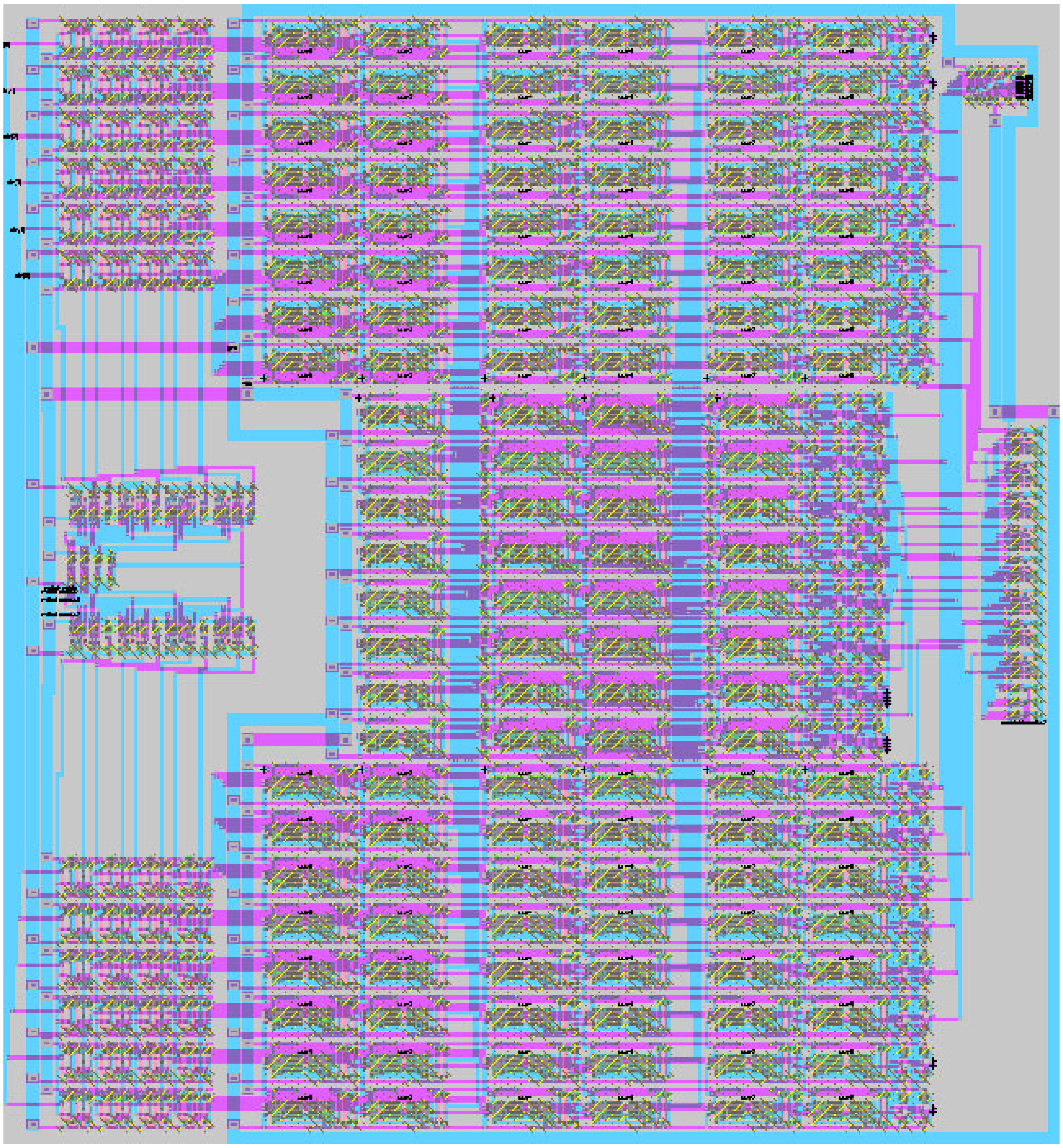
A 6-Bit, 4-Tap Fast Fourier Transform Circuit



James Speros
Mike Sakasegawa
Intro to CMOS VLSI Design
Prof. Harris

Functional Overview

Our system implements a six-bit, four-tap fast Fourier transform circuit. A four-tap fast Fourier transform (FFT) computes the Fourier coefficients of the spectrum of an input sequence that consists of four complex numbers. Our system generates these coefficients for a sequence of four numbers, each of which consists of a six-bit real component and a six-bit imaginary component. In general, an N-tap FFT is accomplished using the following equation:

$$X[k] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-jk\left(2p/N\right)n}, k = 0,\ldots, N-1$$

For a four-tap FFT, this simplifies to:

$$X[0] = \frac{1}{4}\left[x[0] + x[1] + x[2] + x[3]\right]$$
$$X[1] = \frac{1}{4}\left[x[0] - x[2] + j\left(x[1] - x[3]\right)\right]$$
$$X[2] = \frac{1}{4}\left[x[0] - x[1] + x[2] - x[3]\right]$$
$$X[3] = \frac{1}{4}\left[x[0] - x[2] - j\left(x[1] - x[3]\right)\right]$$

In our design, the real and imaginary parts are kept separate and all the multiplication is trivial (shifts or multiplication by 1 or –1).

Because there are only 34 i/o pins available, there are not enough pins to import and export the real and imaginary parts of each tap at the same time. Therefore, six bits are input at one time, along with a three-bit value designating which tap it is, as well as whether it is the real or imaginary part of that tap. This process is controlled by a write enable. The values are registered and fed to the FFT circuitry entirely in parallel. The output of the FFT circuitry is multiplexed using a tristate multiplexer such that it is available one six-bit packet at a time. The external system can select which packet to read through another three input pins. The mapping of address to tap is as shown in the table, below.

Address	***Input Value***	***Output Value***
000	Im{x[0]}	Im{X[0]}
001	Im{x[1]}	Im{X[1]}
010	Im{x[2]}	Im{X[2]}
011	Im{x[3]}	Im{X[3]}
100	Re{x[0]}	Re{X[0]}
101	Re{x[1]}	Re{X[1]}
110	Re{x[2]}	Re{X[2]}
111	Re{x[3]}	Re{X[3]}

In general, the addition of two n-bit number results in an (n+1)-bit sum. In a digital system, this can lead to errors of overflow and underflow. In our system, the six-bit values are added to become seven-bit values, which are in turn added to produce eight-bit values. In order to avoid problems of overflow and underflow, the internal computations carry through all eight bits. Then the division by four is accomplished by truncating the lower two bits. However, the truncation does lead to rounding error. Specifically, any value not evenly divisible by four is rounded down. For example, $5 \div 4 = 1$, and $-5 \div 4 = -2$.

Chip Floorplan

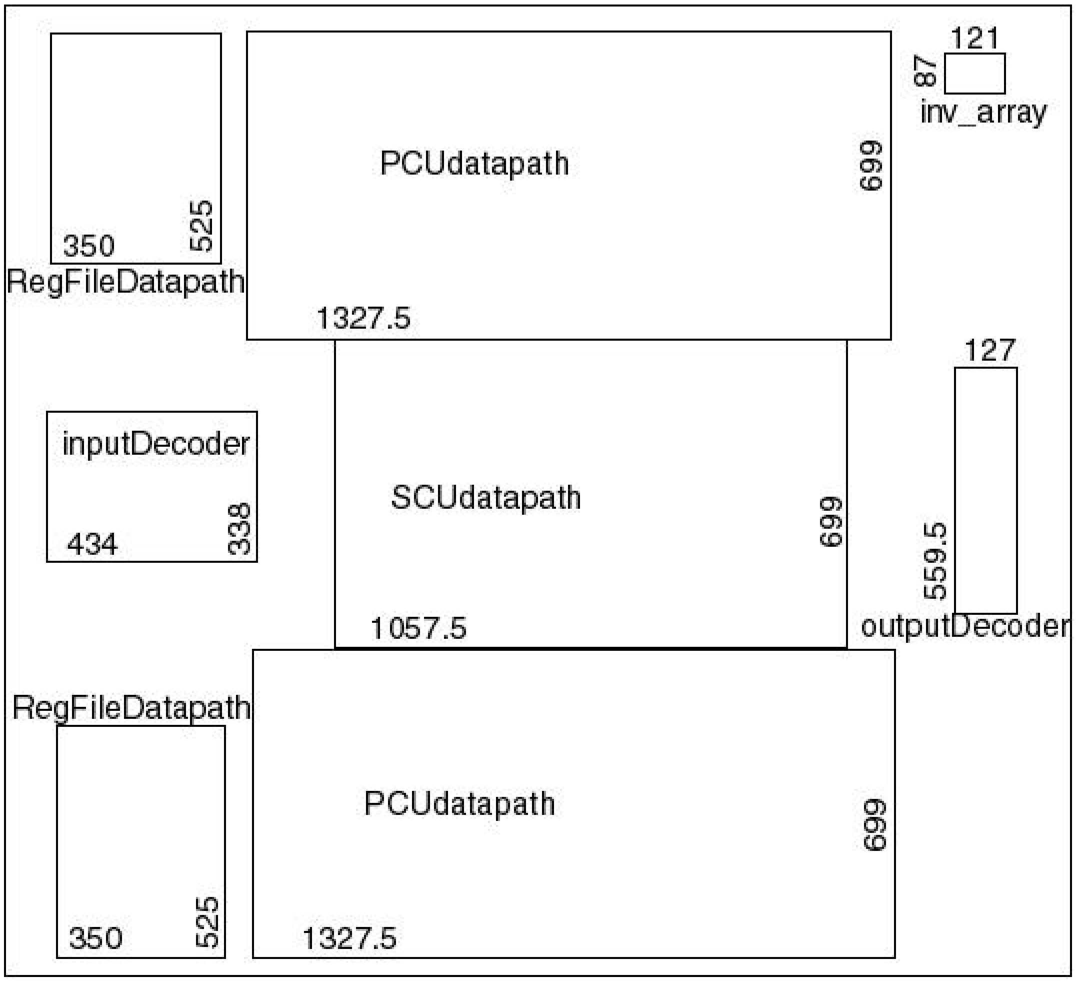


Figure 1: FFT Floorplan

Figure 1, above, shows the floorplan for the FFT circuit. The large square surrounding the modules represents the area constraint of 2200 λ x 2200 λ. The entire top-level module has dimensions 1958.75 λ x 2116.5 λ, which falls within the constraint.

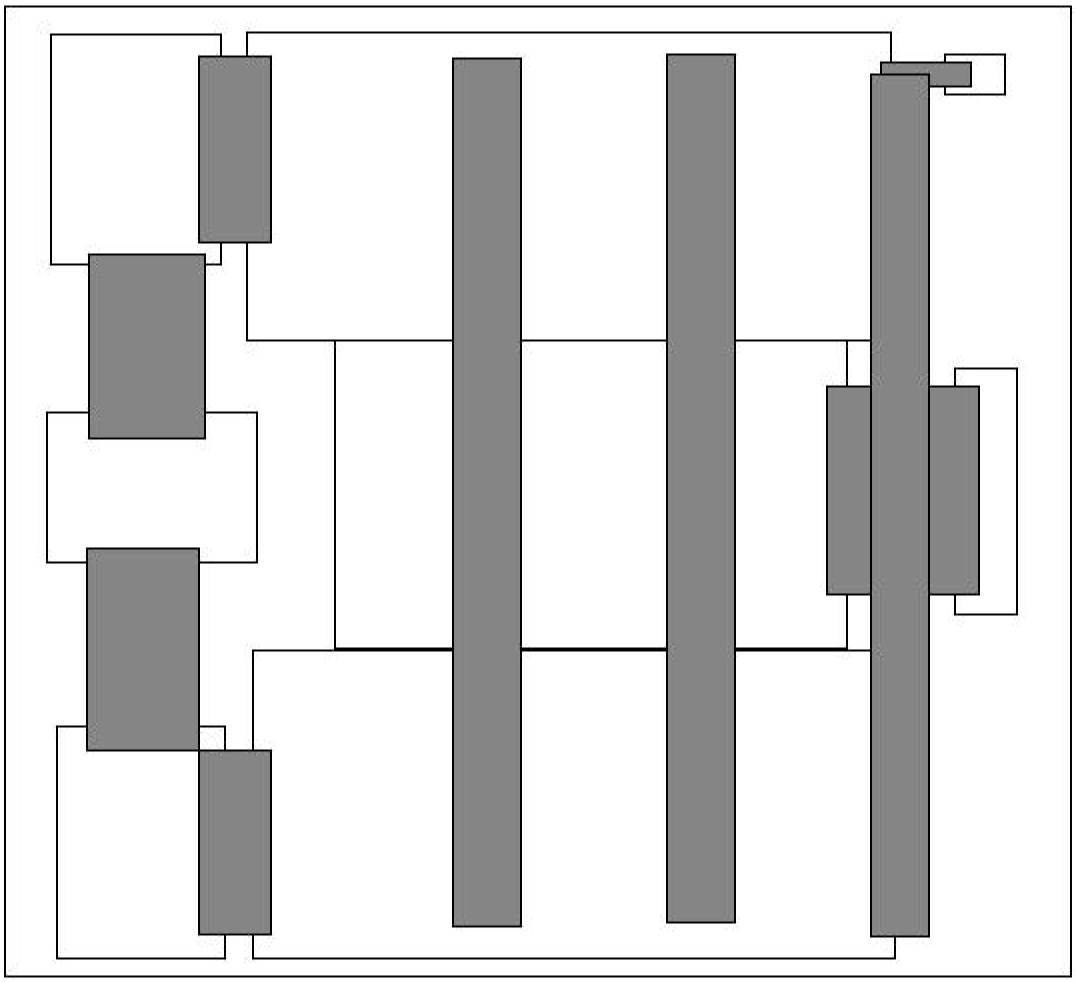


Figure 2: Floorplan with interconnect.

In figure 2, above, the interconnections between cells are represented by the gray squares. Much of the routing between the primary and secondary unit was done over the top of the cells by including routing channels in each bitslice. Each routing channel had to be wide enough to accommodate 8 metal lines.

Area and Design Time

Cell	***Dimensions (l x l)***	***Area*** (l^2)	***Estimated Area*** (l^2)	***Design Time (hrs)***
Full Adder	170 x 87	14,790	12,800	5
Inverter	25 x 87	2,175	1,600	2
Tristate Inverter	30 x 87	2,610	3,200	2.5
Latch	75 x 87	6,525	5,600	3

Inverter Array	121 x 87	10,527	N/A	1
Register File Bitslice	295.5 x 87	25,708.5	N/A	3
Register File Datapath	350 x 525	18,3750	268,800	1
Primary Computation Unit Bitslice	1,269 x 87	110,403	N/A	6
Primary Computation Unit Datapath	1,327.5 x 699	927,922.5	704,000	2
Secondary Computation Unit	999 x 87	86,913	N/A	4
Secondary Computation Unit Datapath	1057.5 x 699	739,192.5	524,800	1
Input Address Decoder	434 x 338	168,392	172,000	2
Output Address Decoder	559.5 x 127	71,056.5	44,000	2
Top Level	1958.75 x 2116.5	4,145,694.375	2,686,400	10

Simulation Results

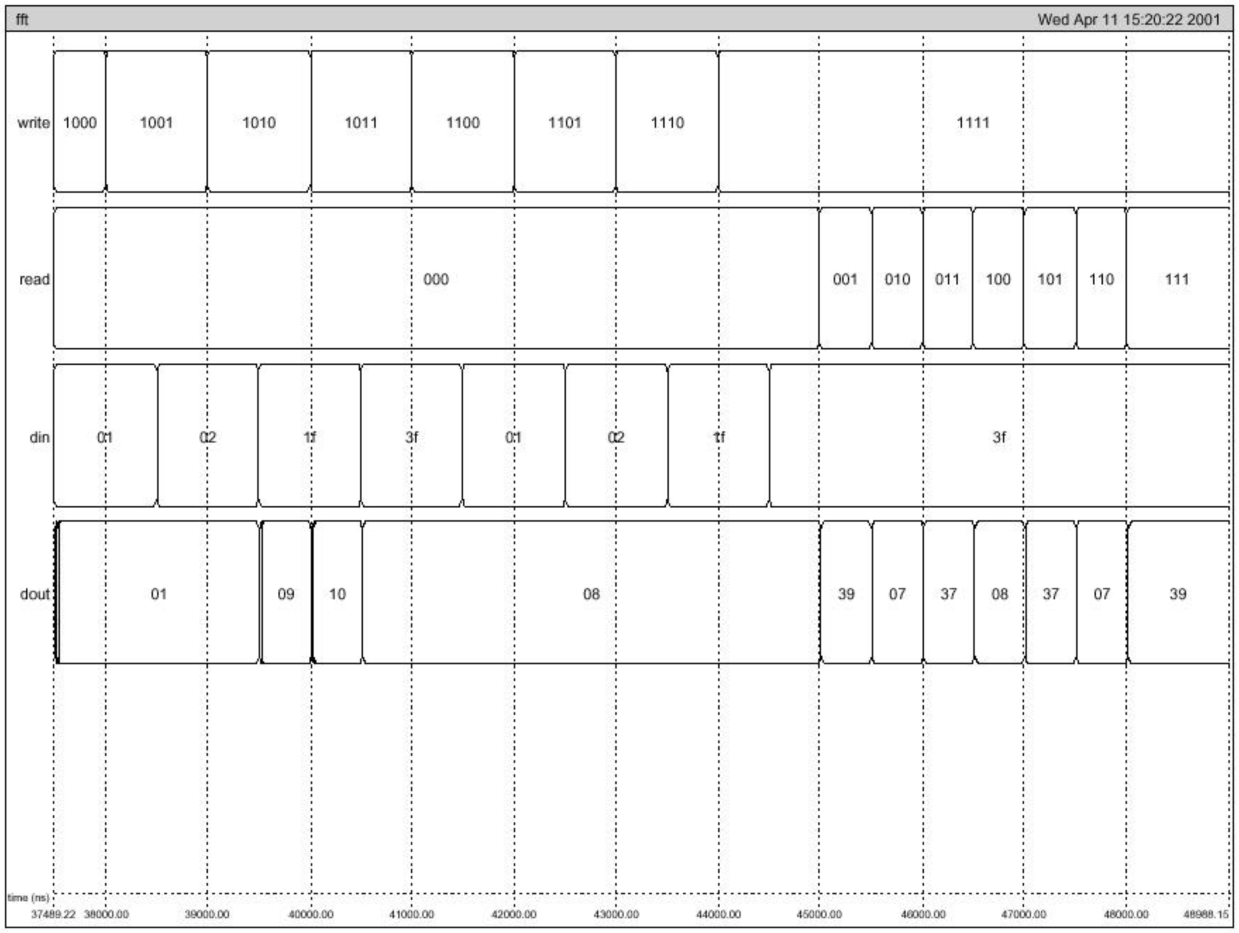


Figure 3: Sample Simulation Waveforms

In order to verify that our design was correct, we simulated a number of test cases. First, we set all of the input values to one and checked the output. In the case of all input values being equal, the 0th output is equal to the input, and all others are zero. The system passed that test. We repeated the test for the values of –1, 2, 31 (the maximum value), and –32 (the minimum value). The system passed all of those tests. We then tested a semi-random mix of those values, as is shown in figure 3, above. The results are tabulated in decimal form, below.

Input	*Value*	*Output*	*Expected Value*	*Actual Value*
Im{x[0]}	1	Im{X[0]}	8	8
Im{x[1]}	2	Im{X[1]}	-7	-7
Im{x[2]}	31	Im{X[2]}	7	7
Im{x[3]}	-1	Im{X[3]}	-9	-9
Re{x[0]}	1	Re{X[0]}	8	8
Re{x[1]}	2	Re{X[1]}	-9	-9
Re{x[2]}	31	Re{X[2]}	7	7
Re{x[3]}	-1	Re{X[3]}	-6	-6

As predicted, the truncation results in downward rounding of all values not evenly divisible by four. Finally, we tested a set of random numbers. The results are tabulated below.

Input	*Value*	*Output*	*Expected Value*	*Actual Value*
Im{x[0]}	5	Im{X[0]}	7	7
Im{x[1]}	20	Im{X[1]}	0	0
Im{x[2]}	11	Im{X[2]}	0	0
Im{x[3]}	-6	Im{X[3]}	-4	-4
Re{x[0]}	-31	Re{X[0]}	-7	-7
Re{x[1]}	-17	Re{X[1]}	-10	-10
Re{x[2]}	-20	Re{X[2]}	19	19
Re{x[3]}	8	Re{X[3]}	3	3

Verification Results

All cells pass DRC, ERC and NCC without errors.

Postfabrication Test Plan

In order to easily test the production of this chip, it is desirable to add hardware to the design such as a ring oscillator to test the functionality of the i/o pads, or scan chains to the register file. Unfortunately, the top level of the FFT layout is too large to incorporate such features. However, it is possible to take advantage of the FFT algorithm in order to test the chip. As mentioned above, the 0th Fourier coefficient is equal to:

$$X[0] = \frac{1}{4}\left[x[0] + x[1] + x[2] + x[3]\right].$$

Therefore, in order to test the production of the input address decoder, register file and primary computation unit, the tester can tie the write enable high, set all of the inputs to zero but one and then examine the output to see if it is equal to one-fourth the non-zero input. It is important to note that all of the values must be initialized at startup, as there is no reset function implemented in the system. The output decoder can be tested by entering some simple test cases that guarantee all four output bytes to be different (such as x[0] = 1, x[1] = 2, x[2] = 3, x[3] = 4), then cycling through the outputs to ensure that none are the same.

Schematics

Leaf Cells

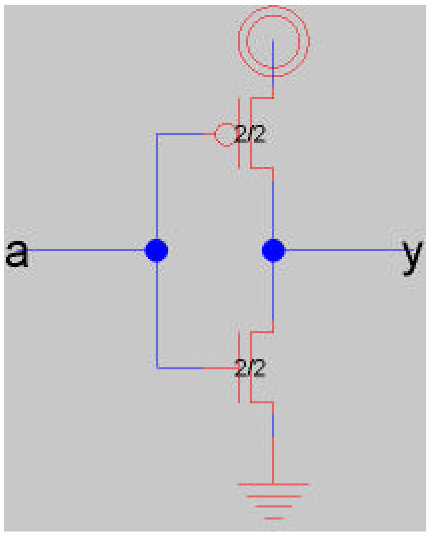


Figure 4: Inverter Schematic

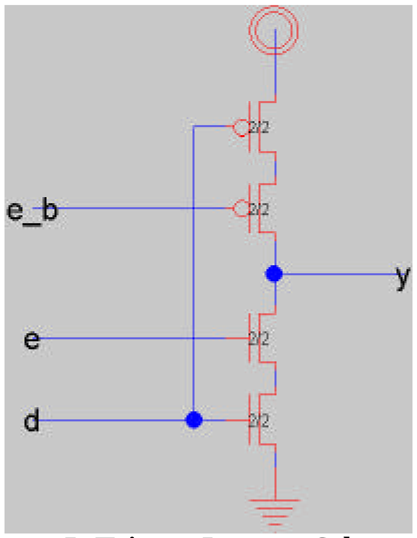


Figure 5: Tristate Inverter Schematic

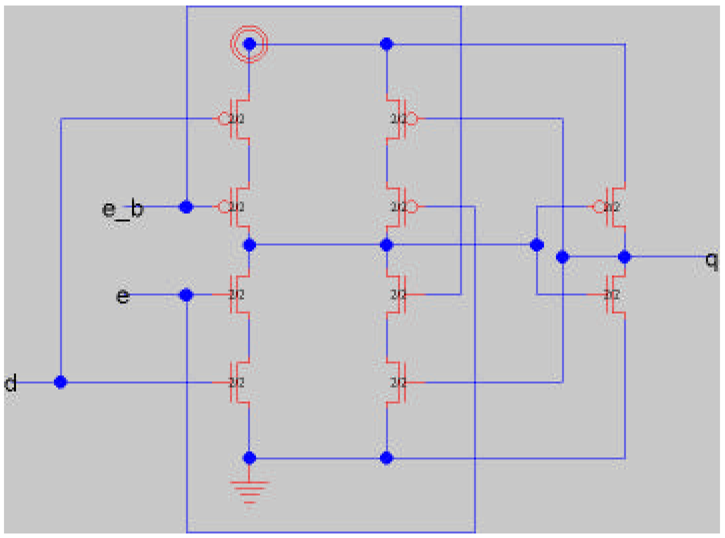


Figure 6: Latch Schematic

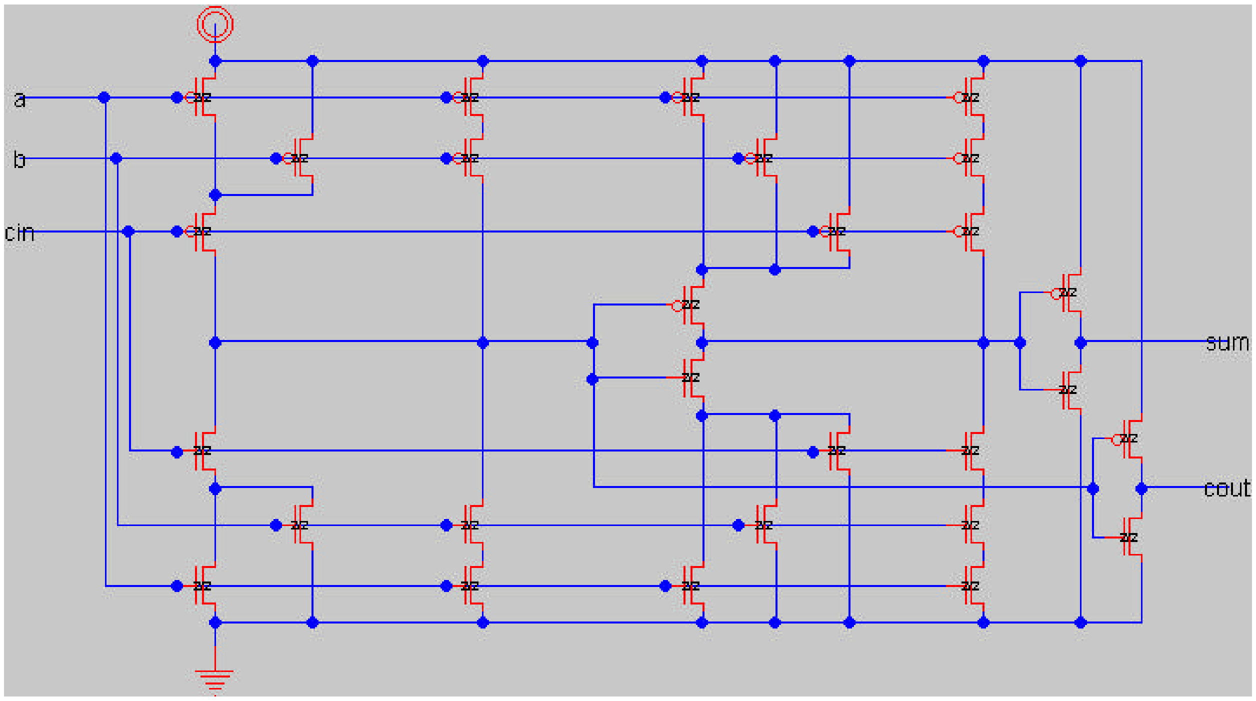


Figure 7: Full Adder Schematic

Non-Leaf Cells

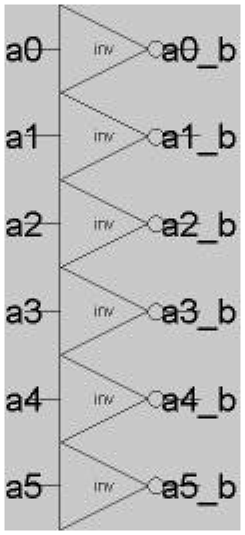


Figure 8: Inverte r Array Schematic

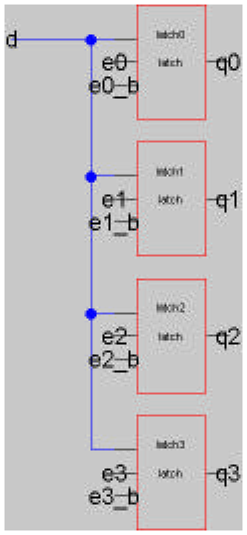


Figure 9: Register File Bitslice Schematic

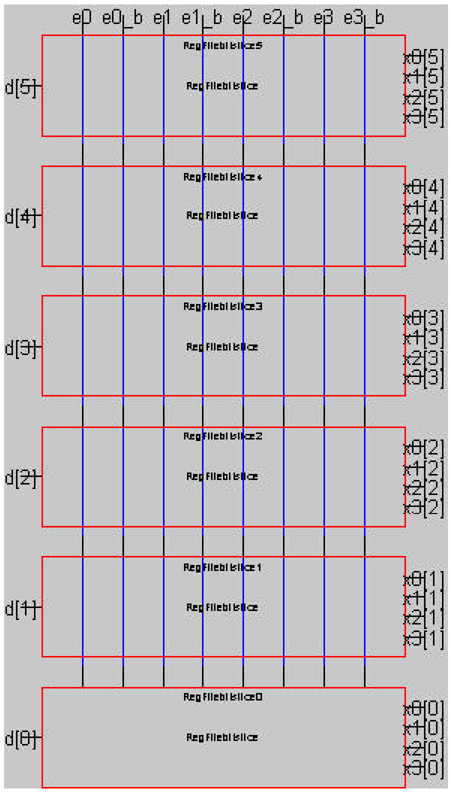


Figure 10: Register File Datapath Schematic

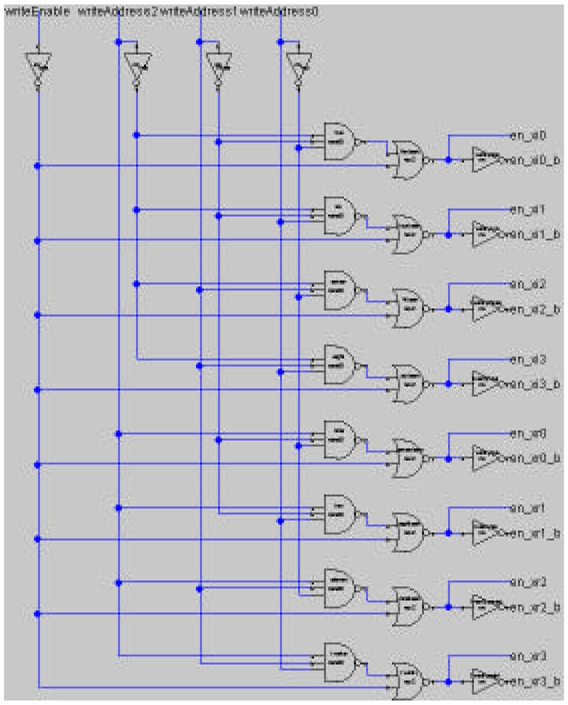


Figure 11: Input Address Decoder Schematic

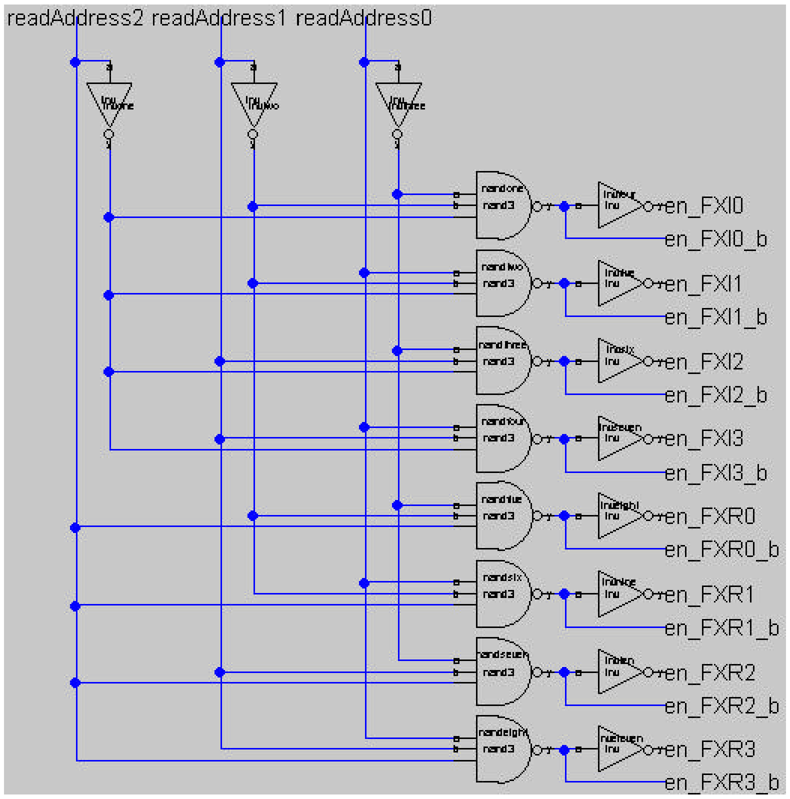


Figure 12: Output Address Decoder Schematic

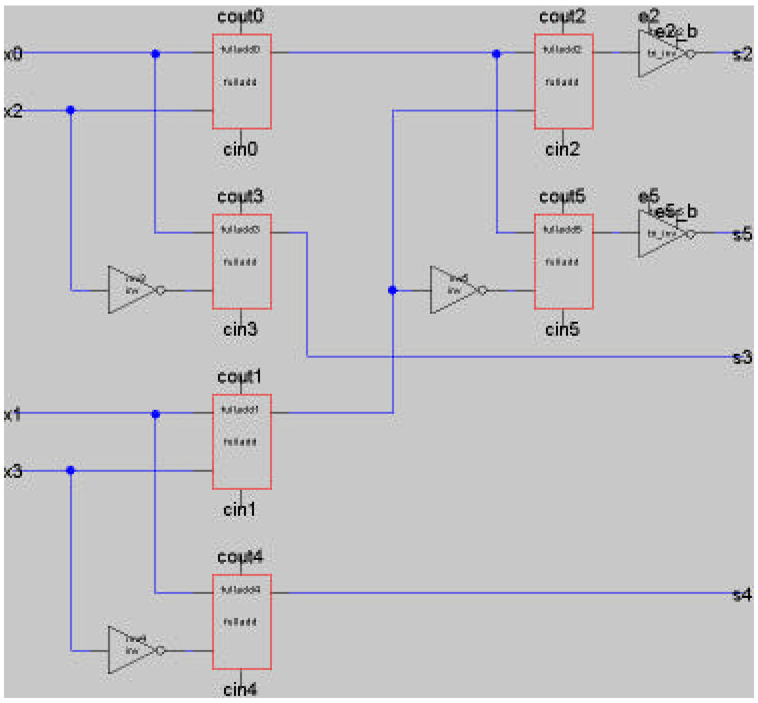


Figure 13: Primary Computation Unit Bitslice Schematic

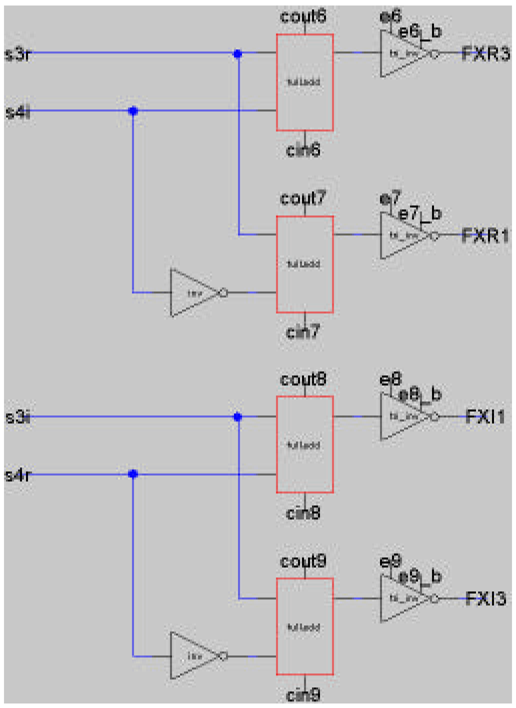


Figure 14: Secondary Computation Unit

Bitslice Schematic

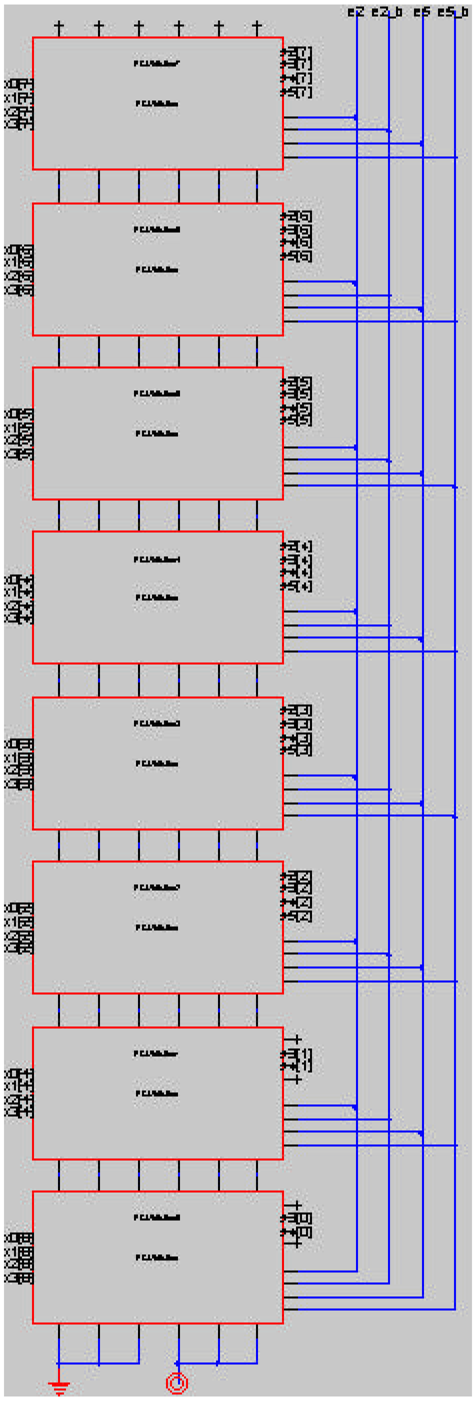


Figure 15: Primary Computation Unit Datapath Schematic

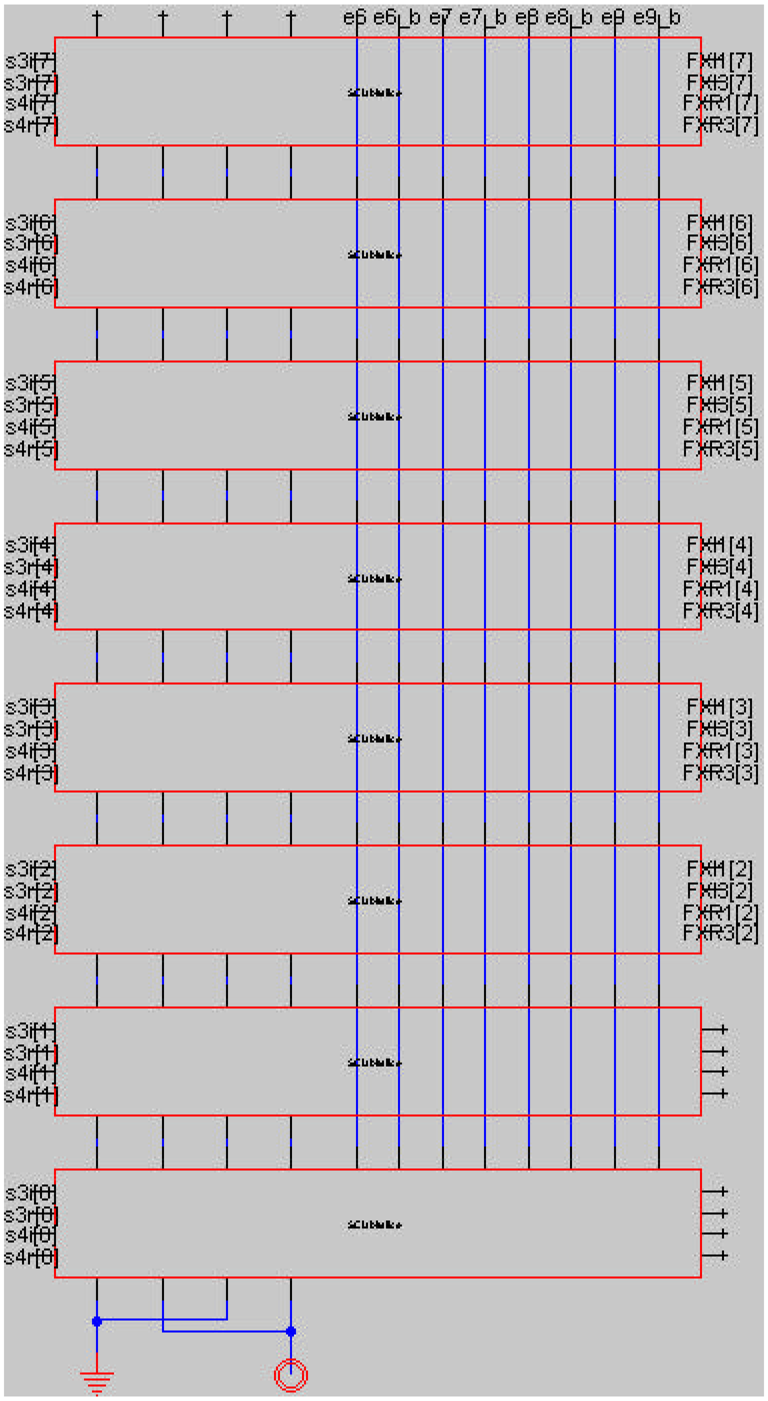


Figure 16: Secondary Computation Unit Datapath Schematic

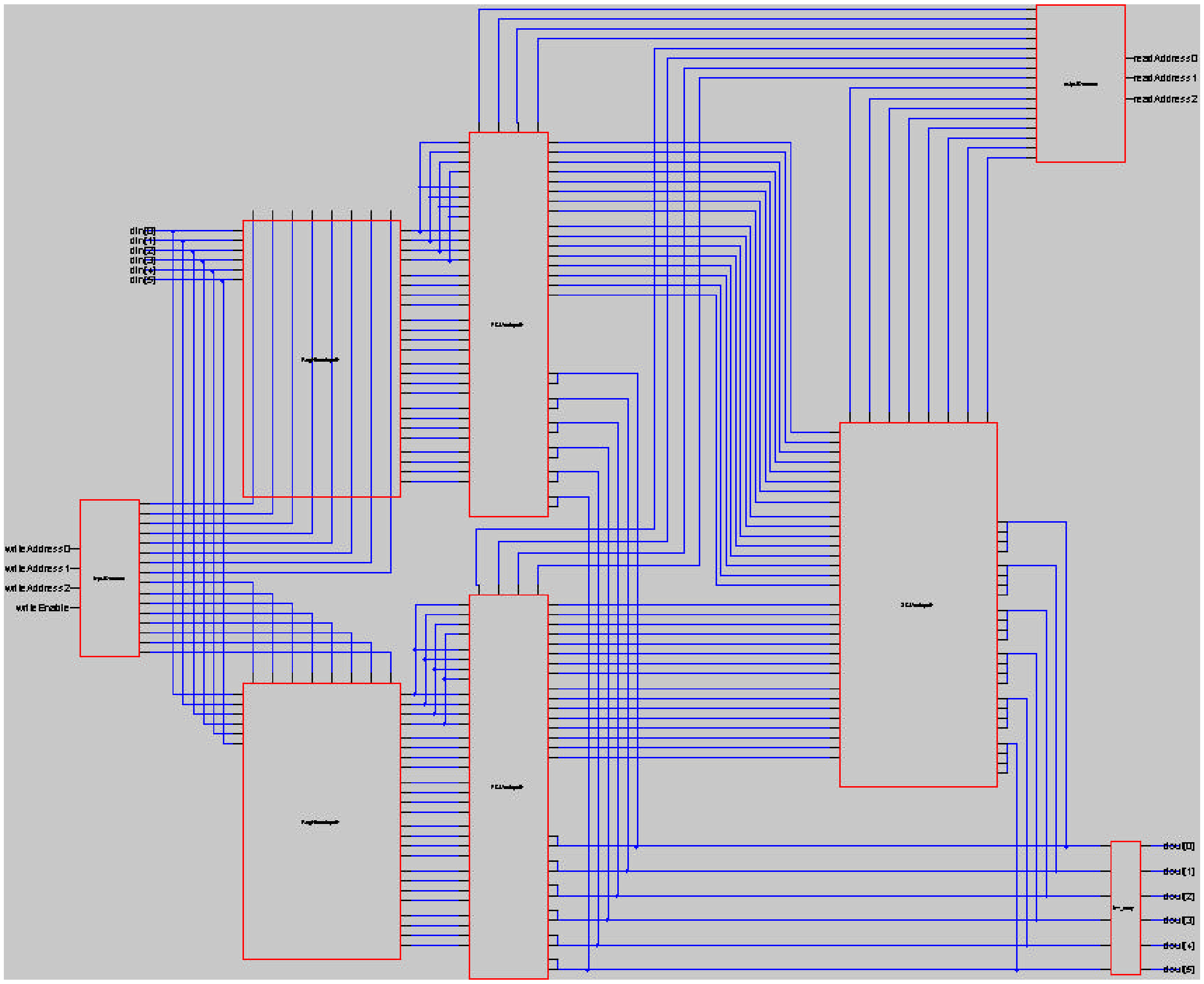


Figure 17: Top-Level Schematic

Layout

Leaf Cells

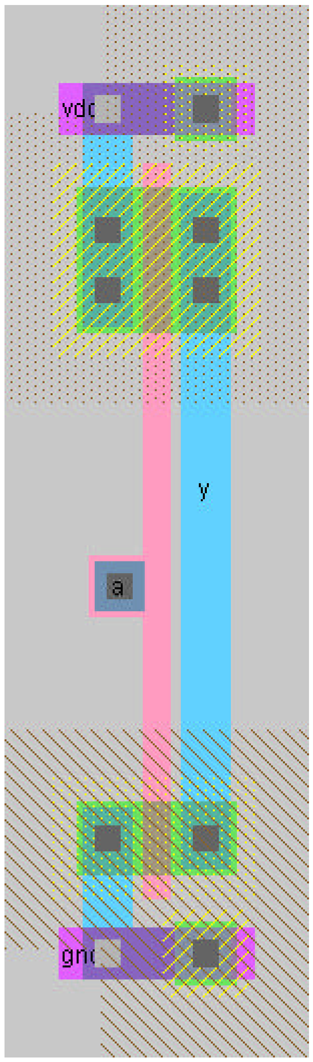


Figure 18: Inverter Layout

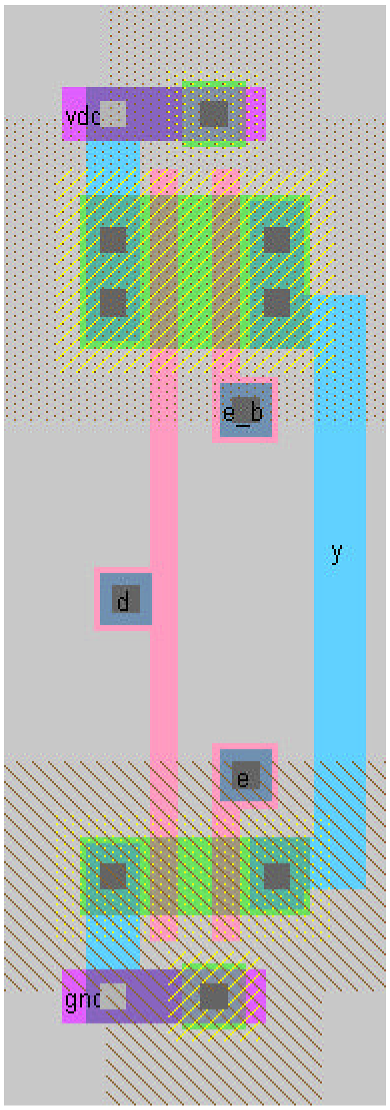


Figure 19: Tristate Inverter Layout

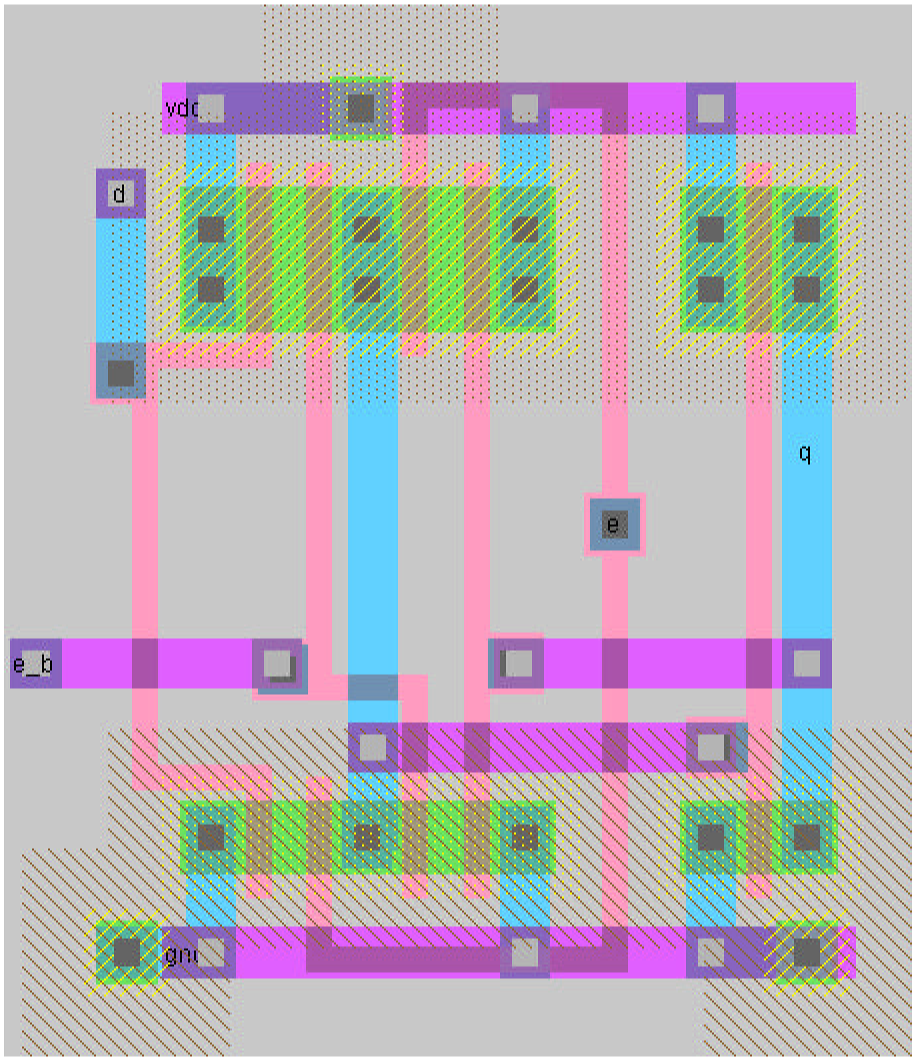


Figure 20: Latch Layout

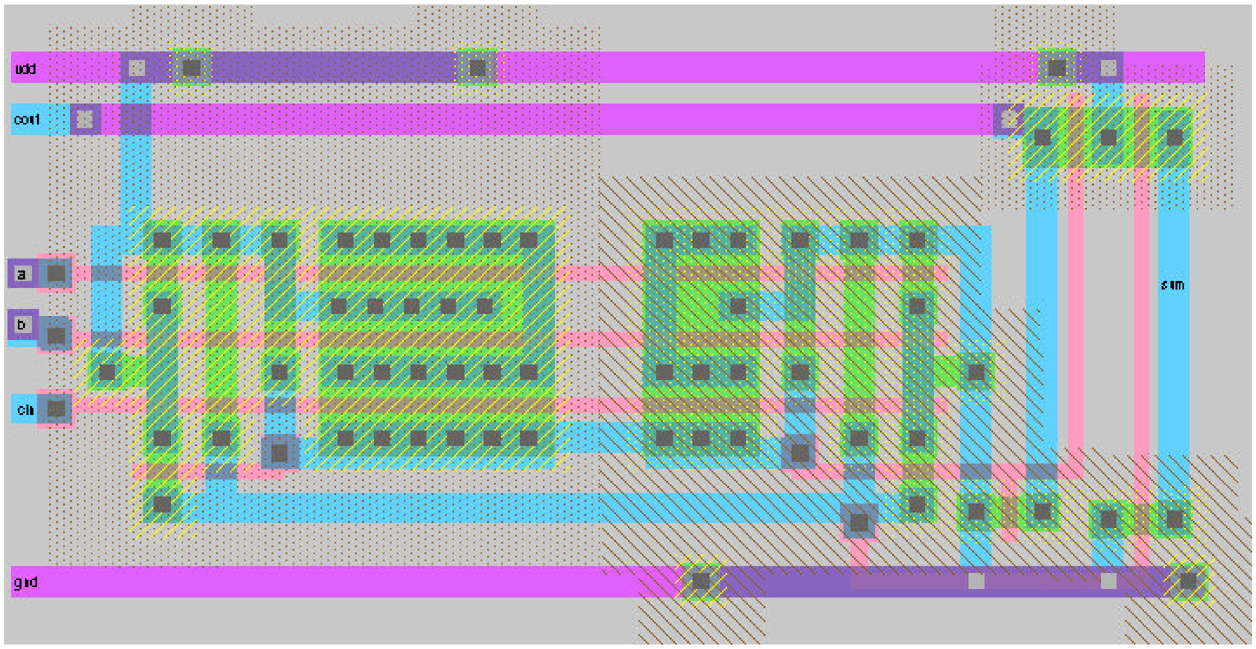


Figure 21: Full Adder Layout

Non-Leaf Cells

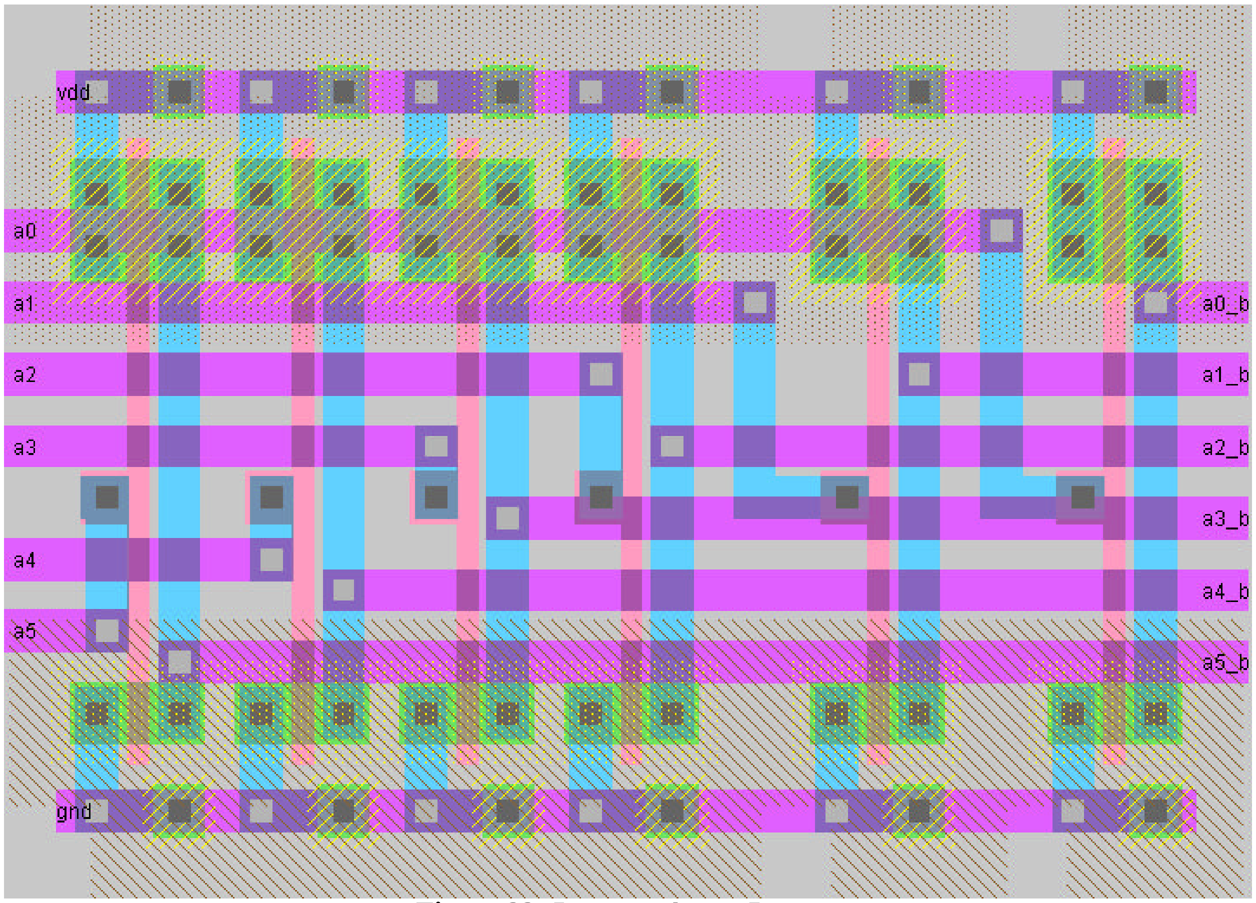


Figure 22: Inverter Array Layout

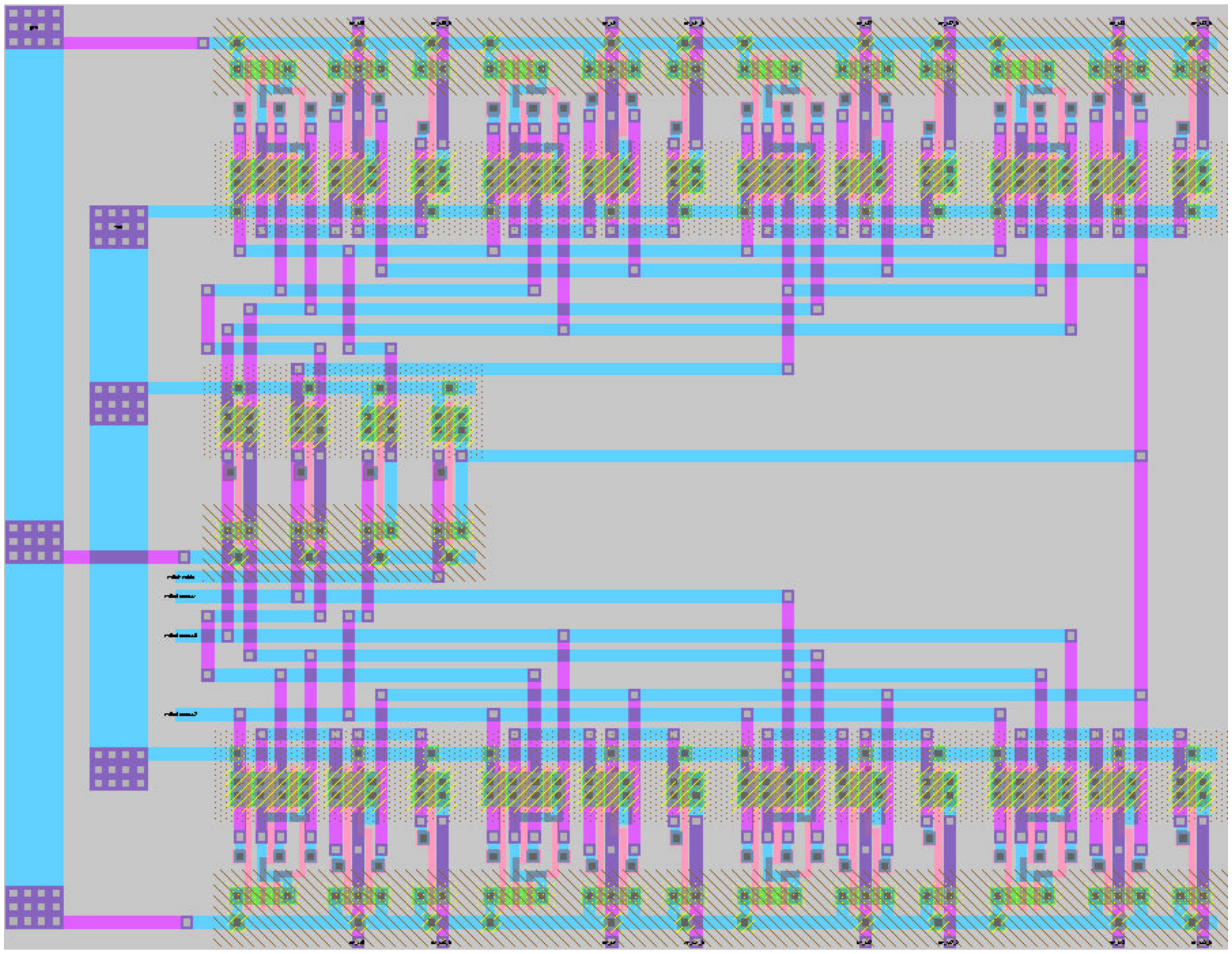


Figure 23: Input Address Decoder Layout

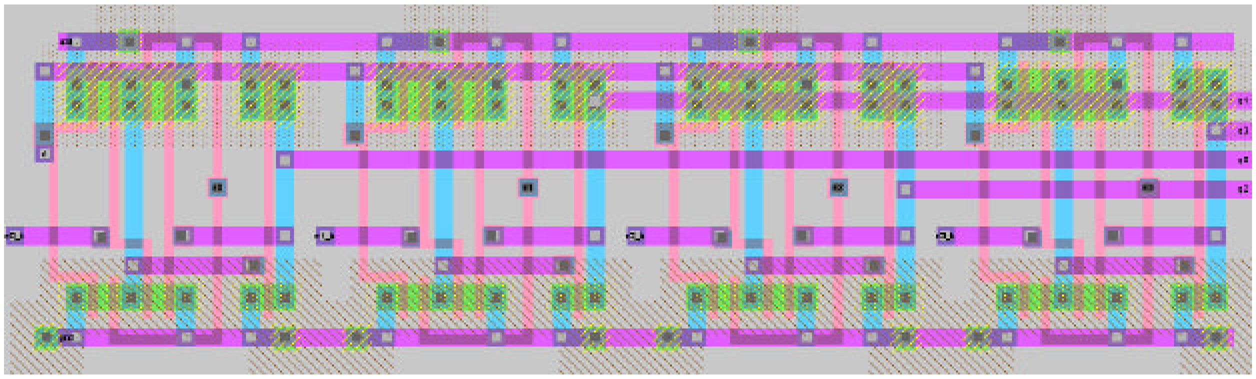


Figure 24: Register File Bitslice Layout

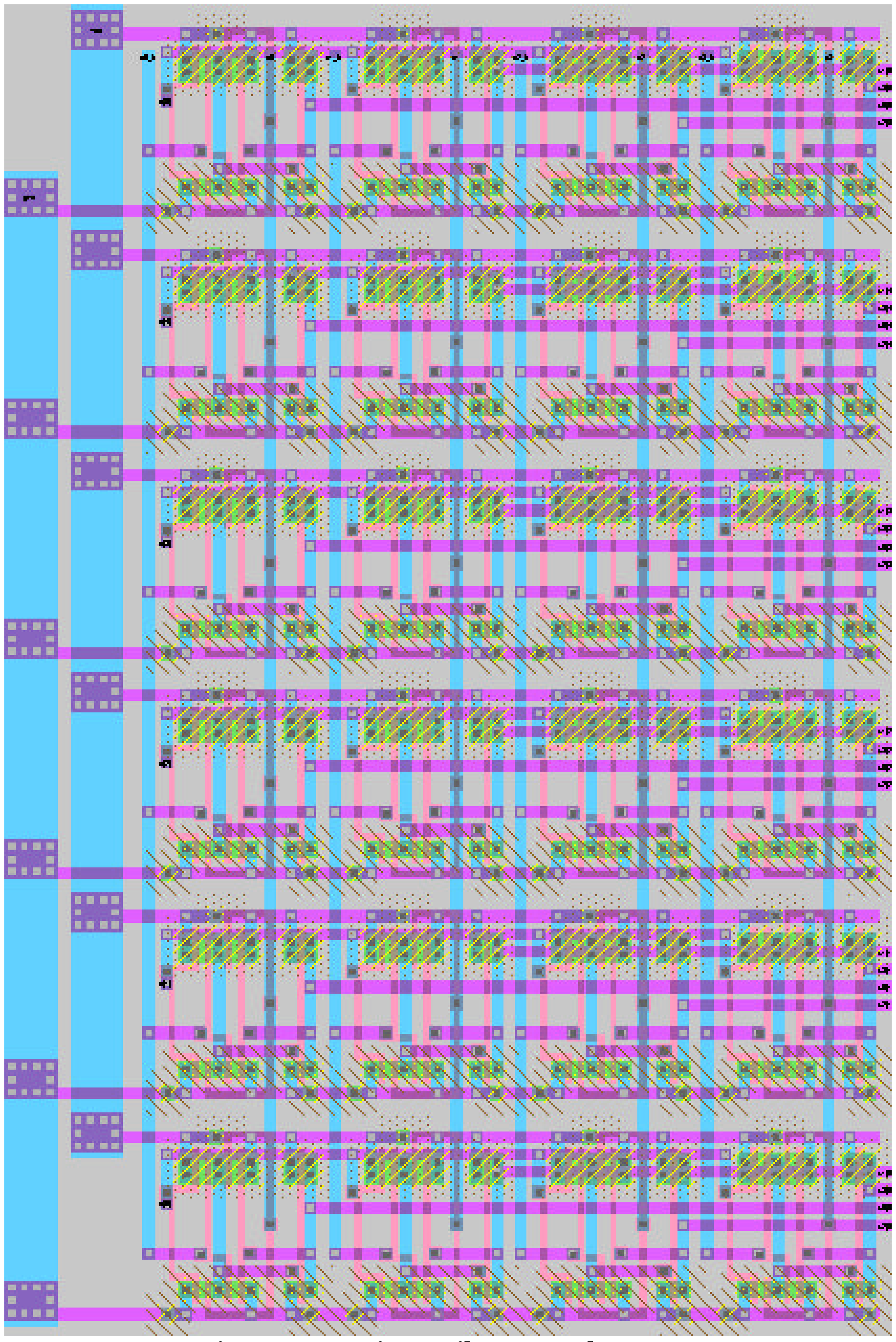


Figure 25: Regis ter File Datapath Layout

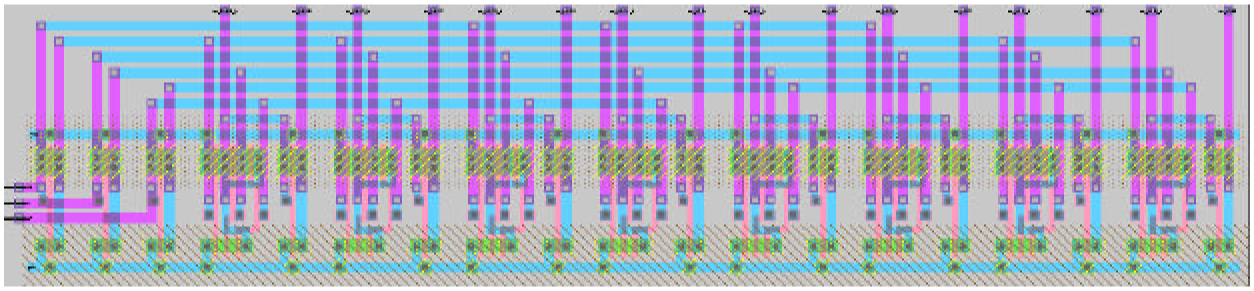


Figure 26: Output Address Decoder

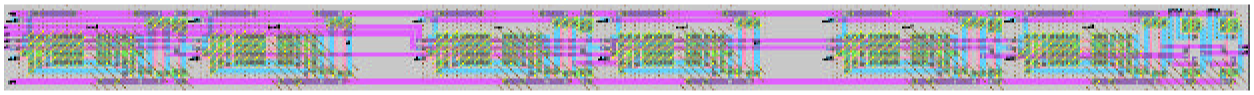


Figure 27: Primary Computation Unit Bitslice Layout

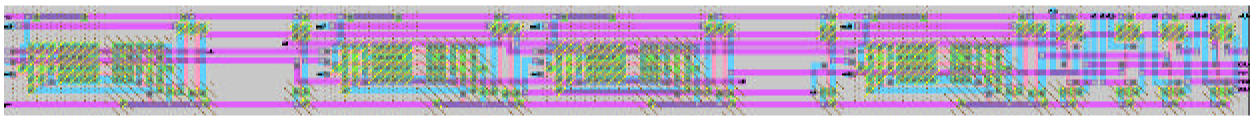


Figure 28: Secondary Computation Unit Bitslice Layout

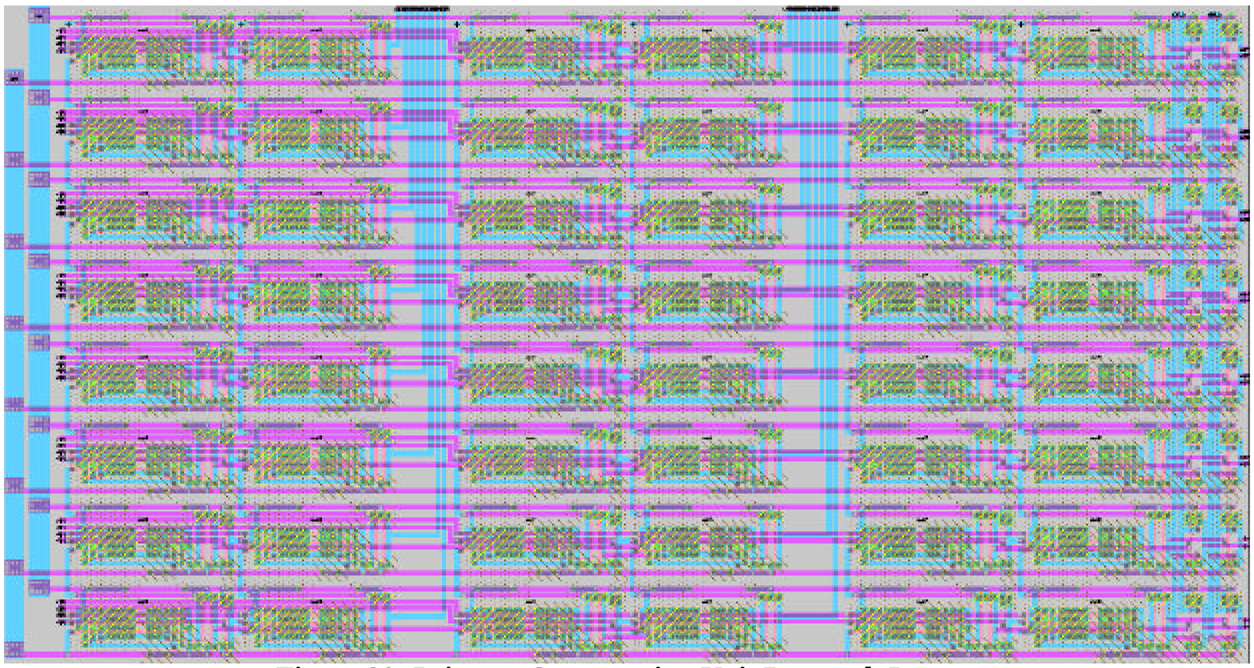


Figure 29: Primary Computation Unit Datapath Layout

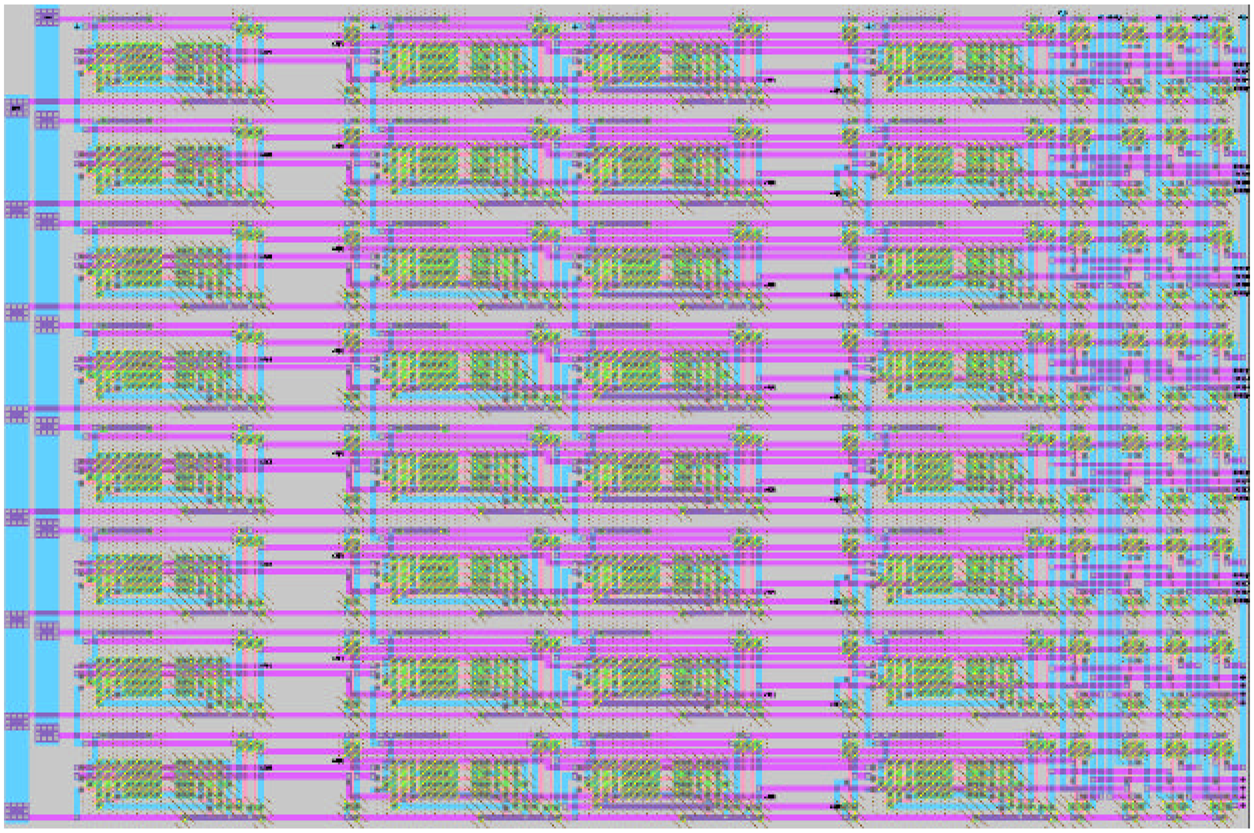


Figure 30: Secondary Computation Unit Datapath Layout

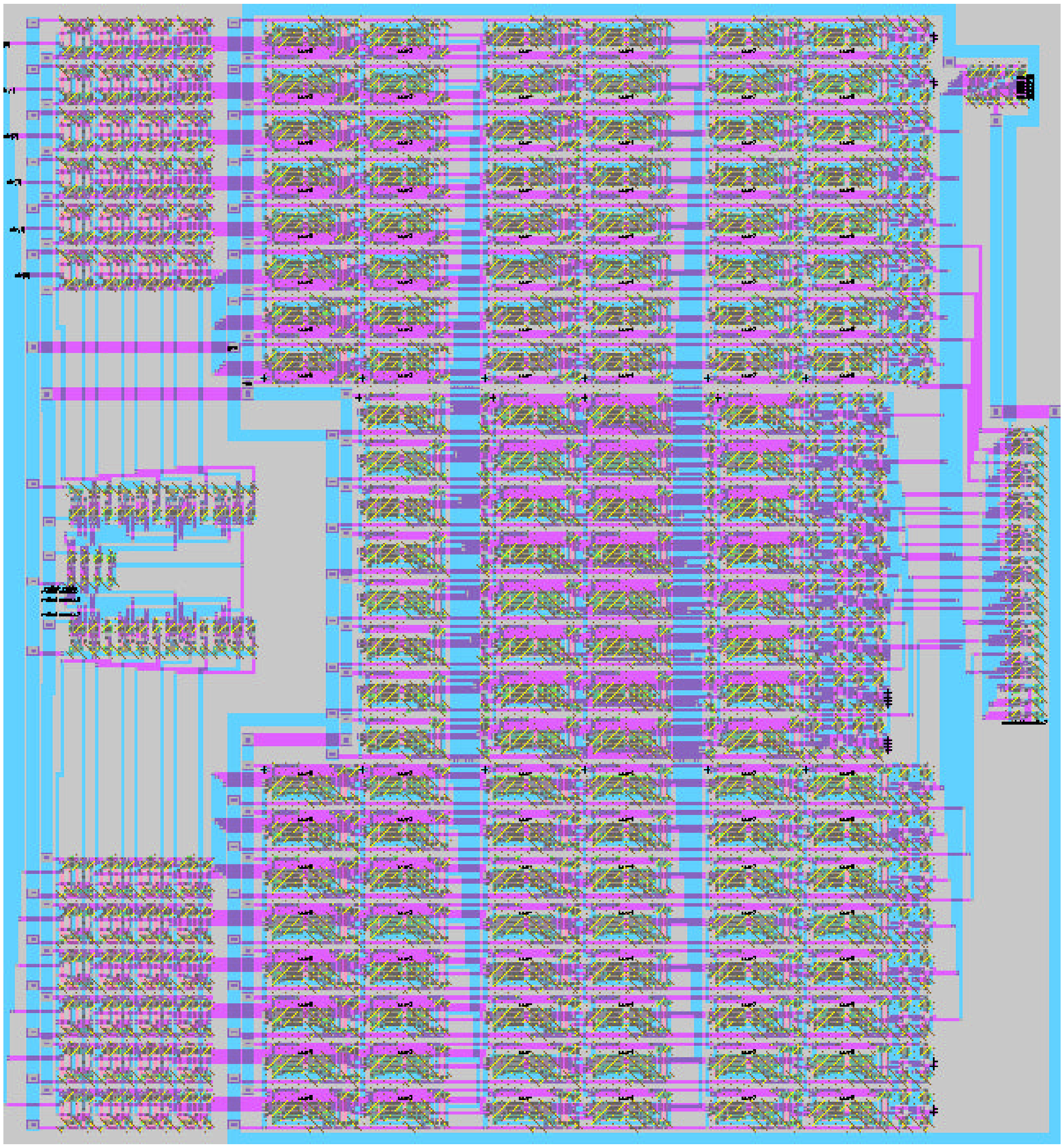


Figure 31: Top-Level Layout