, all '0's if false)
01110	A <= B	Less Than or Equal To (Output is all '1's if true, all '0's if false)
01111	A /= B	Not Equal To (Output is all '1's if true, all '0's if false)
10000	A SLA B	Arithmetic Shift Left by B
10001	A SRA B	Arithmetic Shift Right by B
10010	A ROL B	Rotate Left by B
10011	A ROR B	Rotate Right by B
10100	A SLL B	Logical Shift Left by B
10101	A SRL B	Logical Shift Right by B
10110	A	Pass A (Output is A)
10111	B	Pass B (Output is B)
Others	'Z'	High Impedance

2. Register (REG_D)

Description

The REG_D entity implements a D-type register with a write enable. It stores a dw-bit value. When W_EN is '0' on a rising clock edge, the input D is loaded into the register. When 'W_EN' is '1' on a rising clock edge, the currently stored value is output on Q.

Input/Output Ports

Port	Direction	Width	Description
D	Input	dw (Generic, default 4)	Data input
W_EN	Input	1	Write enable signal (0: Write, 1: Read)
CLK	Input	1	Clock input
Q	Output	dw (Generic, default 4)	Data output

3. Multiplexer (MUX4X1)

Description

The MUX4X1 entity is a 4-to-1 multiplexer. It selects one of four dw-bit inputs (A, B, C, D) based on the 2-bit select input Sel.

Input/Output Ports

Port	Direction	Width	Description
A	Input	dw (Generic, default 4)	First input
B	Input	dw (Generic, default 4)	Second input
C	Input	dw (Generic, default 4)	Third input
D	Input	dw (Generic, default 4)	Fourth input
Sel	Input	dwm (Generic, default 2)	Select input
Z	Output	dw (Generic, default 4)	Output (selected input)

Selection Table

Sel (Binary)	Output (Z)
00	A
01	B
10	C
11	D
Others	'Z' (High Impedance)

Port Declaration

Input/Output Ports

Port	Direction	Width	Description
A	Input	ARR_1D(0 to COLS-1) (DW-1 downto 0)	Array of first input operands for the first row
B	Input	ARR_1D(0 to COLS-1) (DW-1 downto 0)	Array of second input operands for the first row
Y	Output	ARRAY_2D(0 to ROWS-1,0 to COLS-1) (DW-1 downto 0)	Array of CU outputs
CLK	Input	1	Clock input
W_ENA	Input	ARR_EN2D(0 to ROWS-1,0 to COLS-1)	2D array of write enable signals for A registers
W_ENB	Input	ARR_EN2D(0 to ROWS-1,0 to COLS-1)	2D array of write enable signals for B registers
W_ENOP	Input	ARR_EN2D(0 to ROWS-1,0 to COLS-1)	2D array of write enable signals for Operation registers
W_ENY	Input	ARR_EN2D(0 to ROWS-1,0 to COLS-1)	2D array of write enable signals for Output registers
SELMA	Input	ARRAY_2D(0 to ROWS-1,0 to COLS-1)(DWM-1 downto 0)	2D array of select signals for MUXA
SELMB	Input	ARRAY_2D(0 to ROWS-1,0 to COLS-1)(DWM-1 downto 0)	2D array of select signals for MUXB
SELCU	Input	ARRAY_2D(0 to ROWS-1,0 to COLS-1)(DWS-1 downto 0)	2D array of operation select codes for CUs

Interconnections

1. **Input Data:** The inputs A and B provide external data to the *first row* of CUs.
2. **Registers:** Each CU has registers for its A input, B input, the operation code (SEL), and the output.
3. **Multiplexers:** Each CU has two 4-to-1 multiplexers (MUXA and MUXB) to select the data sources for its A and B inputs. The selection is controlled by SELMA and SELMB respectively.
4. **Data Flow:**
    - **Row 0:** Can receive external input.
    - **Rows 1-3:** Can receive inputs from the CUs in the previous row or from CUs in the same row, column 0.
    - **Rows 2 & 3:** Additional connections to rows 0 and 1, respectively.
    - **Feedback:** to the first row, allowing for iterative computations. The output of the last row (REGYo(3,*)) is fed back as a potential input

5. **Output:** The outputs of the CUs are stored in registers (REGYo) and made available on the Y output port.

CU(Row, Col)	MUXA Inputs (Based on SELMA)	MUXB Inputs (Based on SELMB)
CU(0,0)	A(0), REGYo(3,0), "0000", "0000"	B(0), REGYo(3,0), "0000", "0000"
CU(0,1)	A(1), REGYo(0,0), REGYo(3,1), "0000"	B(1), REGYo(0,0), REGYo(3,1), "0000"
CU(1,0)	REGYo(0,0), "0000", "0000", "0000"	REGYo(0,0), "0000", "0000", "0000"
CU(2,1)	REGYo(0,1), REGYo(1,1), REGYo(2,0), "0000"	REGYo(0,1), REGYo(1,1), REGYo(2,0), "0000"

Important Considerations:

- **custom_pack:** The line use work.custom_pack.all; in TopModule suggests that custom package named custom_pack defined in the work library, which probably contains the declarations for ARR_1D, ARRAY_2D, and ARR_EN2D.
- **Write Enables:** The W_ENA, W_ENB, W_ENOP, and W_ENY signals control when the data is written into the registers. These signals are essential for controlling the data flow and the execution of computations.

Overall Design:

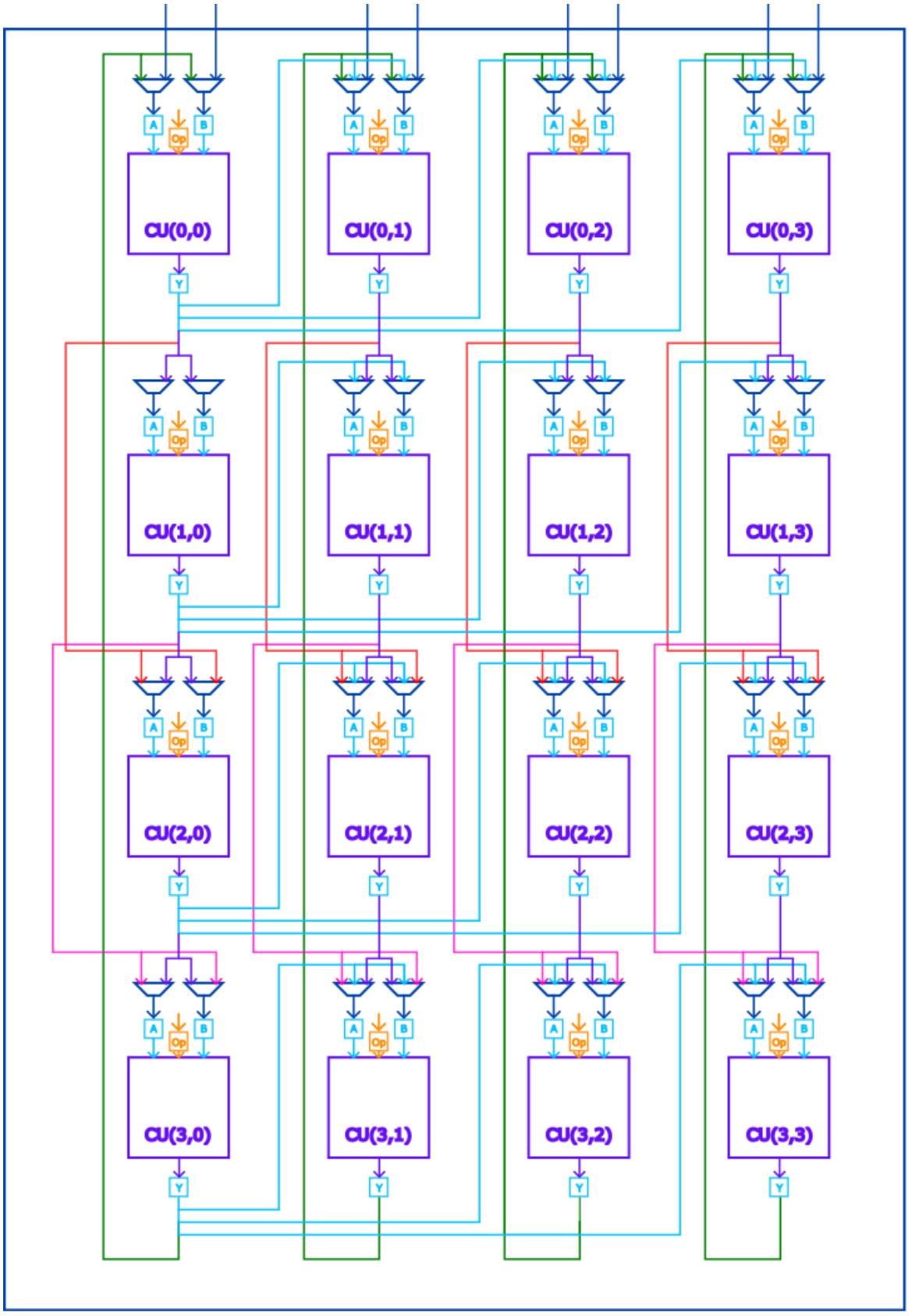


Figure 1 Overall Design

Test Cases:

RUN - 1

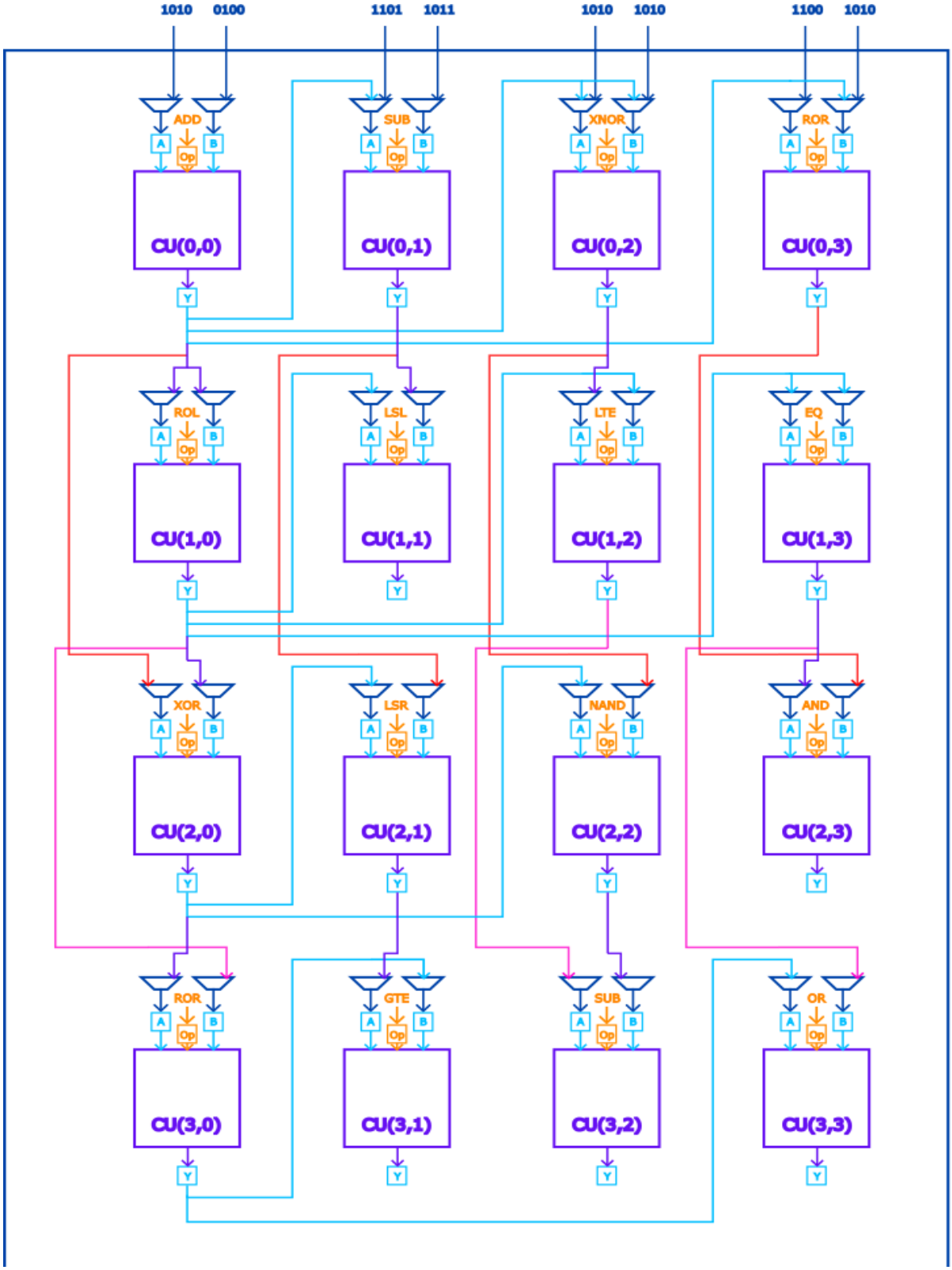


Figure 2 Run 1.

Figure *Error! Bookmark not defined.* ***Run 1.***

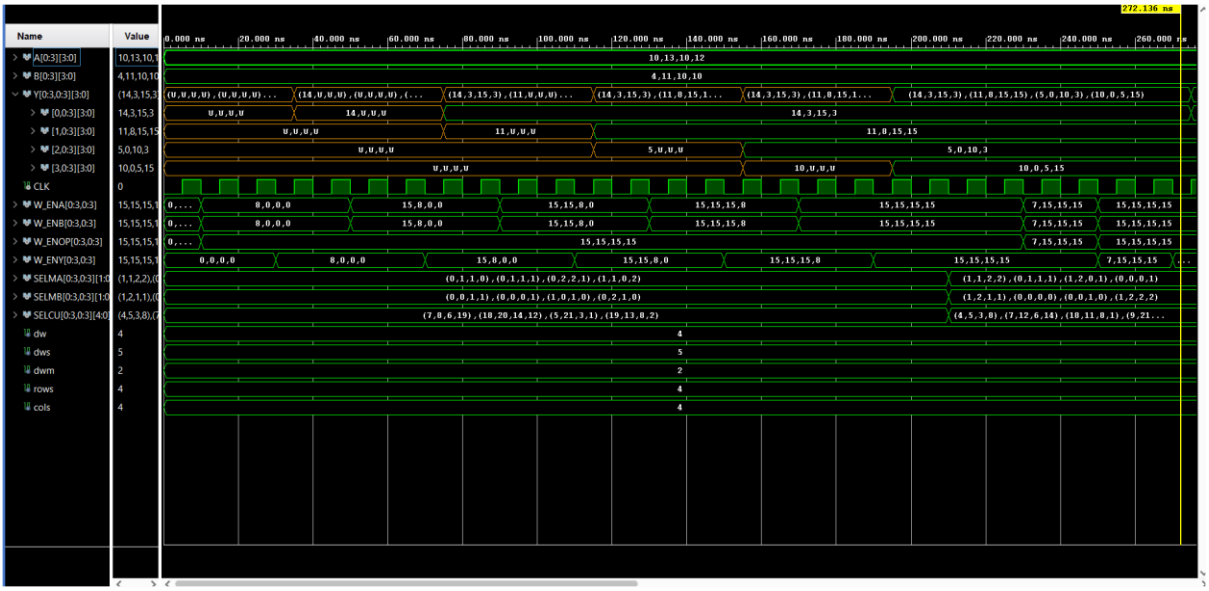
External Data Inputs for Run - 1:

CU#	A	B
CU(0,0)	1010	0100
CU(0,1)	1101	1011
CU(0,2)	1010	1010
CU(0,3)	1100	1010

Connectivity and Operations for Run - 1:

CU#	SourceA	SourceB	Oper
CU(0,0)	External	External	ADD
CU(0,1)	CU(0,0)	External	SUB
CU(0,2)	CU(0,0)	CU(0,0)	XNOR
CU(0,3)	External	CU(0,0)	ROR
CU(1,0)	CU(0,0)	CU(0,0)	ROL
CU(1,1)	CU(1,0)	CU(0,1)	LSL
CU(1,2)	CU(1,0)	CU(0,2)	LTE
CU(1,3)	CU(1,0)	CU(1,0)	EQ
CU(2,0)	CU(0,0)	CU(1,0)	XOR
CU(2,1)	CU(2,0)	CU(0,1)	LSR
CU(2,2)	CU(2,0)	CU(1,2)	NAND
CU(2,3)	CU(1,3)	CU(06,3)	AND
CU(3,0)	CU(2,0)	CU(1,0)	ROR
CU(3,1)	CU(2,1)	CU(3,0)	GTE
CU(3,2)	CU(1,2)	CU(2,2)	SUB
CU(3,3)	CU(3,0)	CU(1,3)	OR

Waveform for Run - 1:



RUN – 2

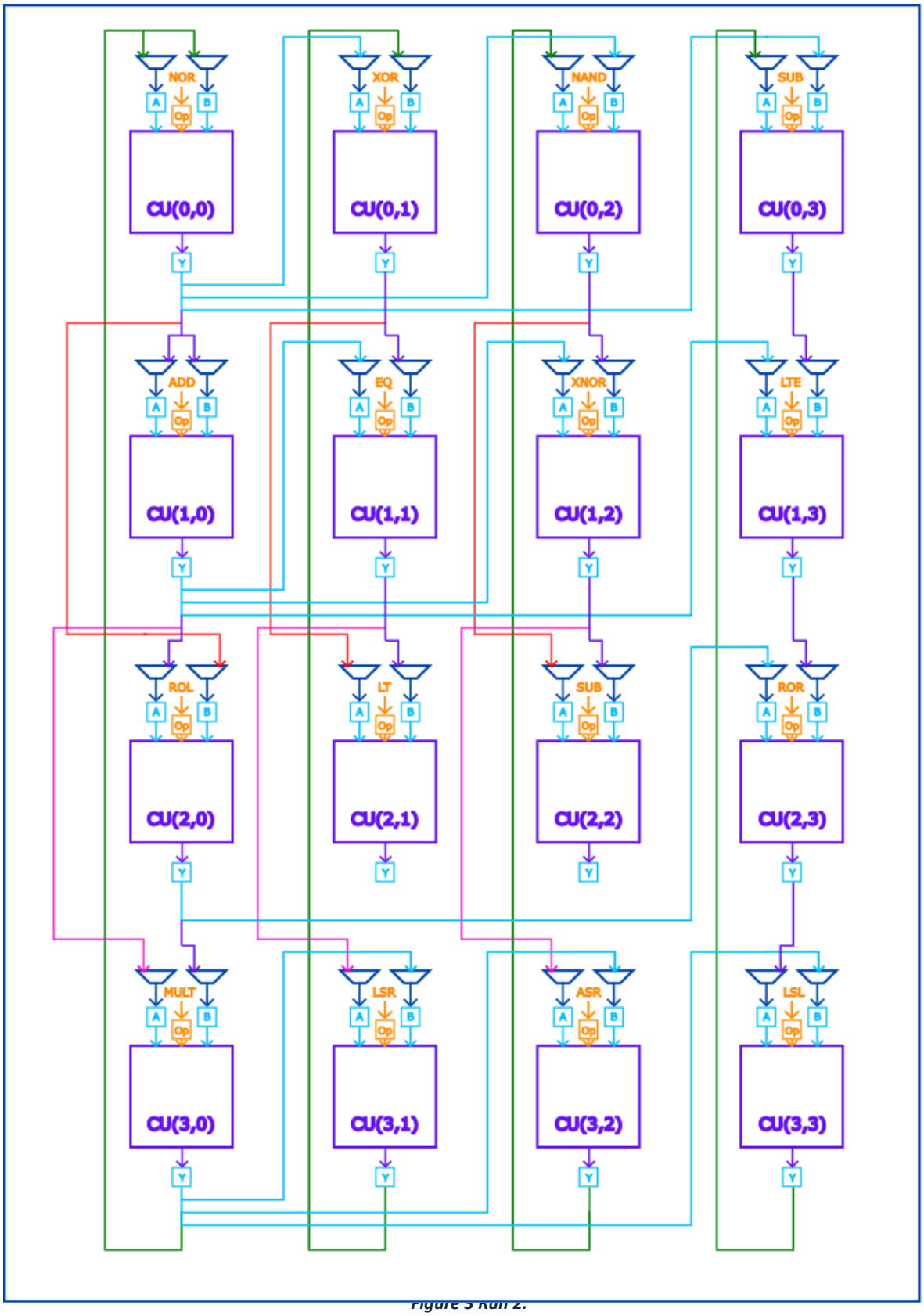


Figure 3 Run 2.

Figure 3 Run 2.

Connectivity and Operations for Run-2**:**

CU#	y	SourceB	Oper
CU(0,0)	CU(3,0)	CU(3,0)	NOR
CU(0,1)	CU(0,0)	CU(3,1)	XOR
CU(0,2)	CU(3,2)	CU(0,0)	NAND
CU(0,3)	CU(3,3)	CU(0,0)	SUB
CU(1,0)	CU(0,0)	CU(0,0)	ADD
CU(1,1)	CU(1,0)	CU(0,1)	EQ
CU(1,2)	CU(1,0)	CU(0,2)	XNOR
CU(1,3)	CU(1,0)	CU(0,3)	LTE
CU(2,0)	CU(1,0)	CU(0,0)	ROL
CU(2,1)	CU(2,0)	CU(0,1)	LT
CU(2,2)	CU(0,2)	CU(1,2)	SUB
CU(2,3)	CU(1,3)	CU(0,3)	AND
CU(3,0)	CU(1,0)	CU(2,0)	MULT
CU(3,1)	CU(1,1)	CU(3,0)	LSR
CU(3,2)	CU(1,2)	CU(3,0)	ASR
CU(3,3)	CU(2,3)	CU(3,0)	LSL

Waveform for Run – 2: