Signal Processing Using Digital Technology

Jeremy Barsten

Jeremy Stockwell

May 6, 2003

Advisors:

Dr. Thomas Stewart

Dr. Vinod Prasad

Digital Signal Processor

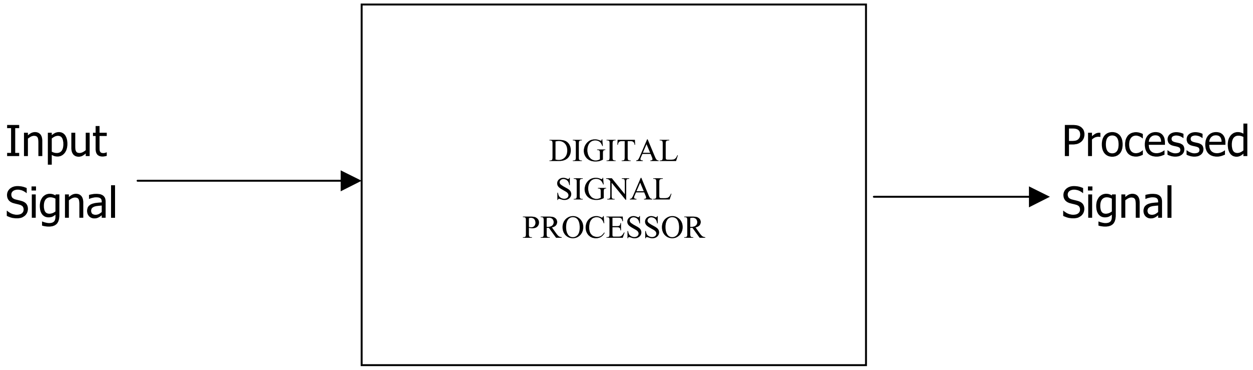
- Project Description
- Design and Simulation of VLSI Processor
- Design and Simulation of the VHDL Processor Implemented on the Xilinx FPGA Board
- FPGA Problems.

Project Description

- All purpose digital signal processor using FPGA/VHDL and ASIC/VLSI technology.
- Useable for a variety of applications:
  - Audio and Video
  - Cellular Technology
- Adapted depending on the application.

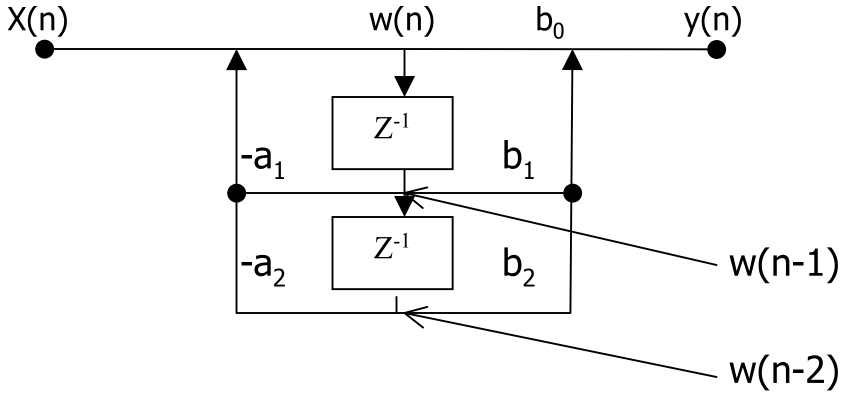
Project Description

- High-level Block Diagram:



Filter Design

- Manipulating a digital input utilizing multipliers and adders.
- Direct Form II realization of an IIR Filter:



$W(n)=X(n)-a_1W(n-1)-a_2W(n-2)$ $Y(n)=b_0W(n)+b_1W(n-1)+b_2W(n-2)$

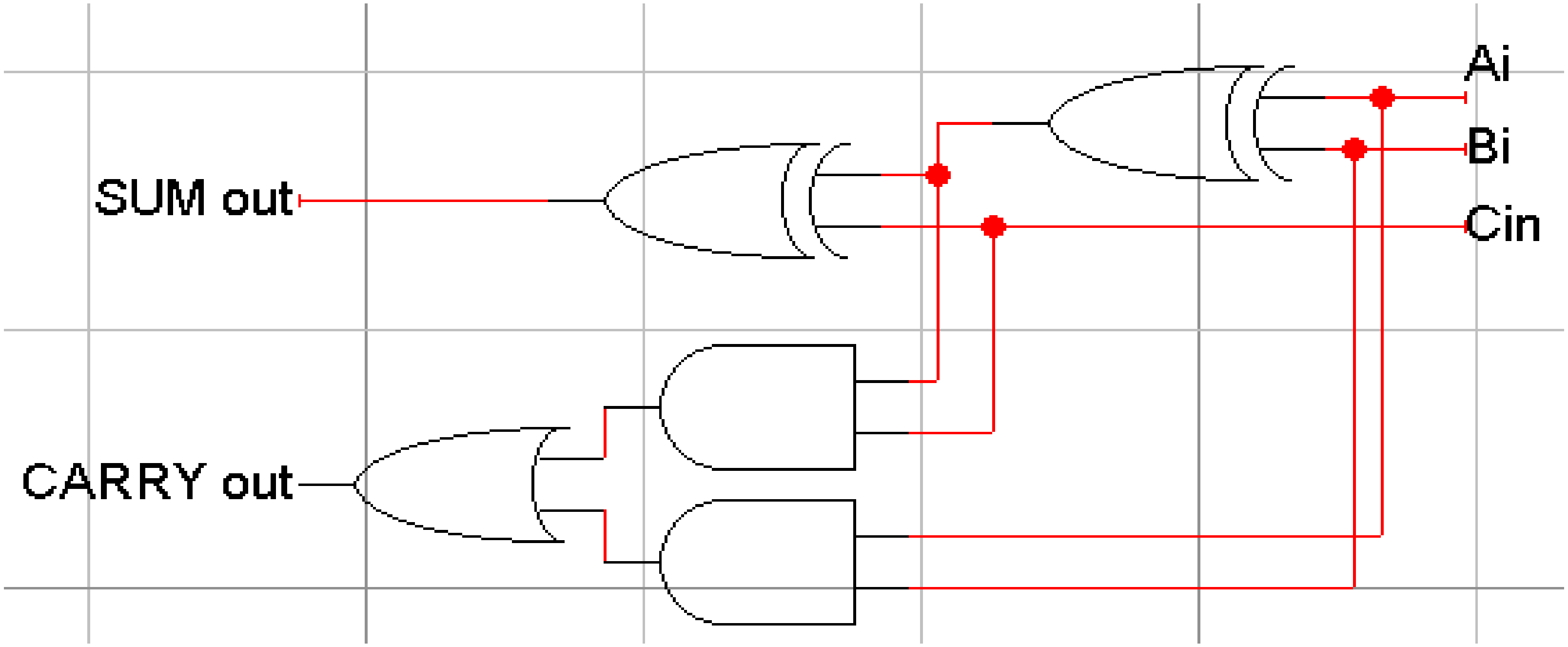
Signal Converters

- Each signal will be analog in nature.
- Requires an analog-to-digital converter at the input stage and a digital-to-analog converter at the output stage.
- Tried to use the 8-bit A/D and D/A converters that were part of the Xilinx FPGA Version II Board.

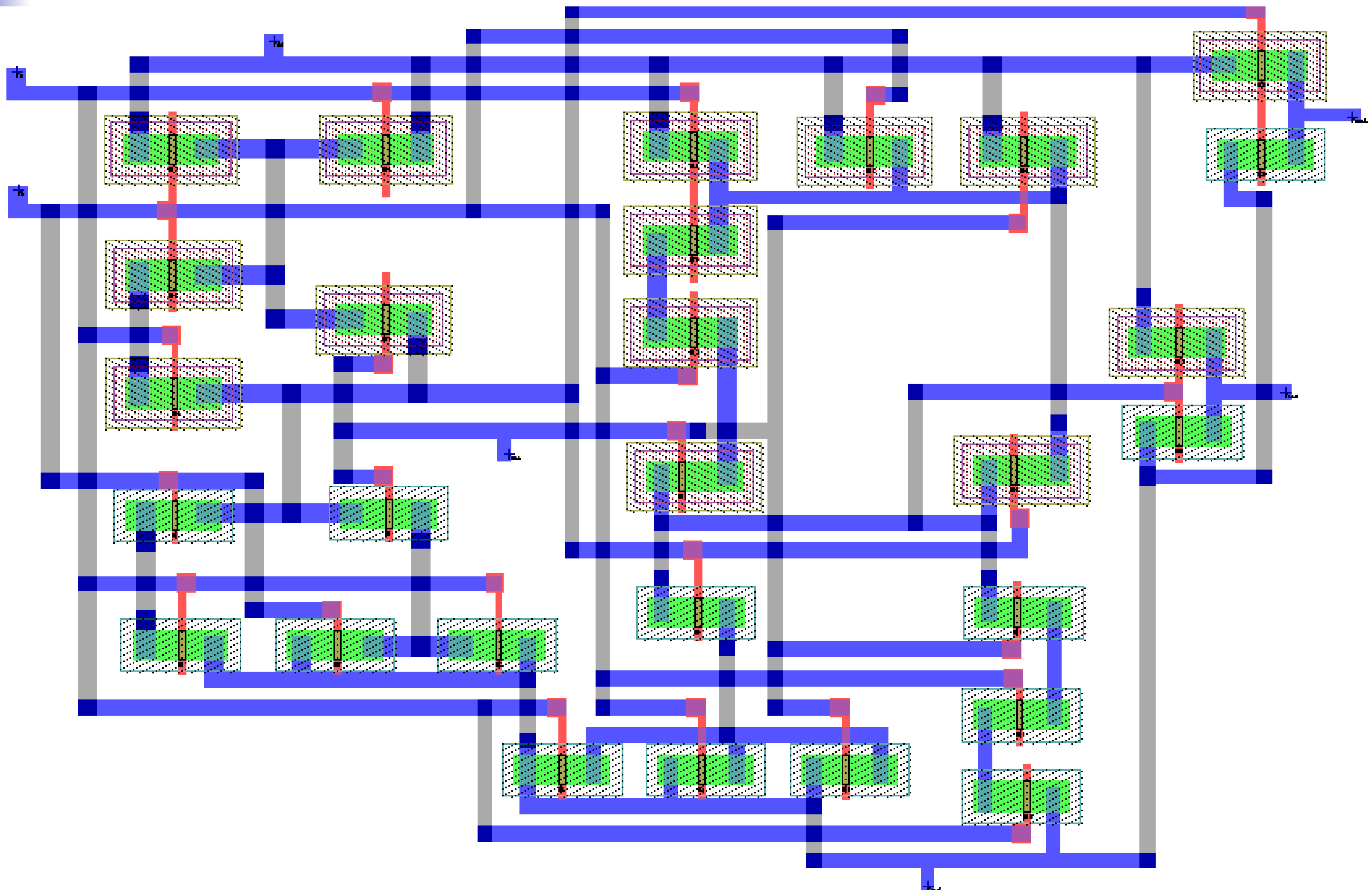
Adder and Multiplier

- Any basic signal processor consists of different stages of addition and multiplication.
- A n-bit by n-bit multiplication will take place and result in a 2*n-bit value.
- This answer will be added to previous values stored in a data register (discussed later).

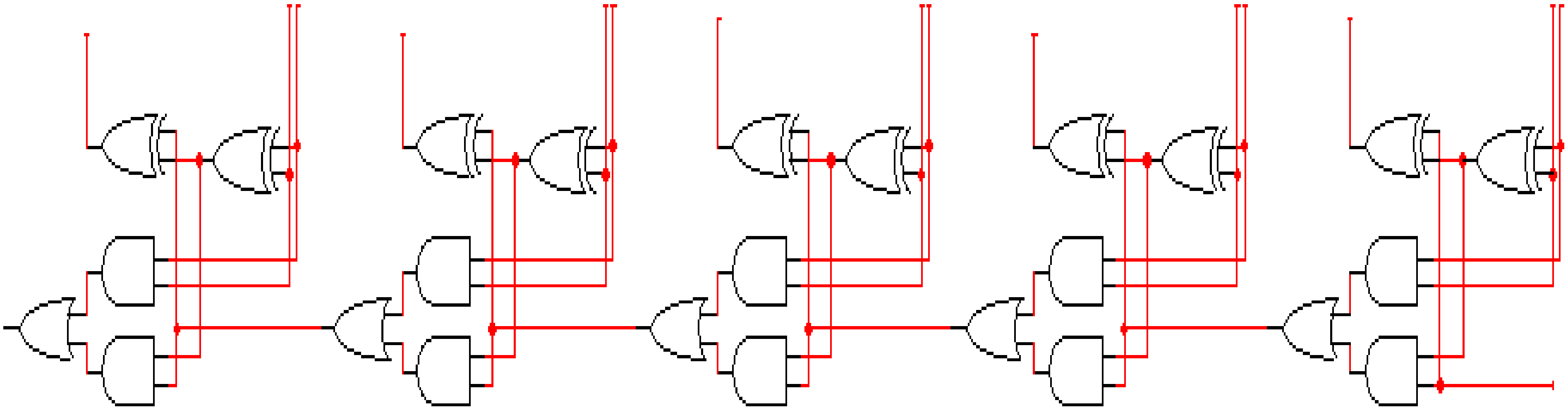
Adder Cell



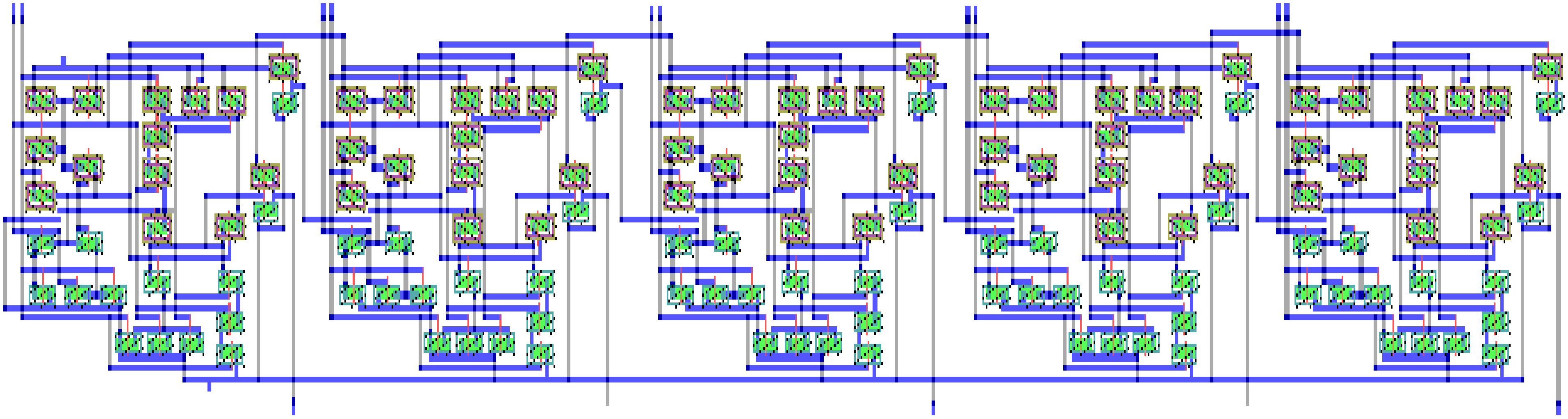
Adder Cell



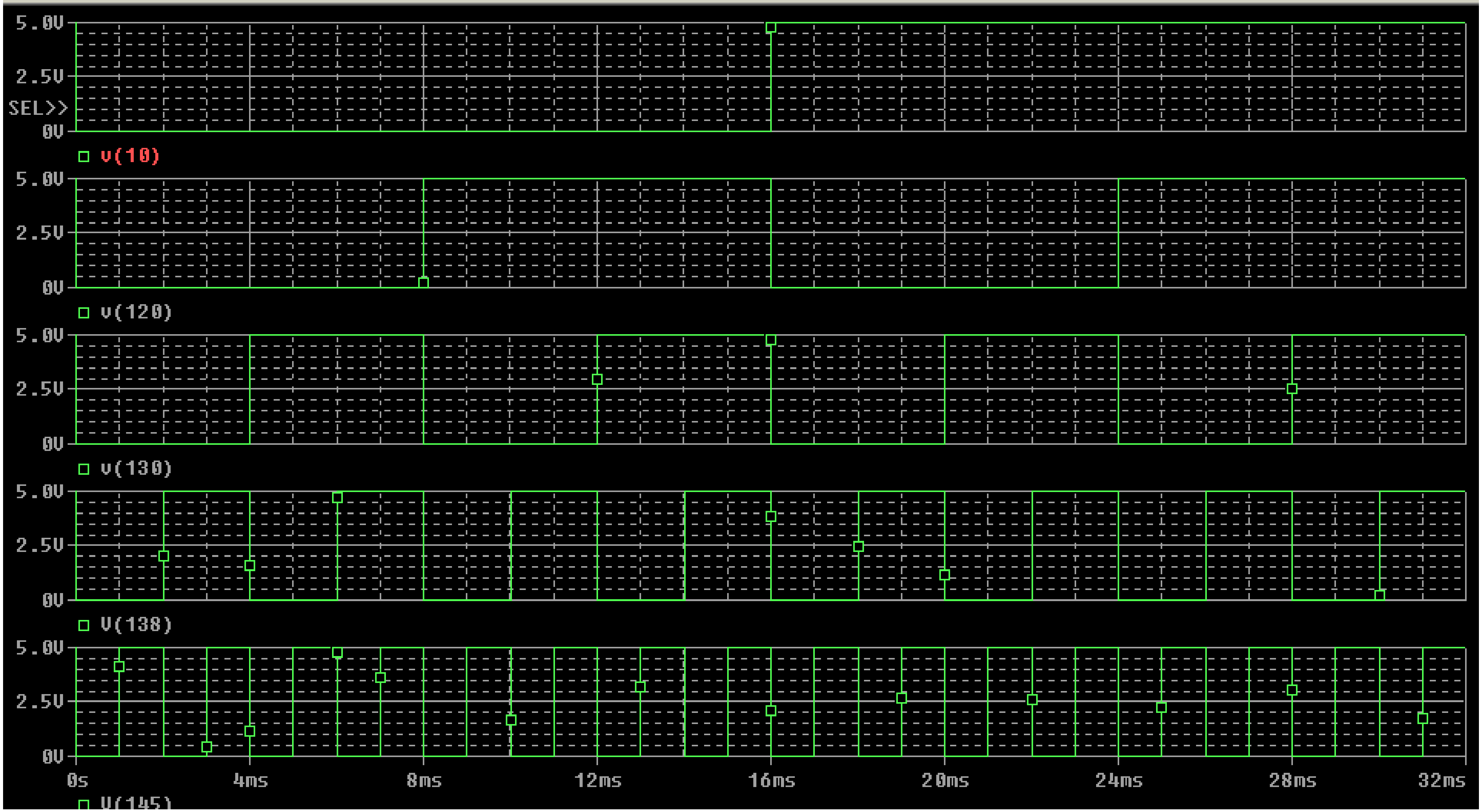
5-Bit Ripple-Carry Adder - Logical Design



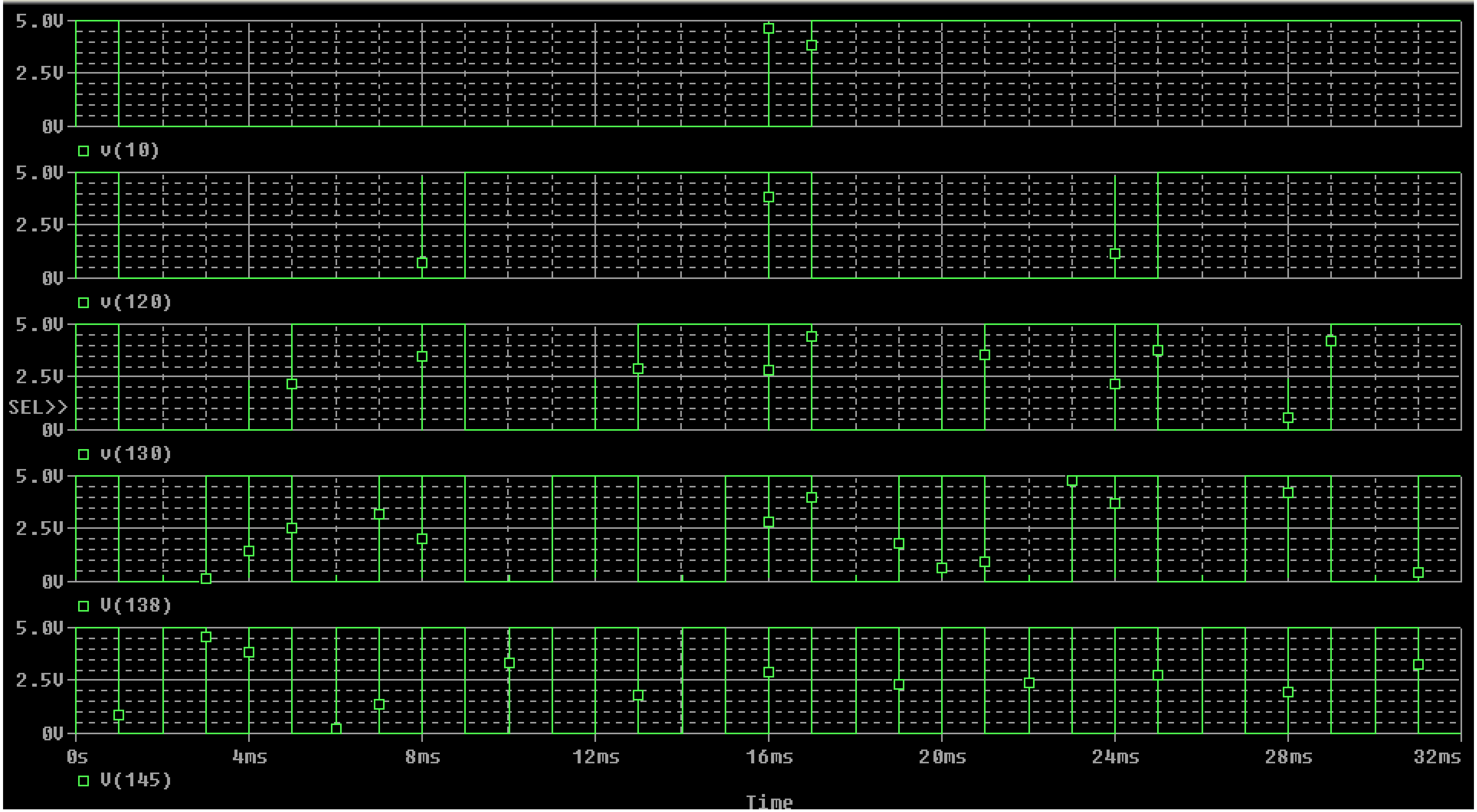
5-Bit Ripple Carry Adder - VLSI Design



5-Bit Adder Simulation – Added to 0

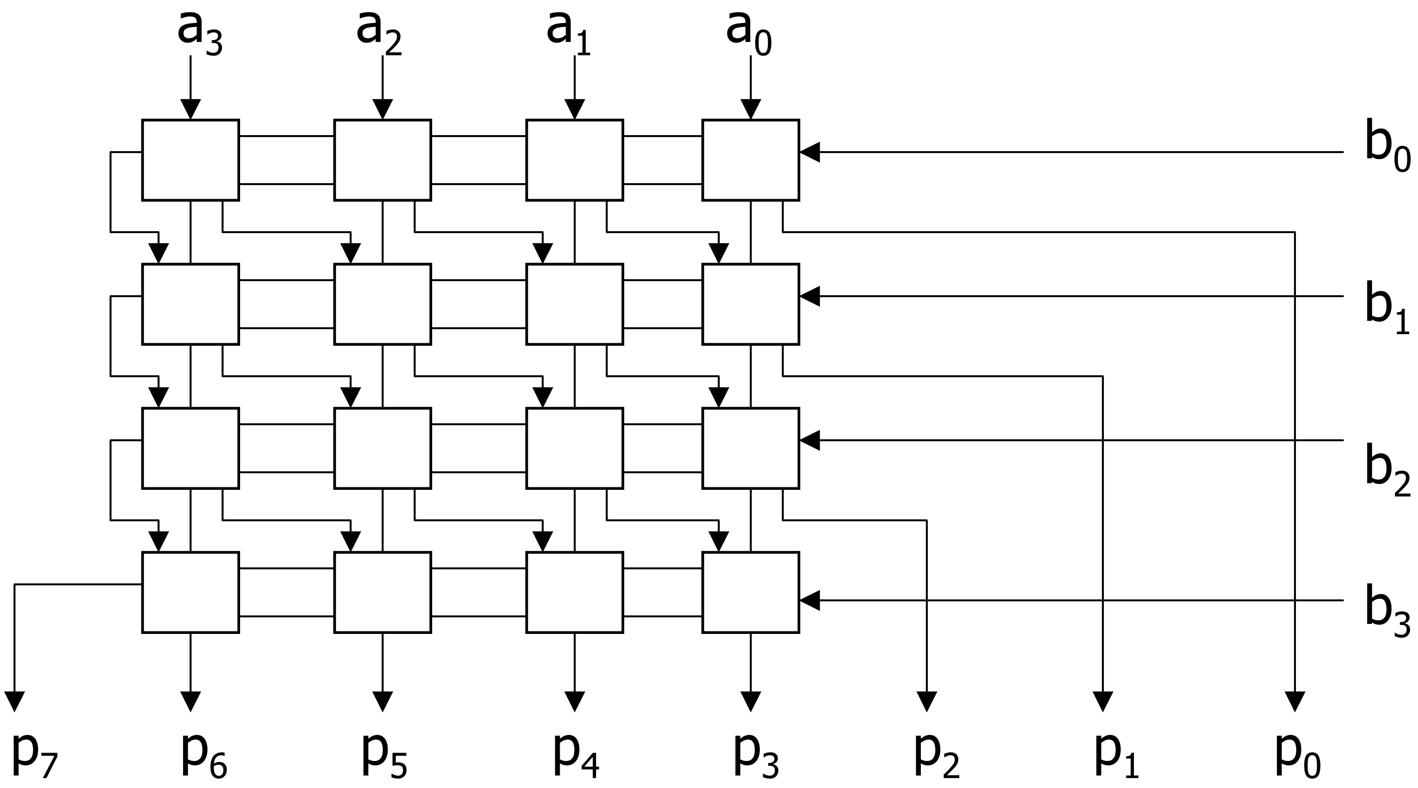


5-Bit Adder Simulation – Added to -1

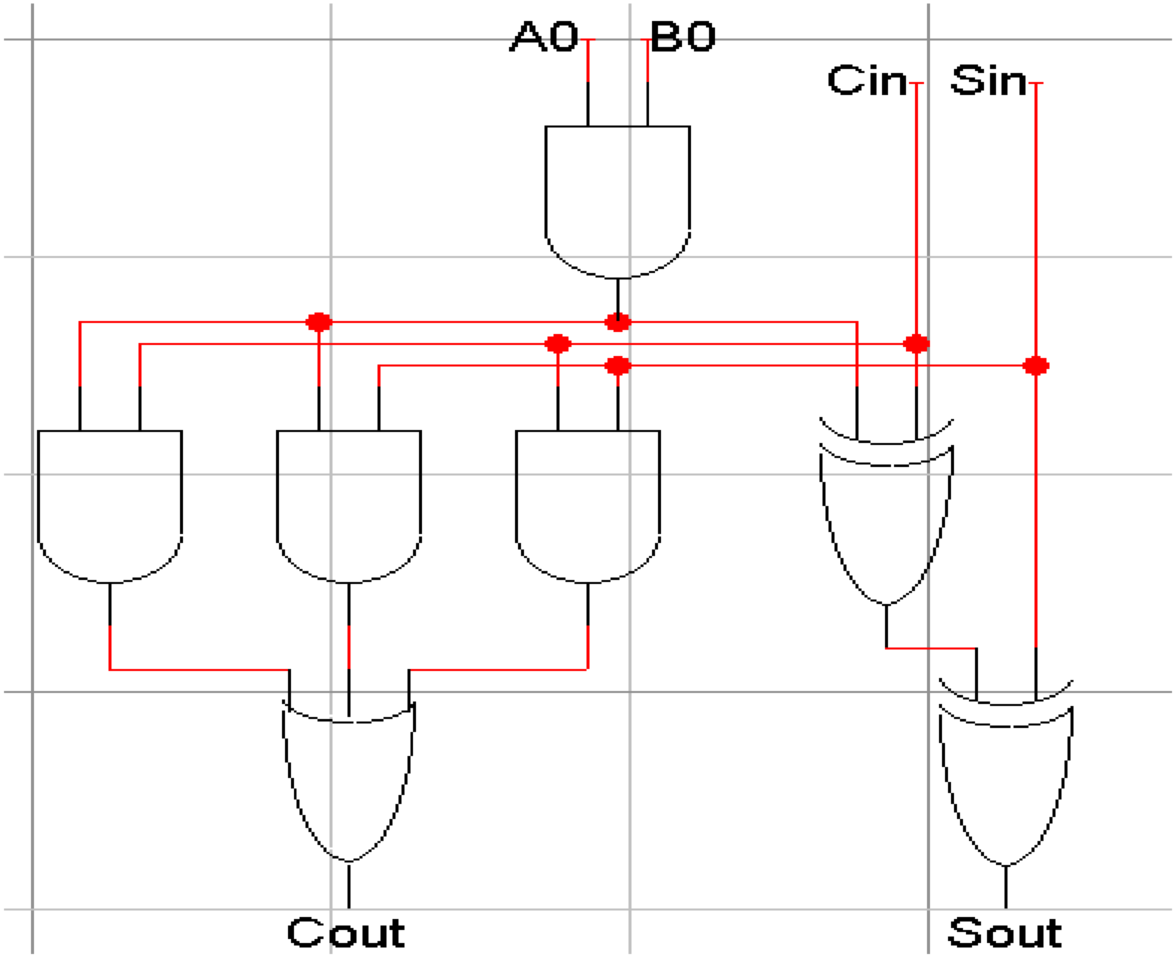


Cellular Multiplication

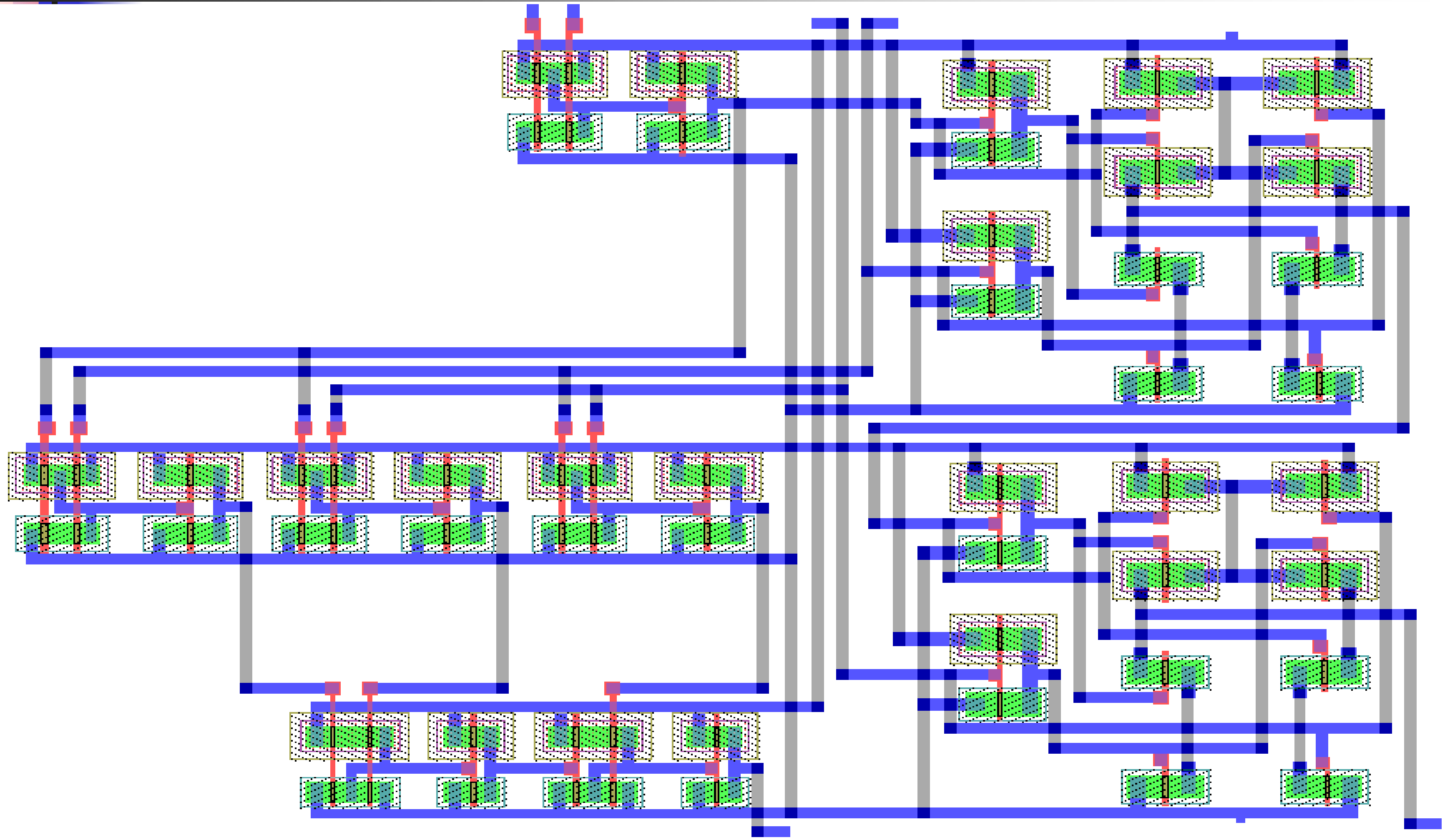
- Cellular Multiplication:



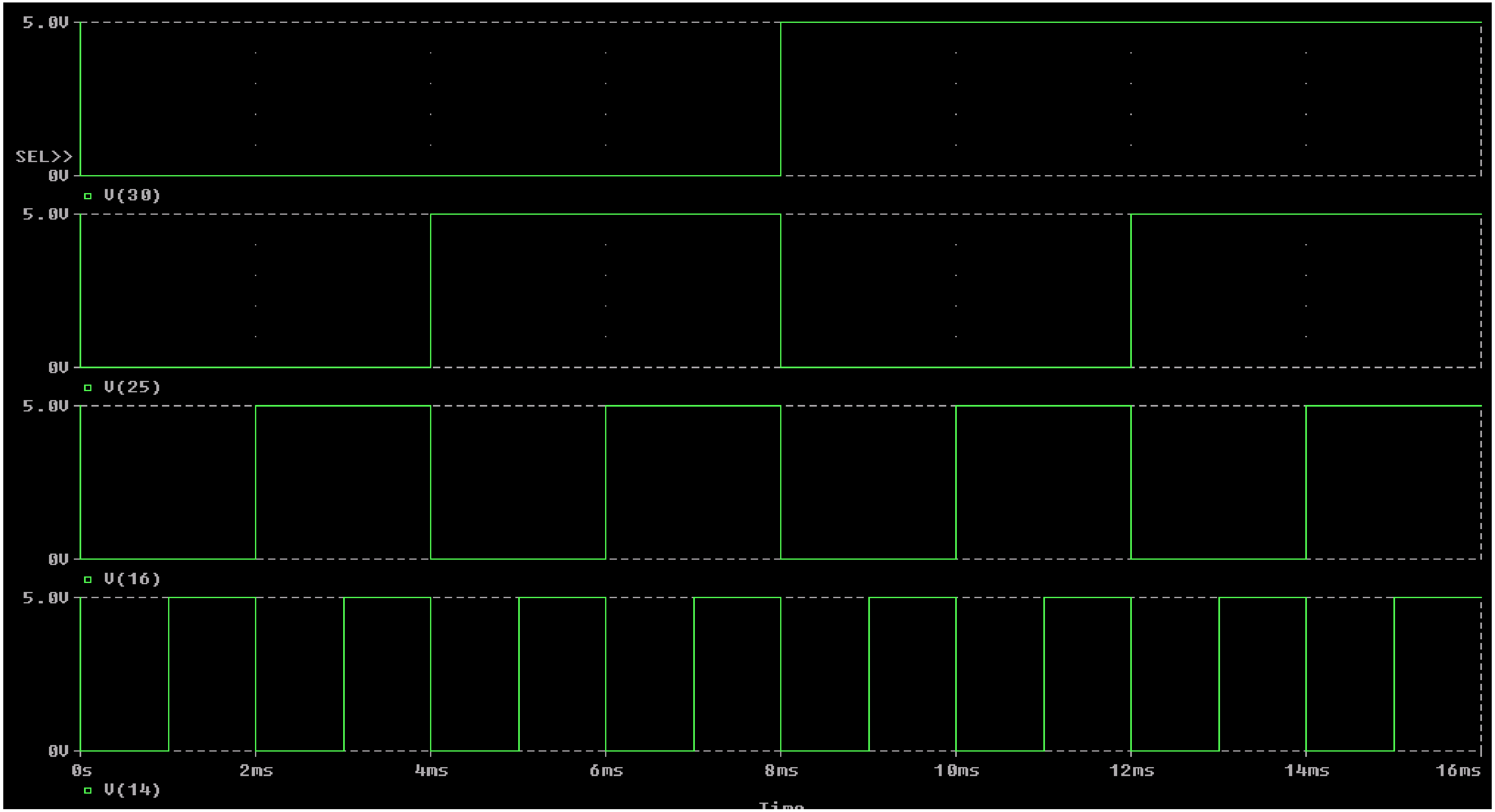
Multiplier Cell - Logical Design



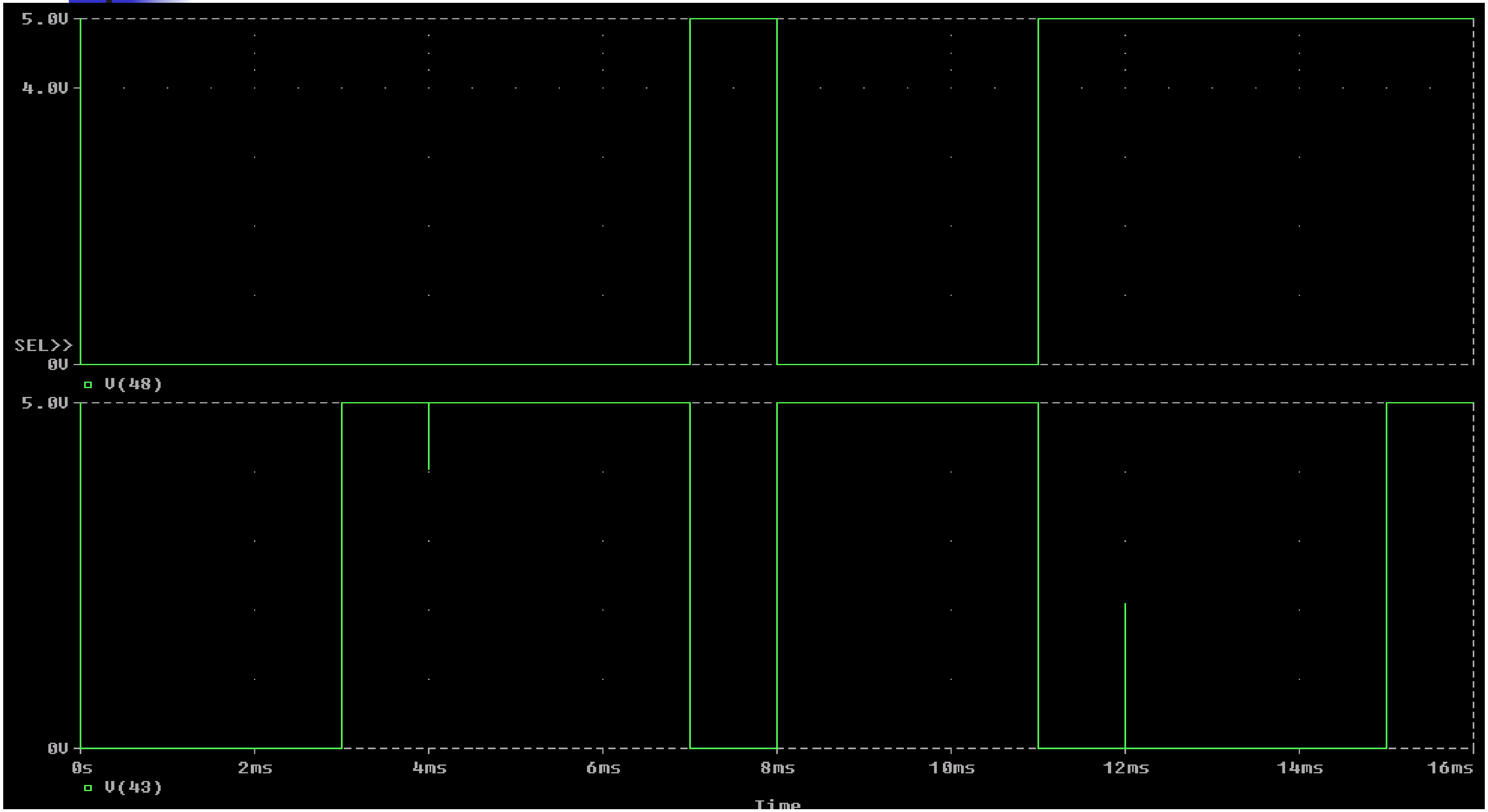
Multiplier Cell - VLSI Design



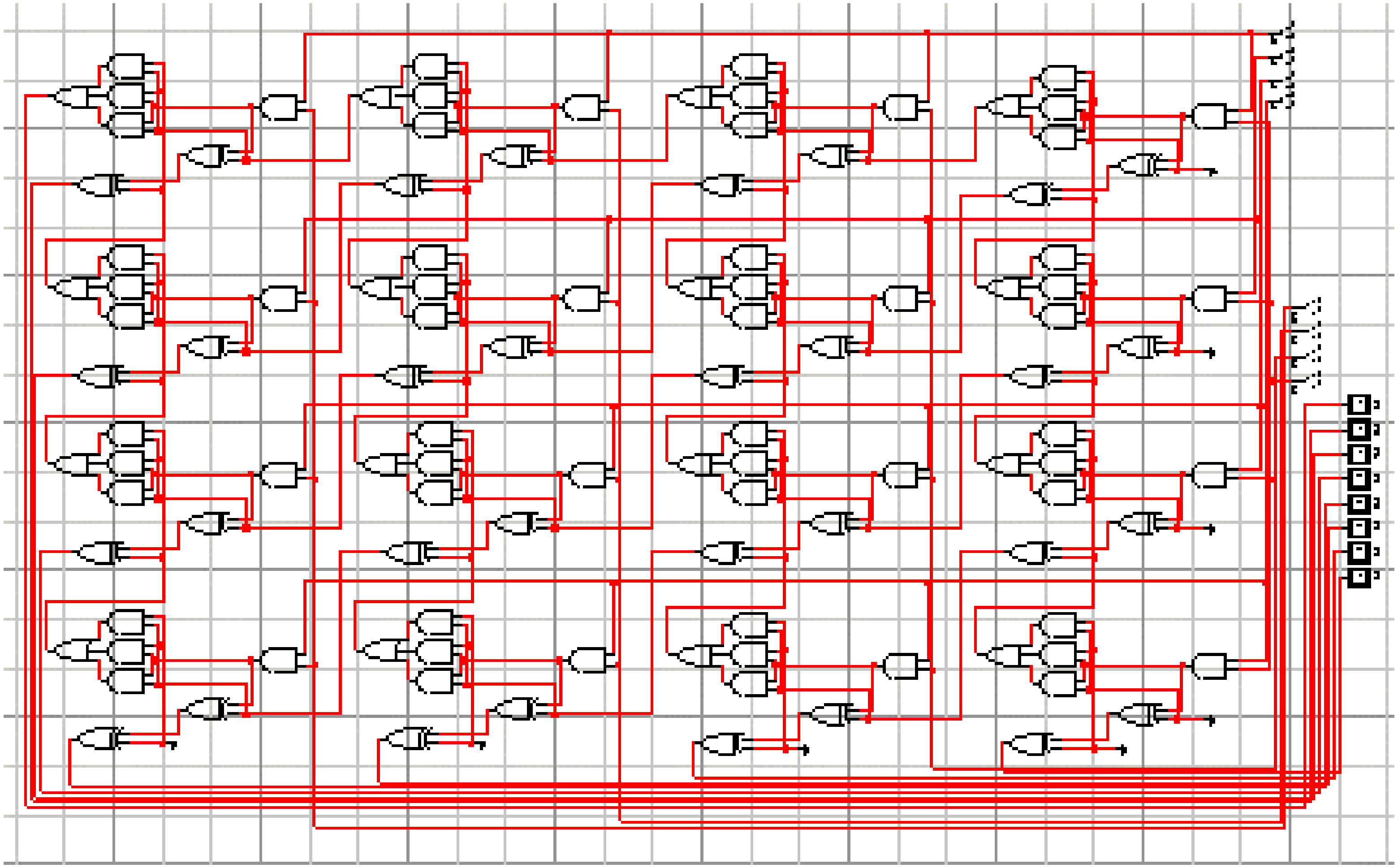
Multiplier Cell Simulation - Inputs



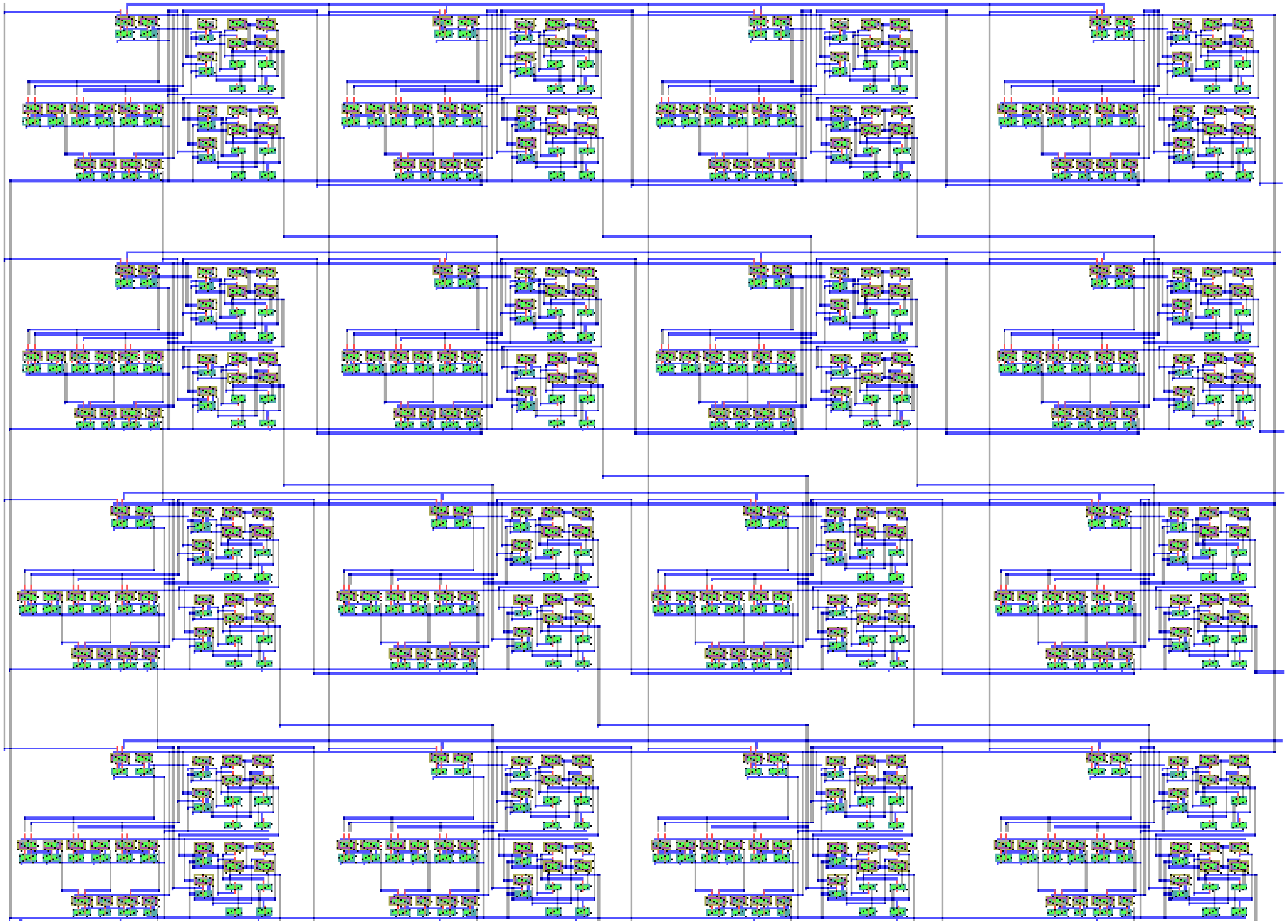
Multiplier Cell Simulation - Outputs



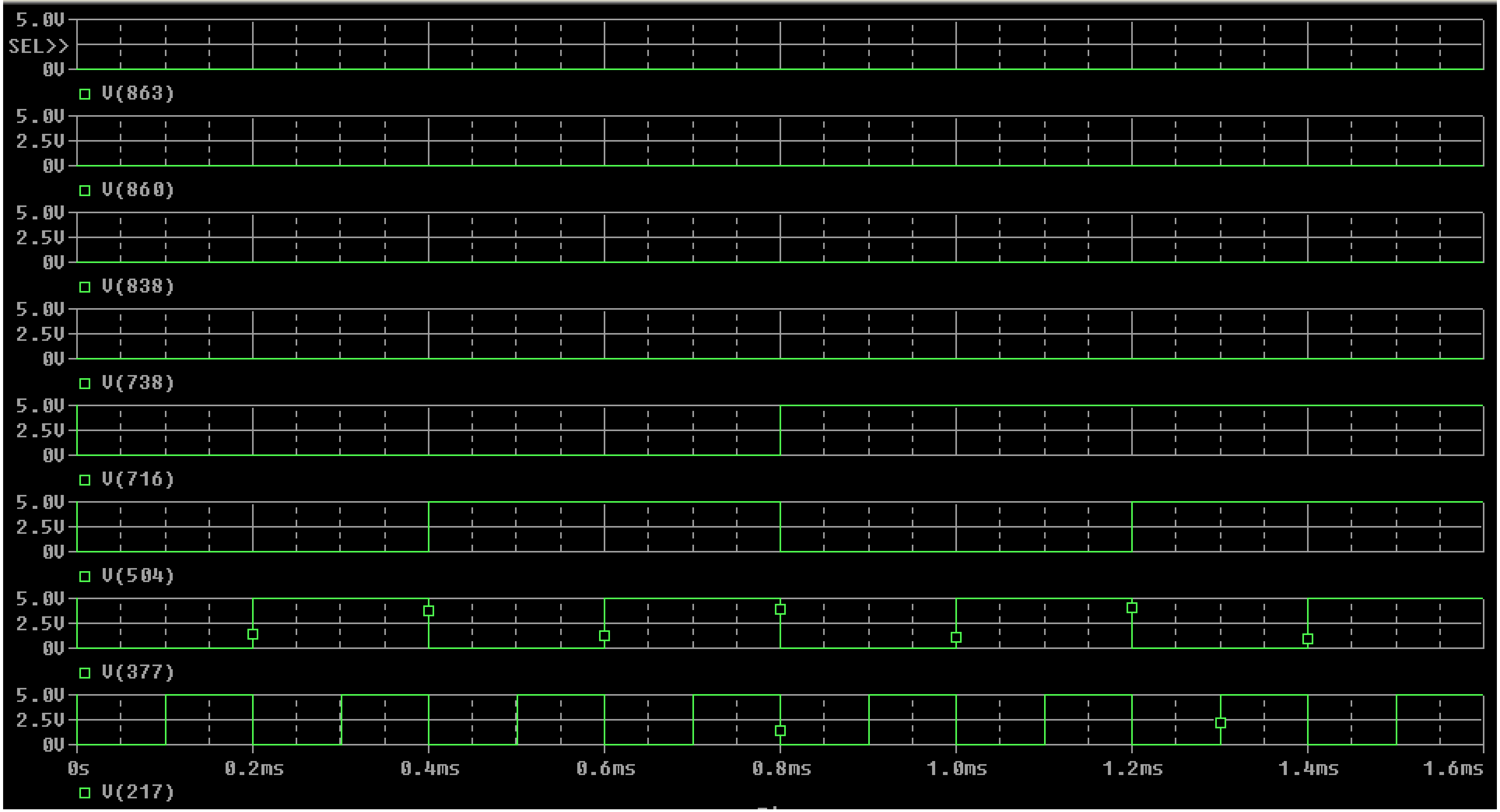
4-bit x 4-bit Multiplier



4-bit x 4-bit Multiplier

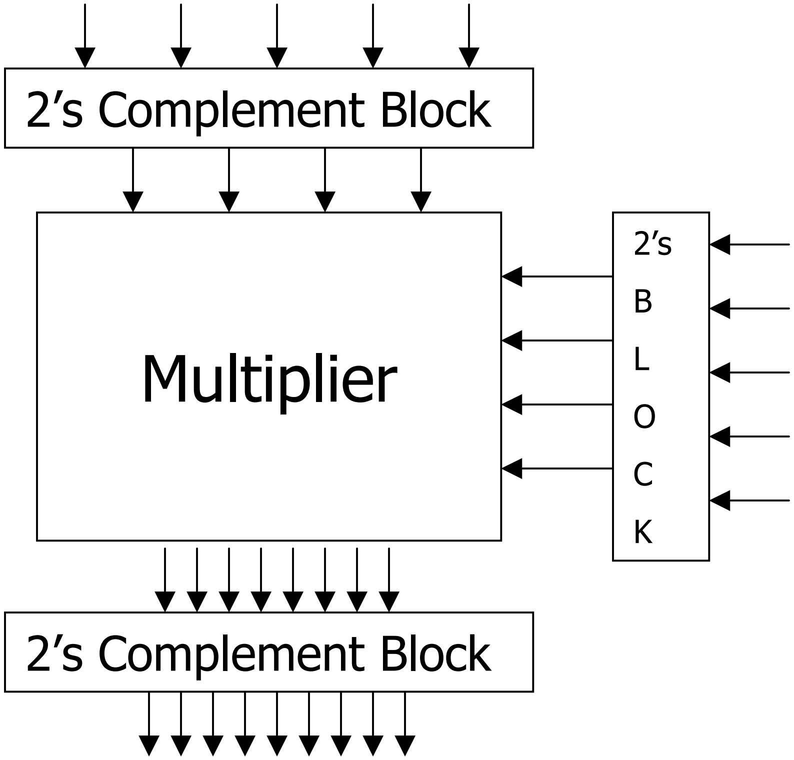


Multiplier Simulation – Multiplied by 1



Adder and Multiplier

- For 2’s complement addition and multiplication:

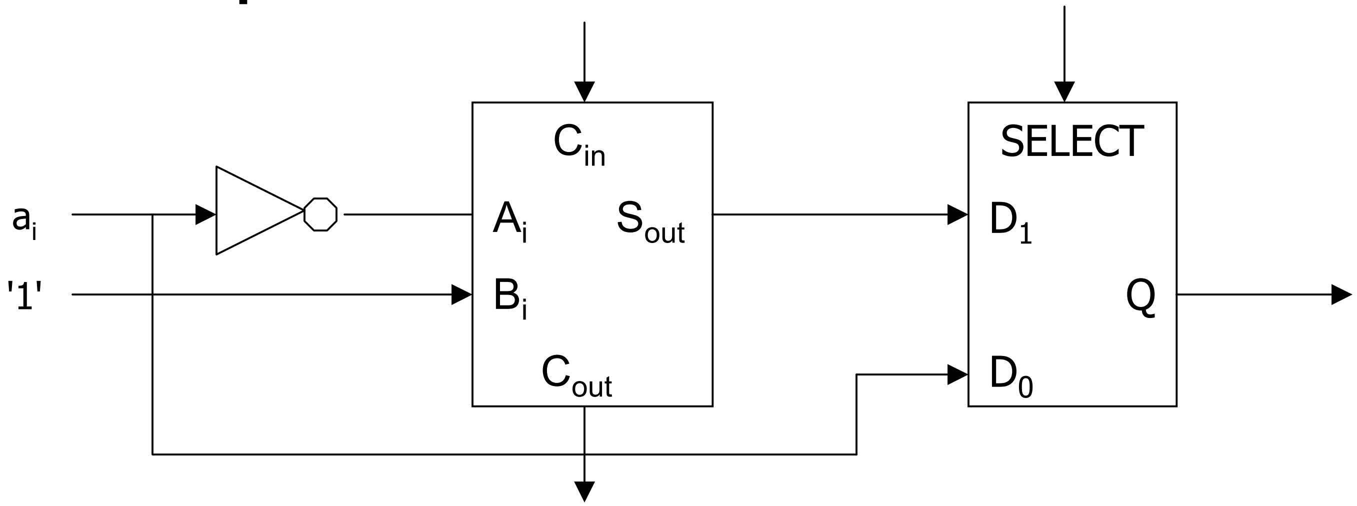


2’s Complement Adjustment

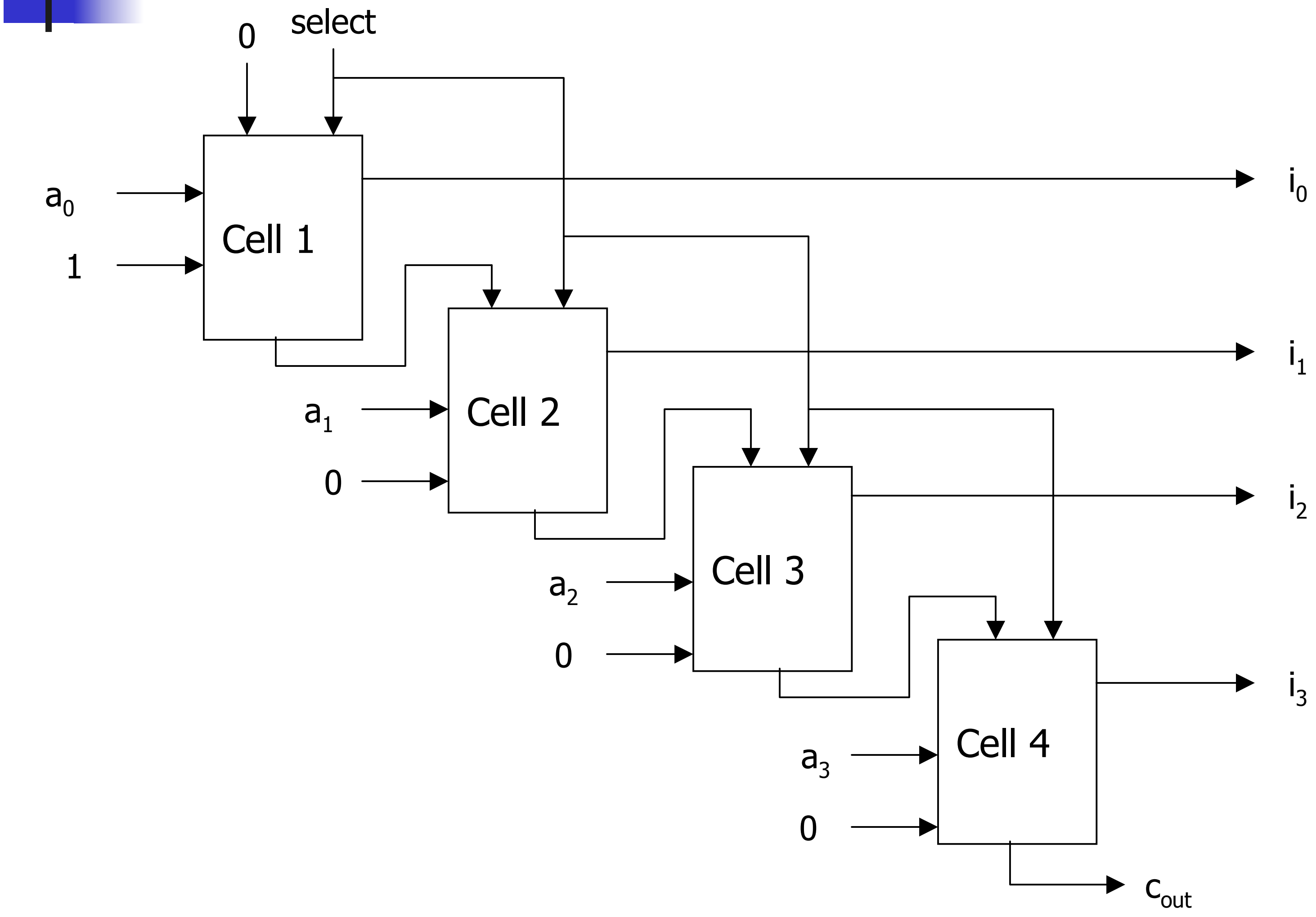
- Need to add circuitry to make the multiplier 2’s complement ready.
- The values from the converter will always be positive but the coefficients will be negative.
- Need blocks on both inputs and the output of the multiplier.
- Special case: input of 10000 and output of 100000000.

2’s Compliment Adjustment

- Basic cell for the adjustment 2’s compliment circuit.



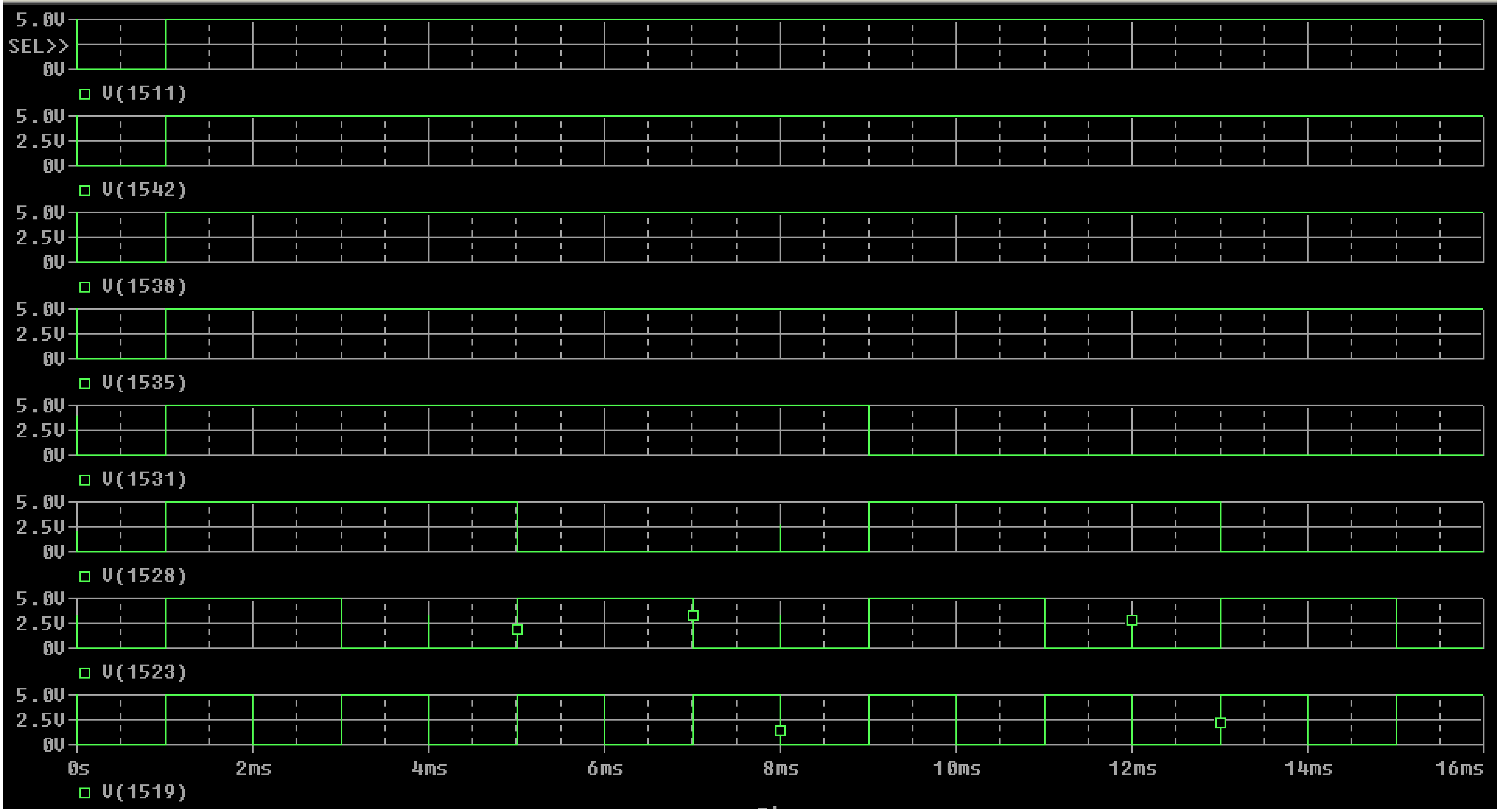
2’s Compliment Adjustment



2’s Complement Adjustment

- The blocks on the previous slide will be cascaded to make the adjustment block for the output adjustment circuitry.
- The select bit for the input will be the sign bit anded with the carry-out bit.
- The select bit for the output will be the same, where the sign bit will be the two input sign bits xored together.

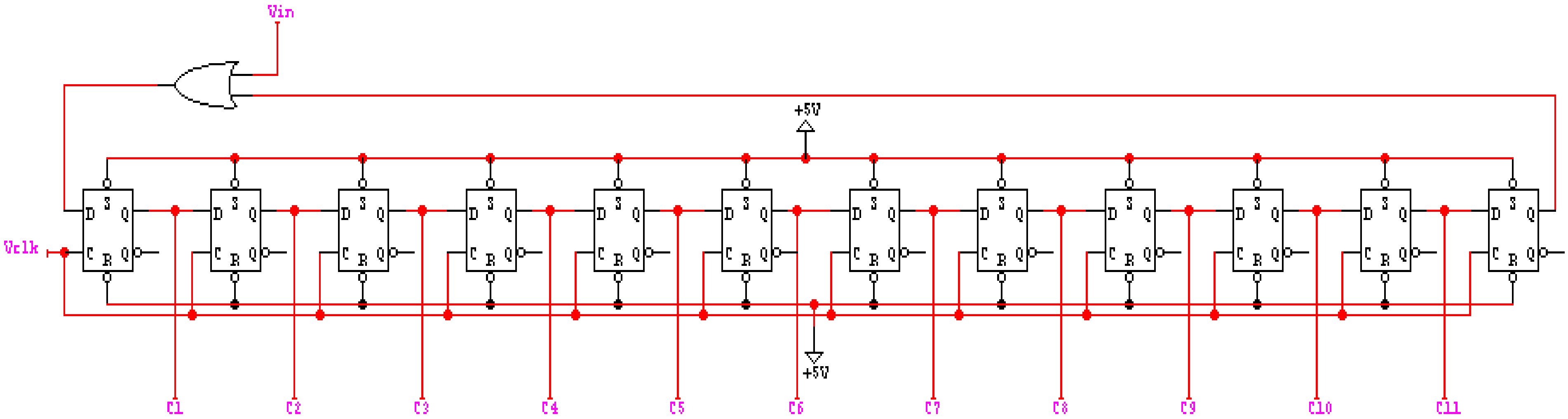
Signed Multiplier Simulation – Multiplied by -1



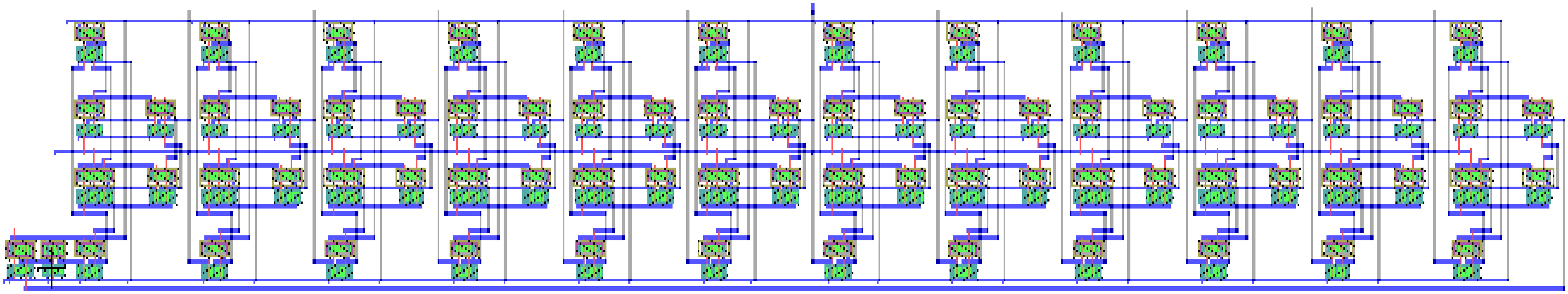
Clock Cycle for Data Management

- 11 different stages that the VLSI processor must go thourgh to complete the required multiplication and addition.
- Used 12 D-type flip-flops to created the clock cycles required.

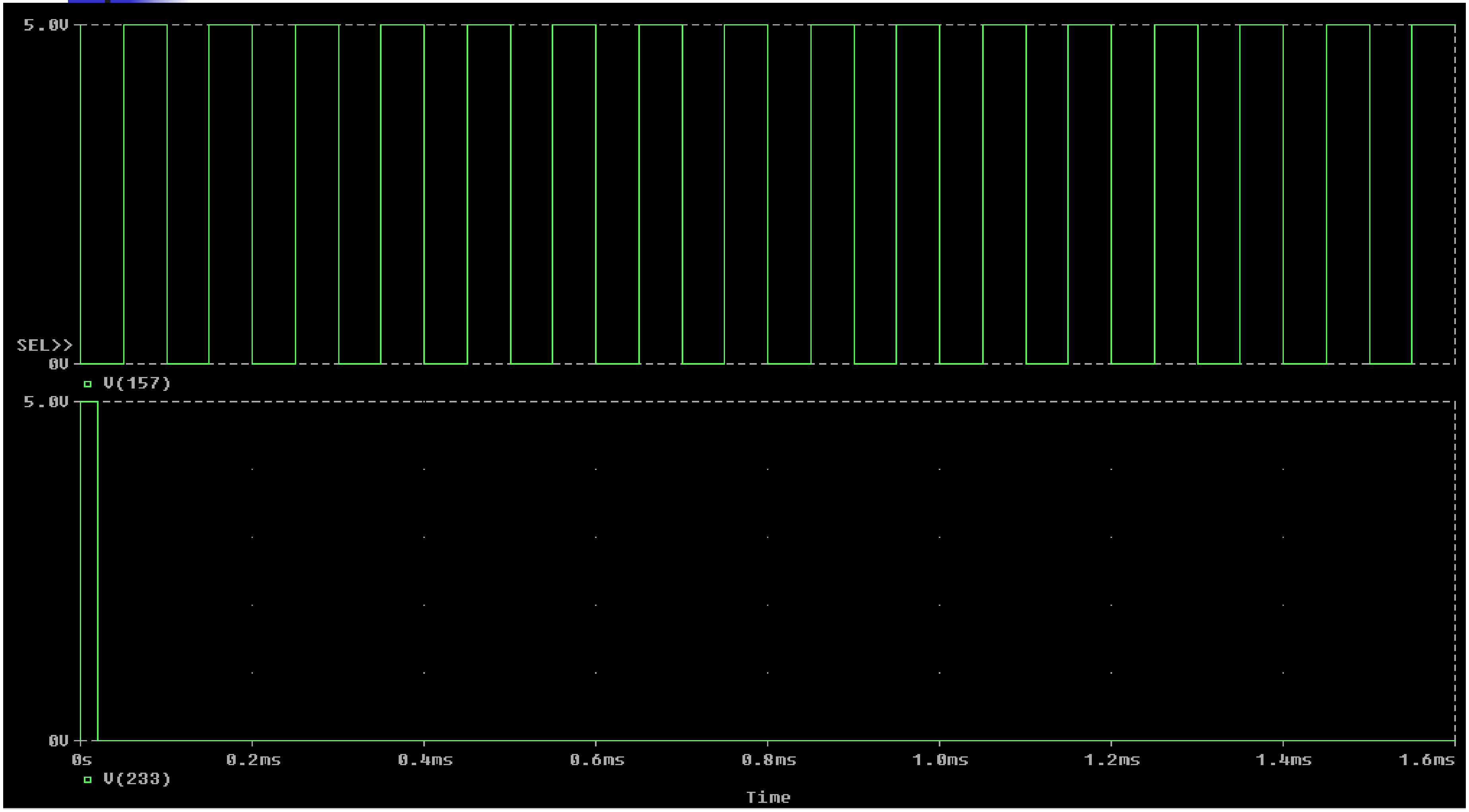
Clock Cycle for Data Management- Logical Design



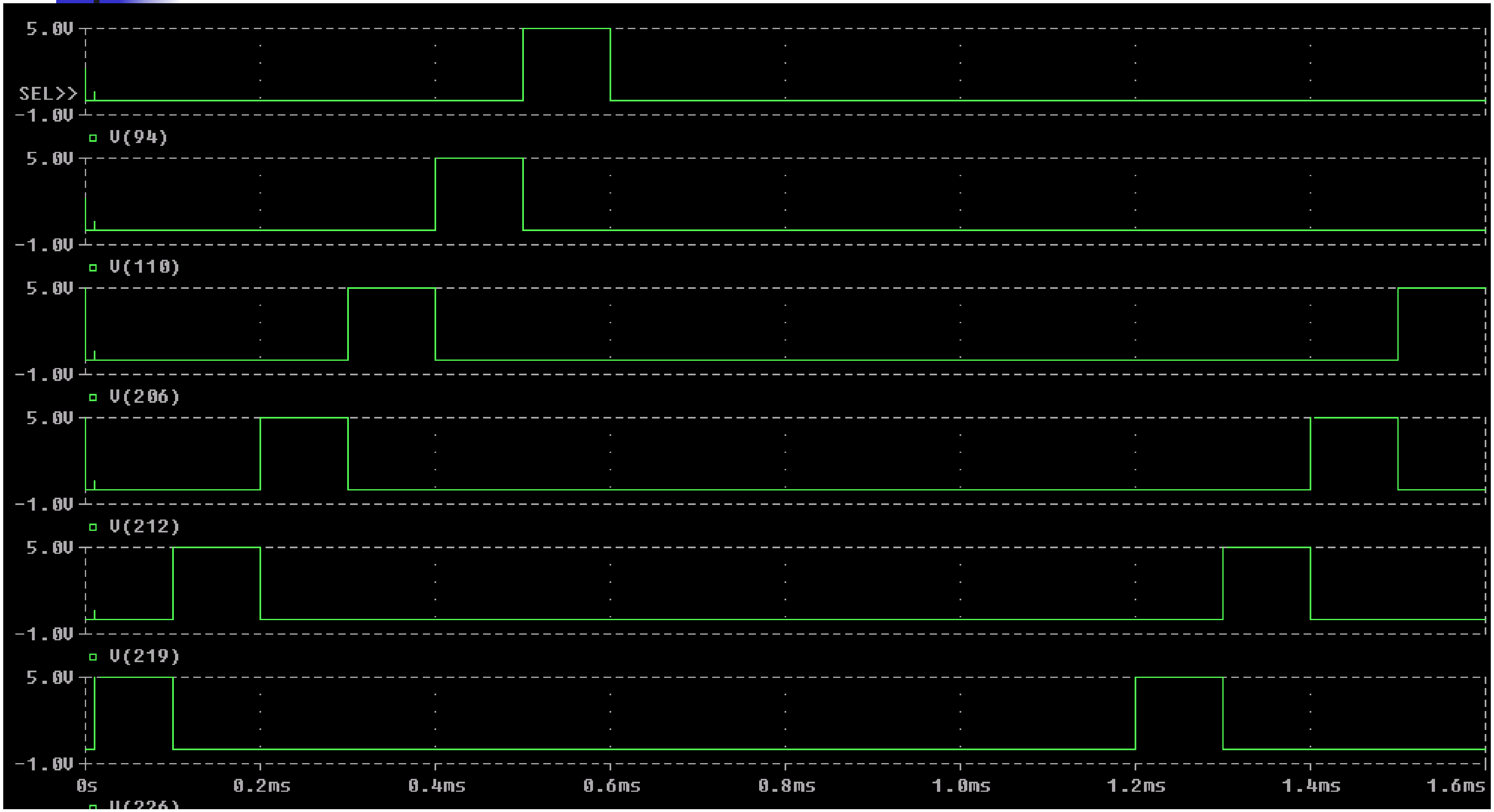
Clock Cycle for Data Management- VLSI Design



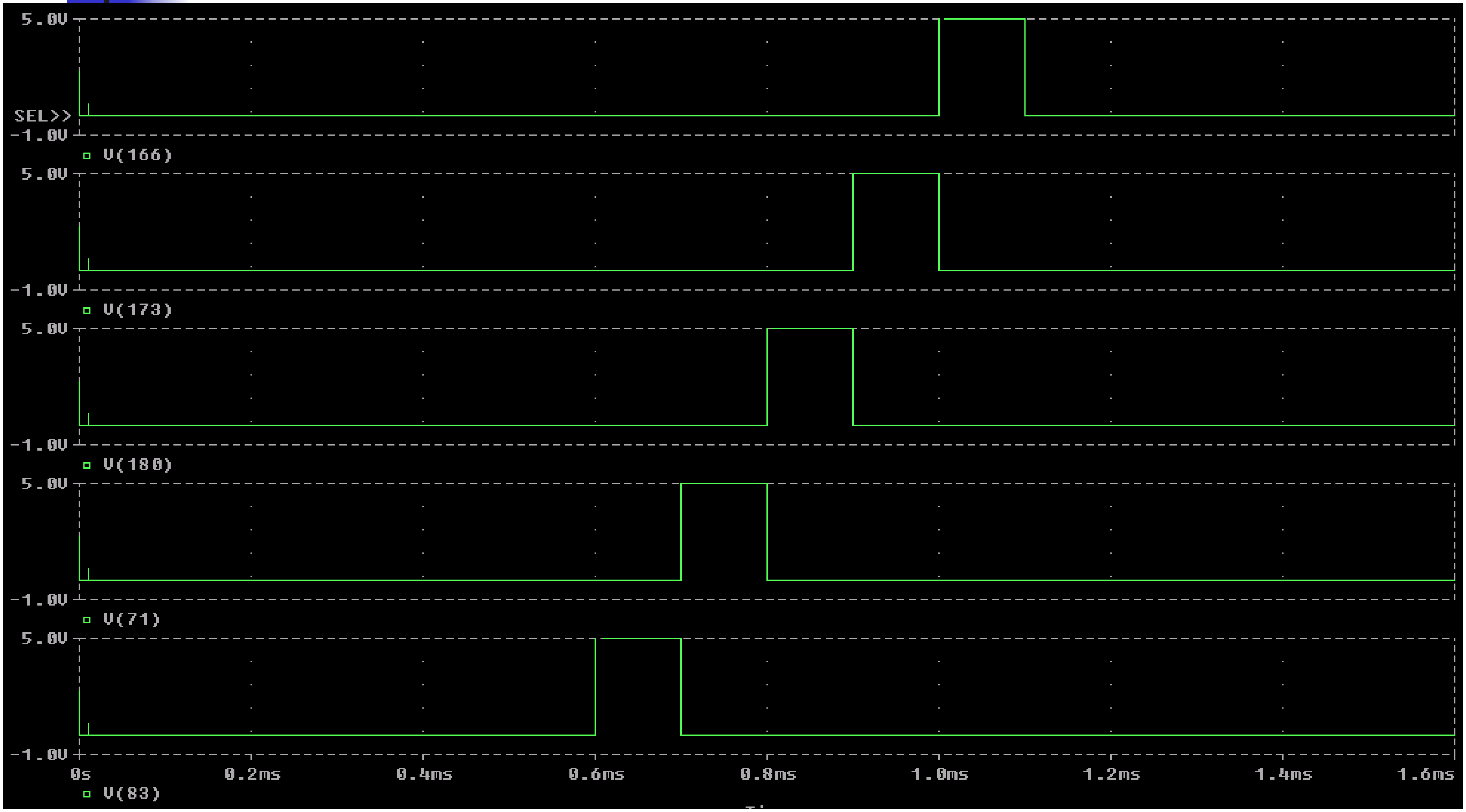
Inputs to the Clock Controller



Outputs C1-C6 from the Clock Controller



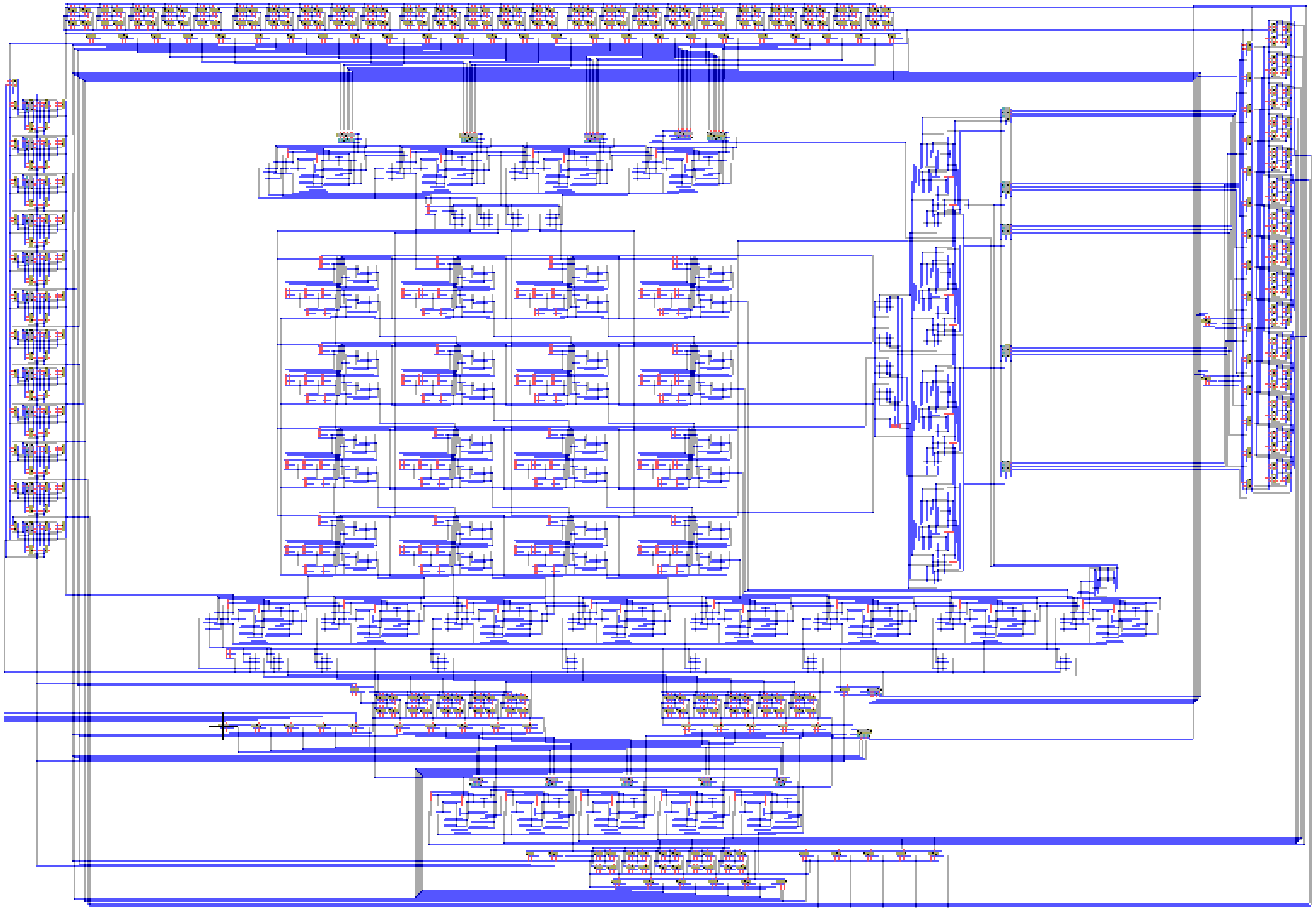
Outputs C7-C11 from the Clock Controller



Cycles for the VLSI Processor

Cycle	Mul1	Mul2	Add1	Add2	Product	Sum
1	a_1	$\omega(n-1)$			R_{mult}	
2			$x(n)$	R_{mult}		R_{add}
3	a_2	$\omega(n-2)$				$\omega(n)$
4			R_{add}	R_{mult}		
5	b_0	$\omega(n)$			R_{temp}	
6	b_1	$\omega(n-1)$			R_{mult}	
7			R_{temp}	R_{mult}		R_{add}
8	b_2	$\omega(n-2)$			R_{mult}	
9			R_{add}	R_{mult}		Y_{out}
10	$\omega(n-1)$ -> $\omega(n-2)$					
11	$\omega(n)$ -> $\omega(n-1)$					

VLSI Digital Signal Processor



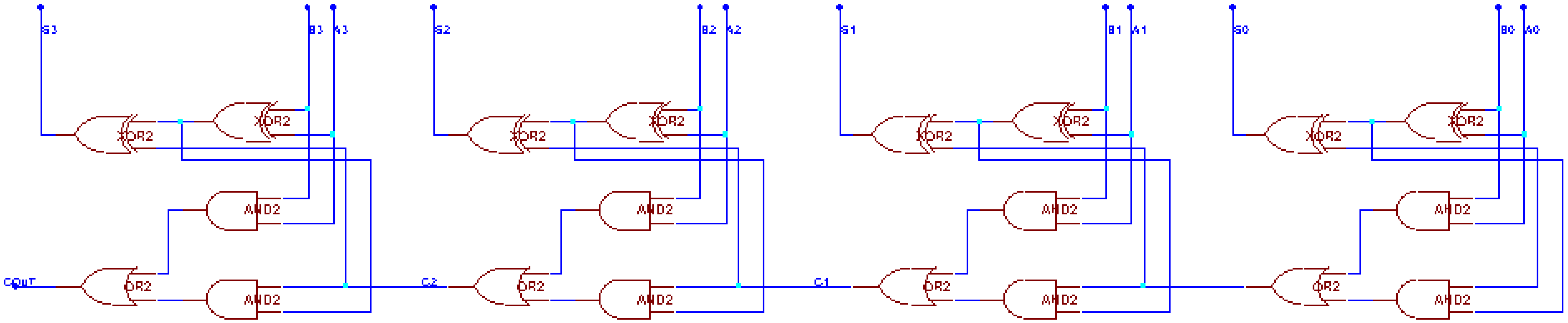
VLSI Troubleshooting

- Needed to add overflow protection for the adder.
- Investigate the speed of the entire processor.

Investigation

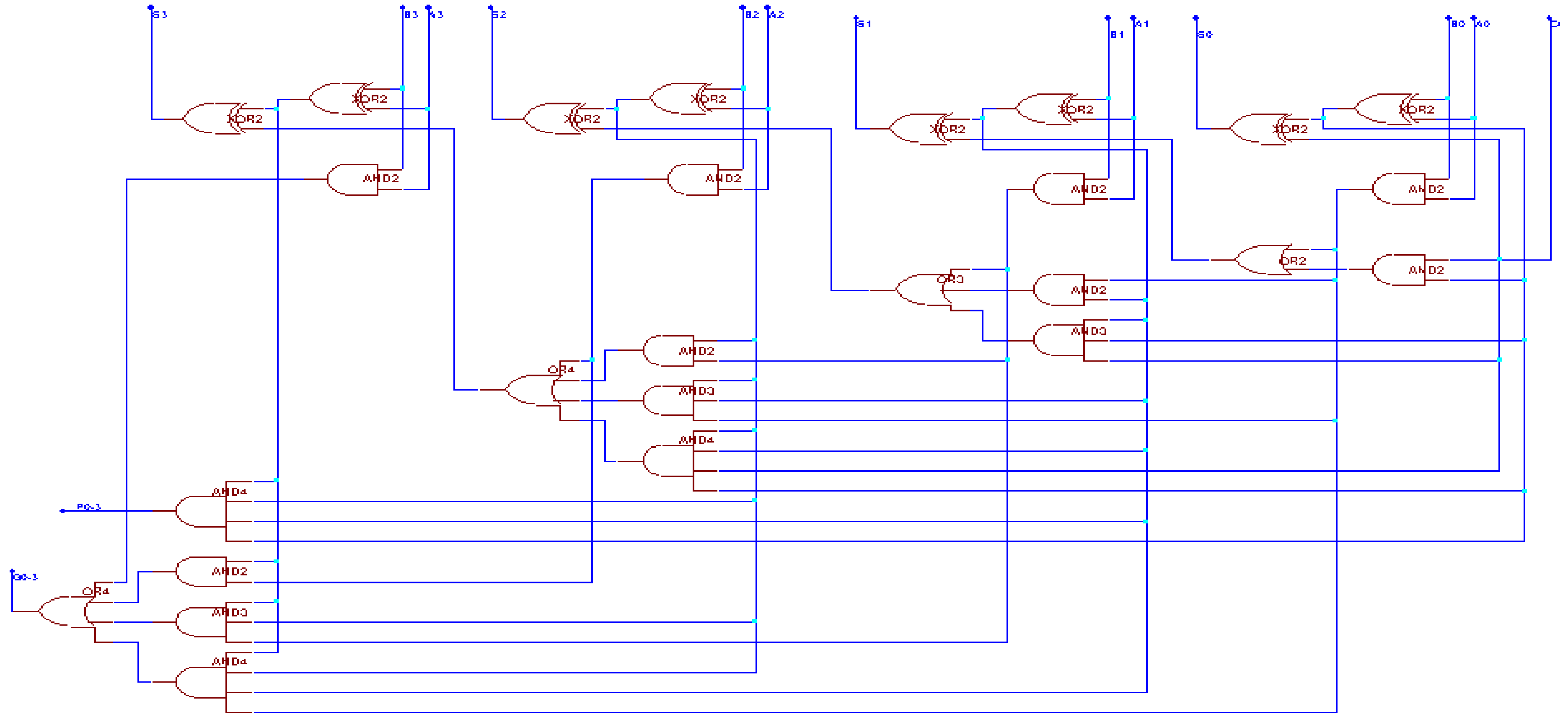
- Behavioral v. Structural
- Ripple carry adder v. Carry look ahead adder.
- Parallel multiplier v. Serial multiplier.

Ripple Carry Adder



*   **16-bit ripple carry adder will have 34 gate delays**

CLA Adder



*   **16-bit CLA adder will have 10 gate delays**

Multiplier

- Advantages and disadvantages of using a parallel multiplier v. a serial multiplier.
- Speed v. Area

Area v. Speed

- Implemented serial and parallel multipliers in VHDL.

	Area	Delay	Area	Delay	Area	Delay
Serial Multiplier	7.14%	96.88ns	13.52%	210.8ns	24.49%	503.68ns
Parallel Multiplier	10.71%	27.41ns	28%	38.76ns	68.50%	75.12ns

Booth’s Multiplier

- To increase the speed a Modified Booth’s Algorithm was used
- For (X)*(Y)

Bit			Operation
Yi+1	Yi	Y	
0	0	0	add zero
0	0	1	add X
0	1	0	add X
0	1	1	add 2X
1	0	0	subtract 2X
1	0	1	Subtract X
1	1	0	Subtract X
1	1	1	Subtract 0

Booth’s Multiplier

- For example (X*Y)= 4 (0100) * -3(1101)
- If it isn’t an odd number of bits add a 0 to Y
- Segment multiplier (Y): 11010

  1(010)

  2(110)

Segment #	bits	Action
1	010	Add X
2	110	Subtract X

Booth’s Multiplier

```
     0100
    x1101
 00000100  (Add X)
 111100    (Sub X)
 11110100 =-12
```

- A N-bit multiplier requires N/2 adds
- Results in a 60 ns delay for a 16 bit multiplier

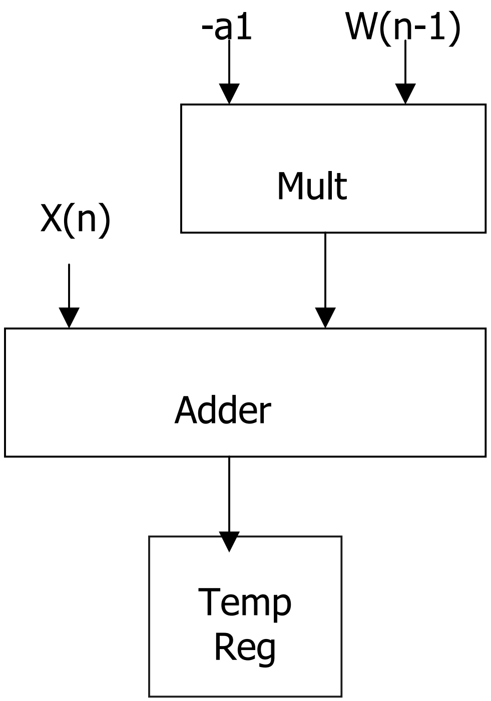
Data Management

- Data Management is necessary to load the appropriate data to the multiplier and adder at the appropriate time, and to store data for later use.
- $W(n)=X(n)-a_1W(n-1)-a2W(n-2)$

  $Y(n)=b_0W(n)+b_1W(n-1)+b_2W(n-2)$
- To accomplish this I used 6 cycles

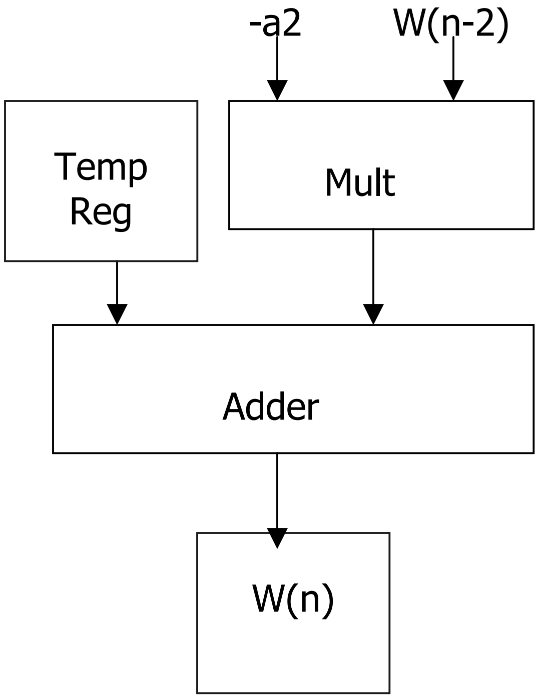
Data Management

Cycle 1



$X(n)-a_1W(n-1)$

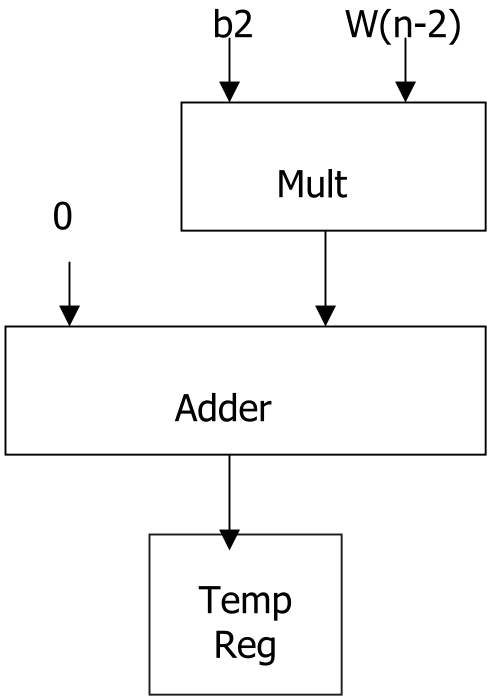
Cycle 2



$X(n)-a_1W(n-1)-a_2W(n-2)=W(n)$

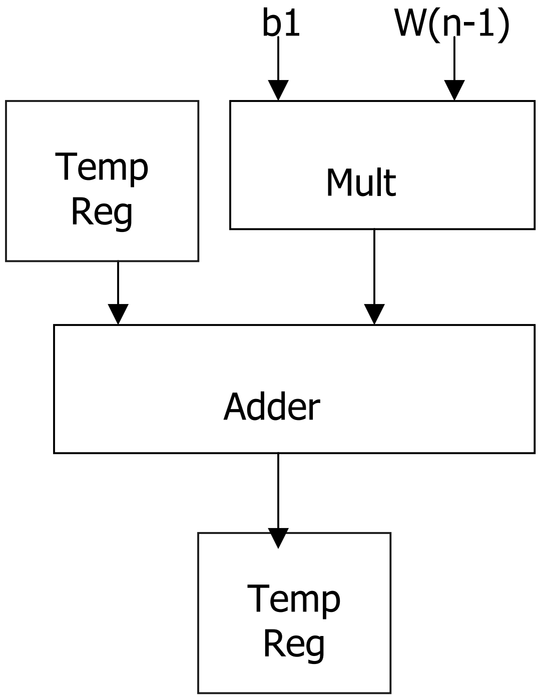
Data Management

Cycle 3



$b_0W(n)$

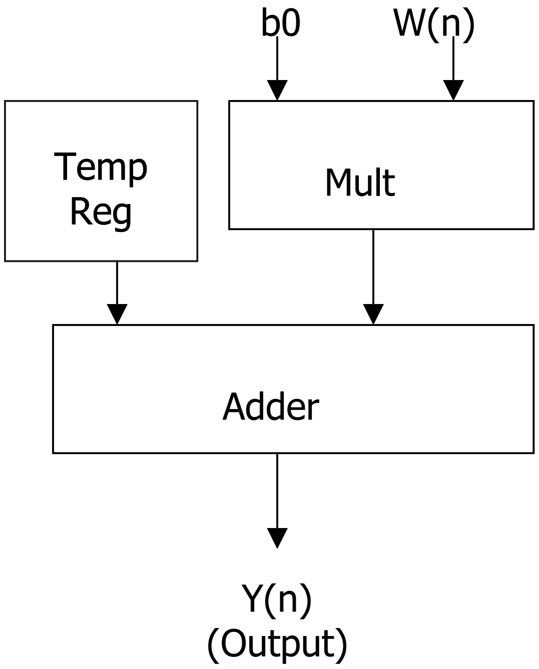
Cycle 4



$b_0W(n)+b_1W(n-1)$

Data Management

Cycle 5



b0

W(n)

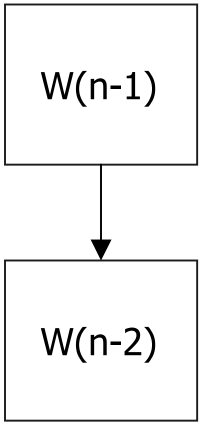
Temp Reg

Mult

Adder

Y(n) (Output)

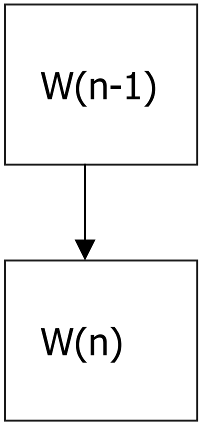
Cycle 5



W(n-1)

W(n-2)

Cycle 6



W(n-1)

W(n)

$$b_0W(n)+b_1W(n-1)+b_2W(n-2)=Y(n)$$

Data Management

- Three 3-input multiplexers were used to accomplish this task.

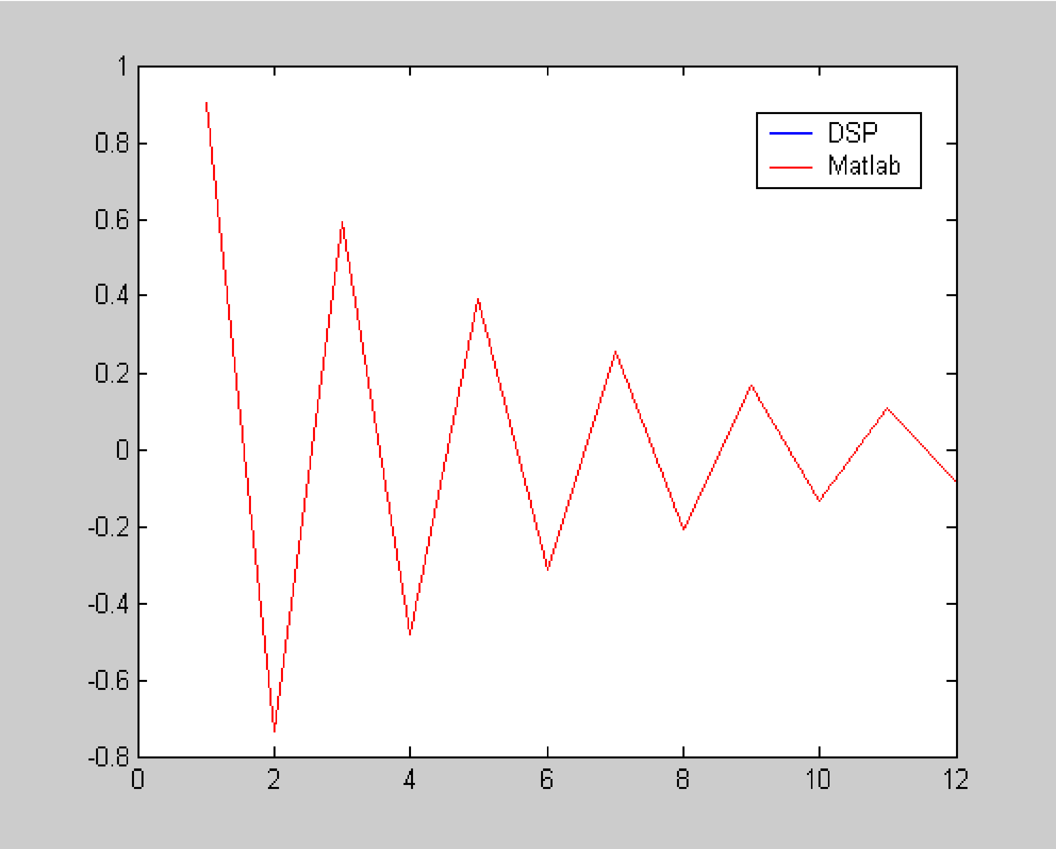
Cycle	**Mul1**	**Mul2**	**Add**		
000	This cycle is used to trigger the A/D convertor				
001	a1	W(n-1)	X(n)		
010	a2	W(n-2)	Temp Reg		
011	b2	W(n-2)	0		
100	b1	W(n-1)	Temp Reg		
101	b0	W(n)	Temp Reg	W(n-1) =>W(n-2)	
110	W(n) => W(n-1)				

Simulation

- Simulated DSP on Modelsim using simple 2nd order low pass filter
- Checked the results with the same filter using matlab

Simulation

Matlab	DSP
0.905	0.905
-0.7331	-0.733
0.5938	0.594
-0.481	-0.481
0.3896	0.3895
-0.3156	-0.3156
0.2556	0.2555
-0.207	-0.2071
0.1677	0.1676
-0.1358	-0.1359
0.11	0.11



Problems

- Multiplier was occasionally producing an extra sign bit
- FPGA clocks
- Impulse response degradation

Signal Processing Using Digital Technology

Jeremy Barsten

Jeremy Stockwell

May 6, 2003

Advisors:

Dr. Thomas Stewart

Dr. Vinod Prasad