CMOS Binary Full Adder

A Survey of Possible Implementations

Group 10:

Eren Turgay
Aaron Daniels
Michael Bacelieri
William Berry

Table of Contents

Key Terminology

AOI	Add-Or-Invert logic
BFA	Binary full-adder
CLA	Carry look-ahead
CMOS	Complementary Metal-Oxide Semiconductor (complementary usage of NMOS and PMOS transistors)
DRC	Dynamic ripple-carry
LALB	Look-ahead logic block
PFA	Partial full-adder
NAND	Negated logical AND
NMOS	n-type metal-oxide semiconductor
NOR	Negated logical OR
PMOS	p-type metal-oxide semiconductor
SRC	Static ripple-carry
XOR	Exclusive logical OR $\left(A \oplus B = A\overline{B} + \overline{A}B\right)$

Introduction

A basic survey of three different logic implementations of an 8-bit binary full adder is provided in this document. The three designs tested are the static ripple-carry, dynamic ripple-carry, and carry look-ahead architectures. Each will first be thoroughly explained, and then the suitability of each for use in a 200MHz RISC embedded processor will be evaluated with the aid of simulation data.

Design Architectures

The ultimate goal of a binary full-adder (BFA) is to implement the following truth table for each bit:

C_{in}	A	B	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1: Truth table for 1-bit adder slice

Logically, $C_{k+1} = A_k B_k + C_k (A_k + B_k)$ and $Sum_k = A_k \oplus B_k \oplus C_k$, where k is an integer 0 to n for an n-bit adder. Generally, adders of n-bits are created by chaining together n of these 1-bit adder slices. Three (3) adder designs have been examined: static ripple-carry, dynamic ripple-carry, and carry look-ahead.

Static Ripple-Carry (SRC) Implementation

The most basic and intuitive BFA is an SRC adder. This type of adder has the benefits of simplicity and asynchronicity. Asynchronicity means that the output of the adder can be accessed at any point during a clock cycle. This allows the adder to be used in two main styles of processors: 1) those that read/calculate data on the rising clock edge and write data on the falling clock edge and 2) those that read/calculate data during one or more full clock cycles and write data during one or more subsequent clock cycles. However, the largest drawback to an SRC adder is that is usually has the longest propagation time compared to other adder designs using the same process technology.

The particular design of SRC adder implemented in this discussion utilizes And-Or-Invert (AOI) logic [1]. AOI logic is a technique of using equivalent Boolean logic expressions to reduce the number of gates required for a particular expression. This, in turn, reduces capacitance and consequently propagation times. For this design, AOI logic has been applied to the calculation of the *Sum* bit:

$$Sum_k = A_k \oplus B_k \oplus C_k = (A_k + B_k + C_k)\overline{C}_{k+1} + A_k B_k C_k$$

Instead of using two (2) XOR gates to implement the *Sum* bit, the circuit takes advantage of the fact that $\overline{C}_{k+1}$ is already computed and uses fewer gates to calculate the rest of the expression.

The schematic of this design is shown in Figure 1**.** An 8-bit implementation using this design is shown in Figure 2.

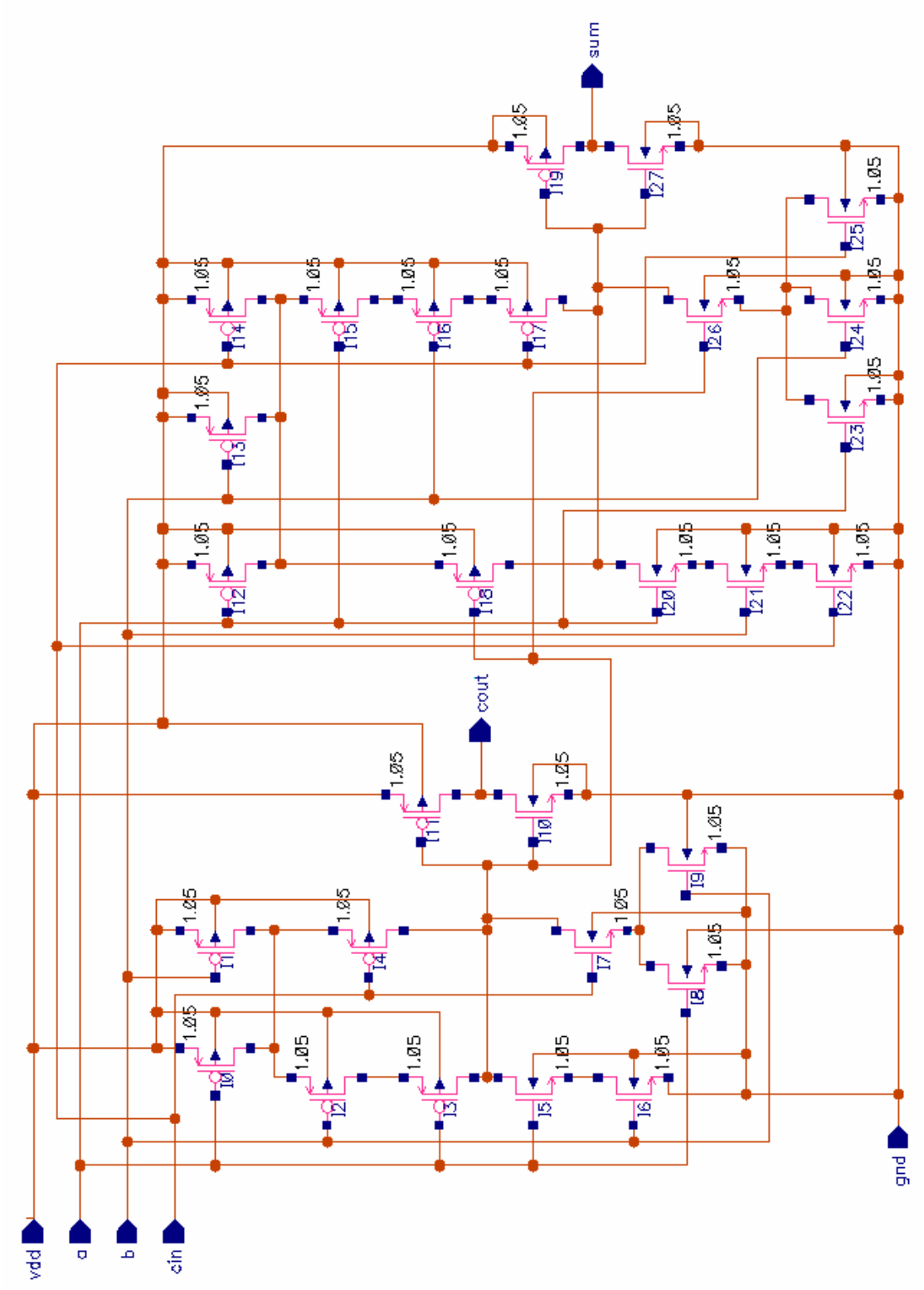

Figure 1: 1-bit SRC adder schematic

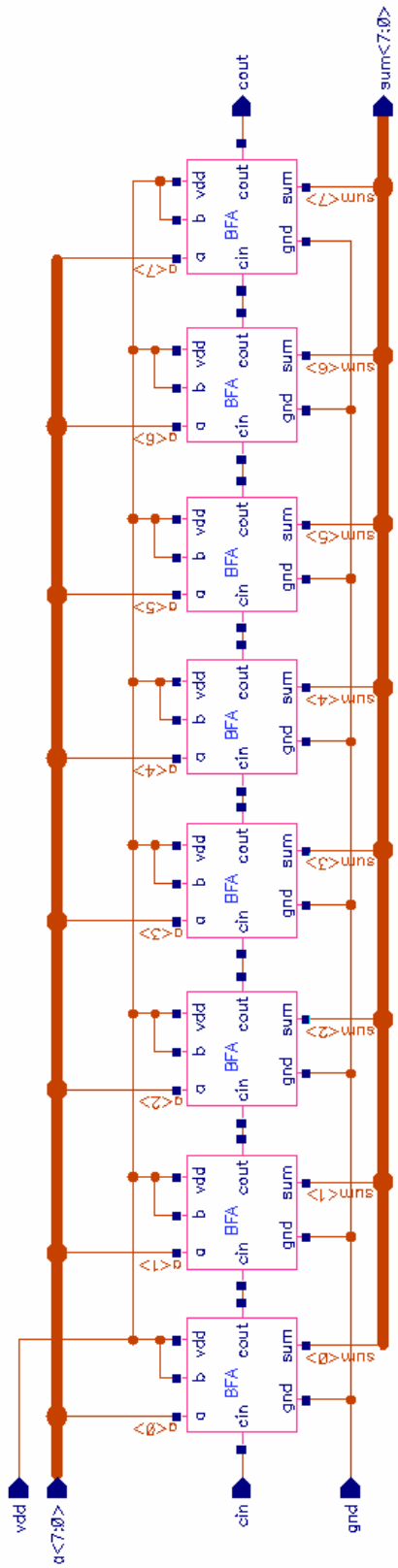

Figure 2: 8-bit SRC adder schematic

For this particular implementation of an *n*-bit SRC adder, the number of gates required is defined as $G_{SRC} = 28n$.

n			
2	**4**	**8**	**16**
56	112	*224*	448

Table 2: Gate counts for *n*-bit SRC adders

Dynamic Ripple-Carry (DRC) Implementation

The DRC adder is an advanced version of the SRC. Utilizing a clock allows the adder to take advantage of a technique known as precharging. This involves charging the sum and carry bits to an intermediate value (usually $V_{DD}/2$). This reduces the rise and fall time when a logic low or high is computed. The downside to this approach, however, is that the adder result is only available when the clock signal is high. Consequently, a latch is generally used to hold the data for the remainder of the clock cycle. Power consumption of the adder is also increased due to the precharging.

A processor designer has a few choices when choosing a clock to work with this type of adder. Since the result can only be calculated when the clock is high, the clock period must be at least twice as long as the adder propagation time. Depending upon the needs of the processor, anywhere from one (1) to *n* number of bits could be computed in one clock cycle.

The schematic for this design is shown in Figure 3**.** [2] An 8-bit implementation using this design is shown in Figure 4**.**

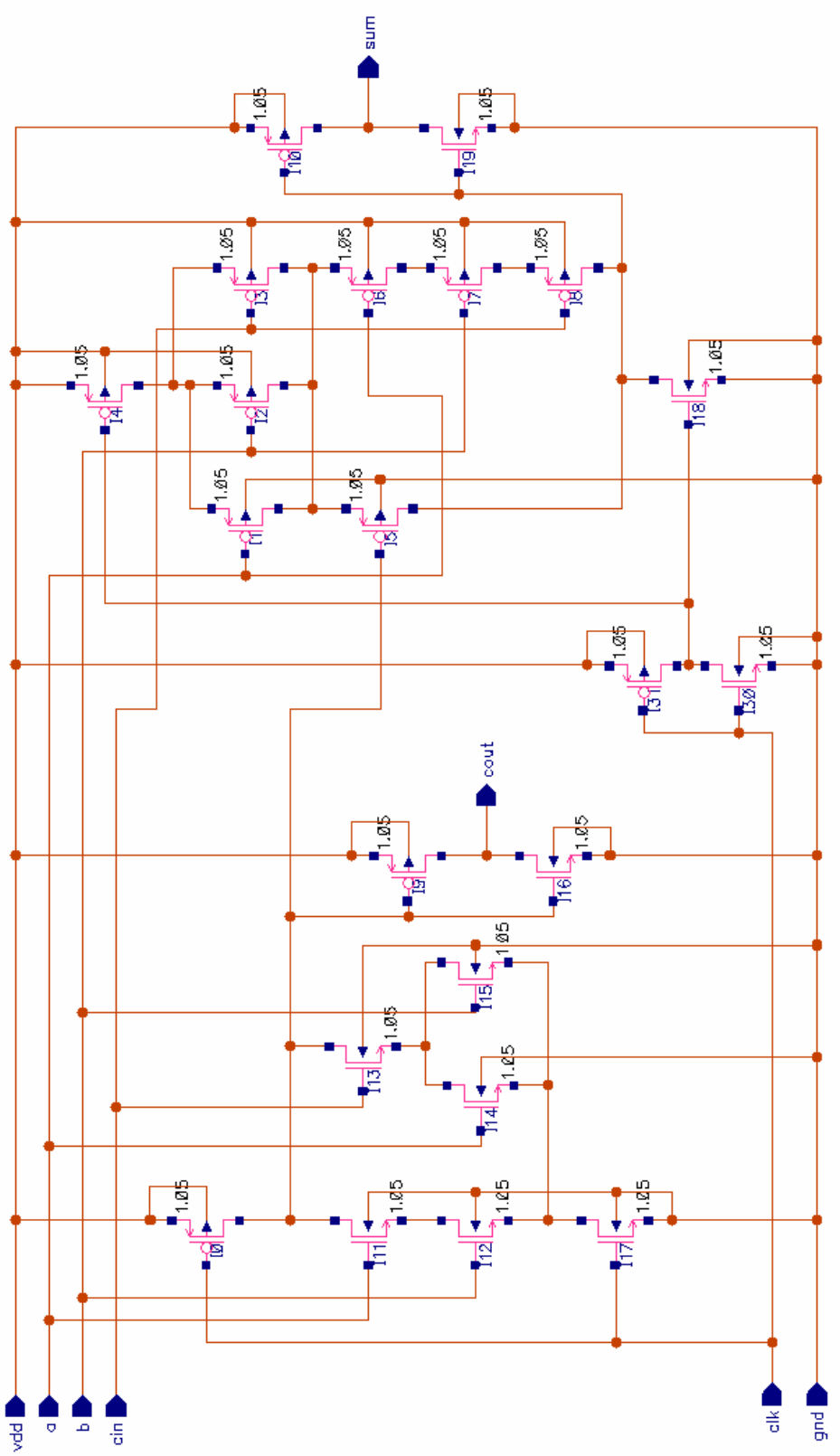

Figure 3: 1-bit DRC adder schematic

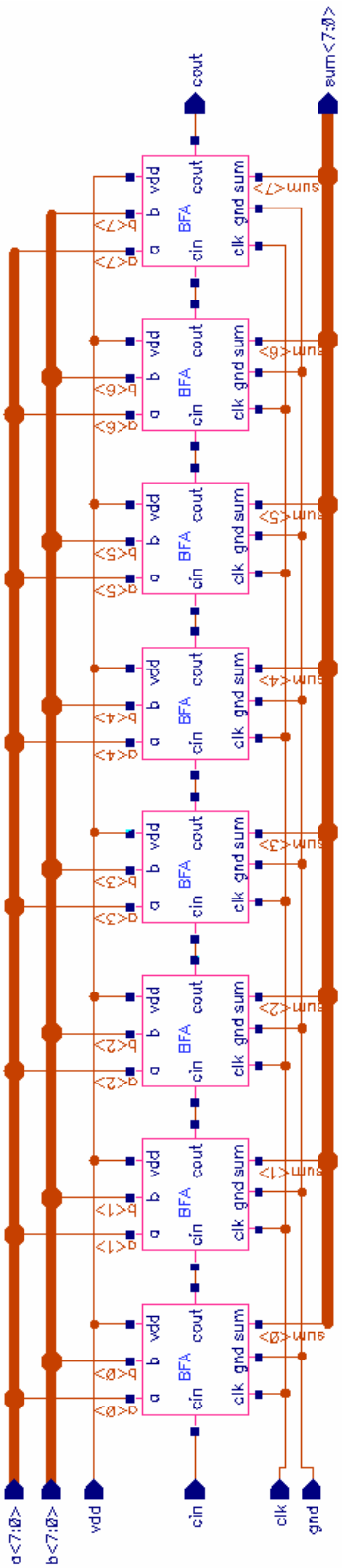

Figure 4: 8-bit DRC adder schematic

For this particular implementation of an *n*-bit dynamic ripple carry adder, the number of gates is defined as $G_{DRC} = 22n$.

		n	
2	**4**	**8**	**16**
44	88	*176*	352

Table 3: Gate counts for *n*-bit dynamic ripple-carry adders

Carry Look-Ahead (CLA) Implementation

CLA adders take advantage of computational parallelization at the cost of increased complexity and power consumption. This parallelization yields significant decreases in propagation time. Intermediate terms, called propagate and generate bits, are used to calculate sum and carry bits. The logic equations for the calculations are as follows:

$$P_k = A_k \oplus B_k$$
$$G_k = A_k B_k$$
$$Sum_k = A_k \oplus B_k \oplus C_k = P_k \oplus C_k$$
$$C_{k+1} = C_k P_k + G_k$$

A CLA adder uses two fundamental logic blocks – a partial full-adder (PFA) and a look-ahead logic block (LALB). The PFA computes the propagate, generate and sum bits. The LALB uses the propagate and generate bits from *m* number of PFAs to compute each of C_1 through C_m carry bits, where *m* is the number of look-ahead bits. For maximum performance, *m* is equal to *n*. However, in practice *n* is usually a multiple of *m*, resulting in a hybrid of look-ahead and ripple logic.

The carry look-ahead adder was implemented using eight (8) 1-bit PFAs and two (2) 4-bit LALBs. A 4-bit LALB design was chosen as a balance between the smaller area and lower power of a 2-bit block and the speed of a full 8-bit block. The carry equations for a 4-bit LALB are as follows:

$$C_1 = C_0 P_0 + G_0$$
$$C_2 = C_1 P_1 + G_1 = C_0 P_0 P_1 + G_0 P_1 + G_1$$
$$C_3 = C_2 P_2 + G_2 = C_0 P_0 P_1 P_2 + G_0 P_1 P_2 + G_1 P_2 + G_2$$
$$C_4 = C_3 P_3 + G_3 = C_0 P_0 P_1 P_2 P_3 + G_0 P_1 P_2 P_3 + G_1 P_2 P_3 + G_2 P_3 + G_3$$

These logic functions were implemented using parallel cascading NAND gates (see Figure 6).

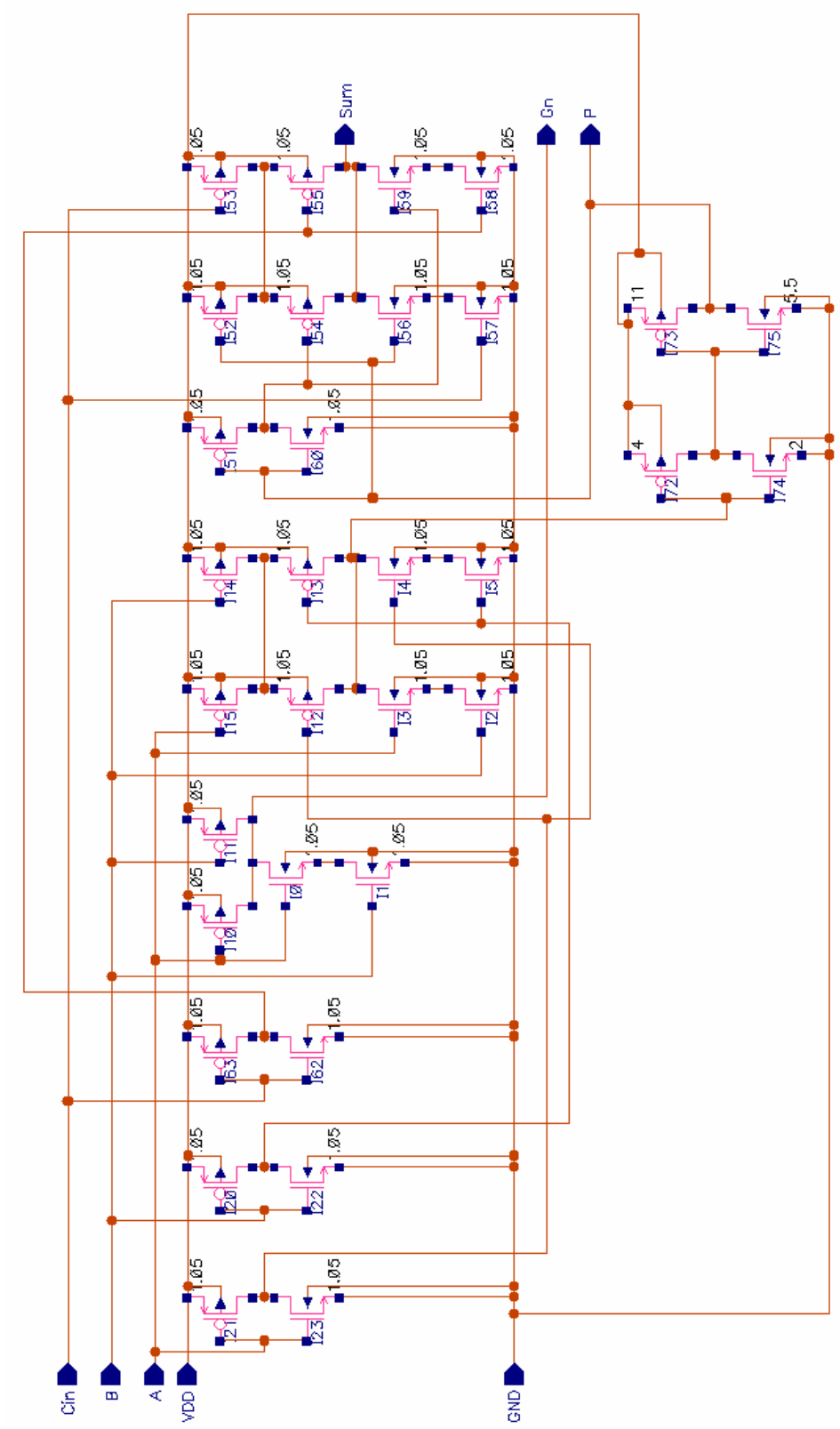

Figure 5: 1-bit PFA schematic

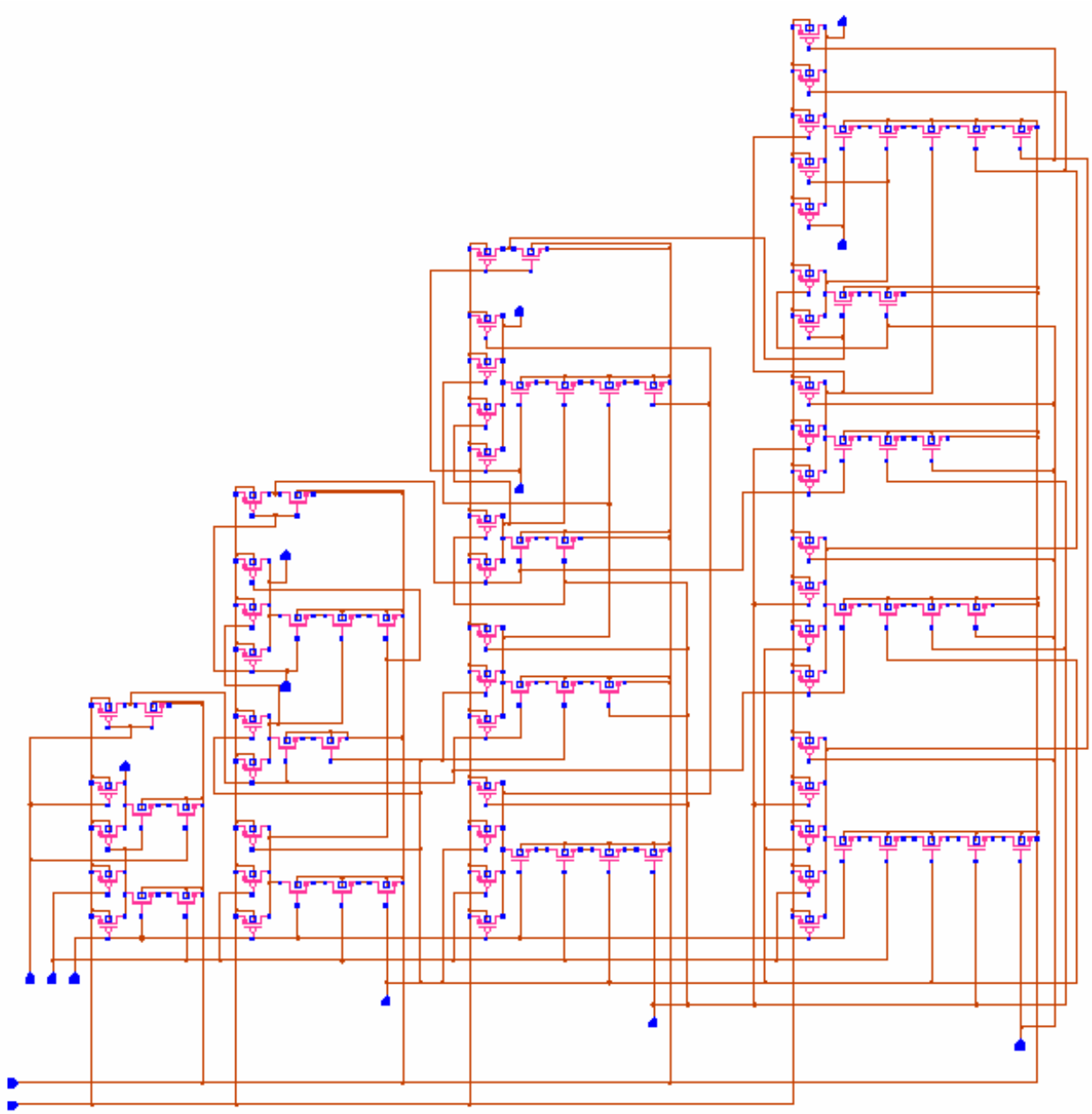

Figure 6: 4-bit LALB schematic

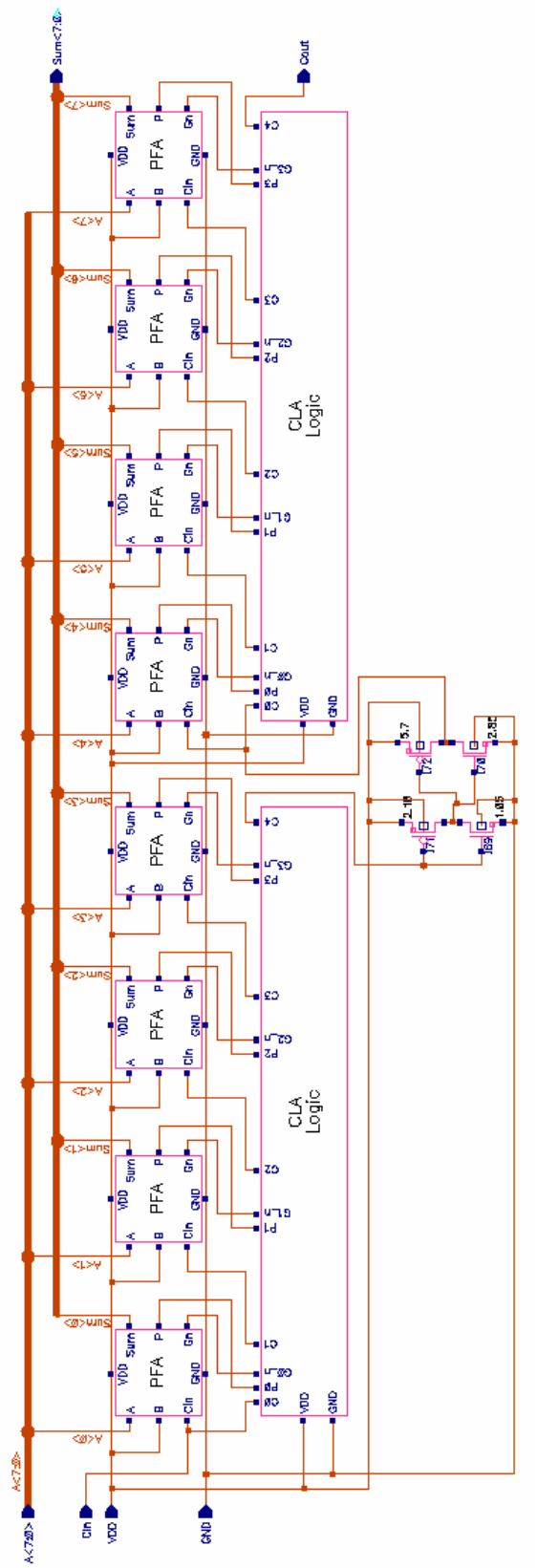

Figure 7: 8-bit CLA schematic

While ripple-carry adders scale linearly with *n* number of adder bits, carry look-ahead adders scale roughly with $m^2 n$. For optimum performance, *m* is equal to *n*. However, in practice *n* is usually a multiple of *m*.

For this particular implementation of an *n*-bit carry look-ahead adder with *m*-bit look-ahead logic, the number of gates is defined as follows:

$$G_{CLA} = 32n + \lceil n/m \rceil \left(\sum_{k=1}^{m} [(k+1)(k+4)] - 2 \right) + 4(\lceil n/m \rceil - 1)$$

where $32n$ is the number of gates in a 1-bit PFA, $\lceil n/m \rceil$ is the number of carry look-ahead logic blocks needed, $\sum_{k=1}^{m} [(k+1)(k+4)] - 2$ is the number of gates in a logic block, and $4(\lceil n/m \rceil - 1)$ is the number of gates for the buffers between logic blocks.

		m			
		2	**4**	**8**	**16**
n	**2**	90			
	4	184	222		
	8	372	*448*	670	
	16	748	900	1344	2750

Table 4: Gate counts for various sized CLA adders

Transistor sizing to optimize performance

A number of attempts were made, particularly on the carry look-ahead adder, to optimize performance through proper transistor sizing. In general, it was found that minimizing the transistor sizes also minimized the propagation delays. However, in cases where just a few transistors were used to drive a large number of outputs, increasing the width to length ratios of the driving gates often increased performance. This technique was used on each of the NAND gates driving the cout pins in the look-ahead logic block. Increasing the width of the NMOS transistors in these NAND gates to double the minimum value while keeping the PMOS transistors minimum optimized the performance.

When the number of gates driven by a single pair of transistors was particularly large, a buffer was used to decrease the output delays. This proved beneficial in two cases; once when driving the P0 (propagate) signal in the PFA logic, and once when driving the C0 input to the second look-ahead logic block. In both cases, a buffer with only two stages was used, and the sizes of the first inverters were kept to a minimum in order to minimize the input capacitance to the first stage. The width of the second stage was made to be very nearly *e* times the first, in keeping with the theory for minimum buffer delay.

Design Validation

All of the designs outlined above were sufficiently tested in order to prove that the correct results were produced for all inputs. For both the static and dynamic implementations of the ripple carry adder, the one-bit slices were first exhaustively tested and confirmed to produce outputs matching the truth table shown in Table 1. The full 8-bit chained adders were not exhaustively tested, however, since testing all of the possible 2^17 input combinations was simply not feasible. Instead, selected inputs and worst-case scenarios were chosen and tested. The primary concern was to ensure that all of the carry bits were connected to the next bit slice properly and that there were no loading effects, since each bit slice had already been proven correct.

The validation of the carry look-ahead adder was slightly more involved than the other designs. Each PFA slice was first exhaustively tested and confirmed to produce the correct propagate, generate and sum bits. Additionally, the LALB was selectively tested to validate the four carry bits. Finally, the full 8-bit adder was tested using selected inputs and worst-case scenarios.

All three designs were successfully validated.

Design Performance

Performance Across Corners

To assess the theoretical range of conditions under which each design would function properly, the performance of each implementation was evaluated at the three primary process corners: fast-fast, typi-typi, and slow-slow.

Speed

The Shmoo plots shown in Figure 8-10 display the maximum allowable operating frequency of each design at the three primary process corners. At each corner, VDD is varied 10% from its nominal value of 1.8V and the temperature is varied from –30–150 °C. It is assumed that the target operating frequency is 200MHz. Values below this are colored red to indicate failure to meet this condition, while green indicates success.

Temperature	**Carry Lookahead Frequency (MHz) Fast-Fast**						
(C) 150	485.20	555.56	628.14	701.75	775.19	848.18	919.96
120	439.37	511.51	586.51	662.69	739.64	816.33	891.27
90	392.77	466.42	543.77	623.05	703.23	783.09	862.07
60	345.30	420.17	499.50	581.73	665.34	749.06	831.26
30	296.74	372.16	453.51	538.50	625.78	713.27	800.00
0	246.12	321.23	404.20	492.61	583.43	675.22	766.28
-30	193.99	267.31	351.12	442.28	537.35	633.71	730.46
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Temperature	**Carry Lookahead Frequency (MHz) Typi-Typi**						
(C) 150	247.73	294.05	345.75	399.23	452.29	510.22	566.89

120	215.89	264.52	313.93	368.78	426.68	484.75	544.25
90	184.60	231.09	281.99	336.40	395.22	457.00	517.72
60	153.12	198.70	249.51	304.97	364.60	428.05	489.31
30	121.97	165.56	215.86	270.92	332.93	397.02	463.65
0	91.78	129.97	179.90	235.13	297.69	362.43	431.69
-30	63.36	97.87	142.97	196.64	258.48	325.46	394.59
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Temperature	Carry Lookahead (MHz) Slow_Slow						
(C) 150	94.16	119.75	149.25	182.55	218.96	257.86	298.60
120	75.47	99.40	127.89	160.72	197.28	236.80	278.55
90	58.00	79.81	106.73	138.68	175.01	214.92	257.47
60	42.19	61.20	85.91	116.33	152.00	191.90	235.07
30	28.46	44.05	65.70	93.81	128.04	167.39	210.88
0	17.34	29.06	46.75	71.48	103.23	141.28	184.43
-30	9.20	16.94	29.95	49.98	77.88	113.43	155.38
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Figure 8: Carry lookahead shmoo plots

Temperature	Dynamic Ripple Carry Adder Frequency (MHz) Fast_Fast						
(C) 150	285.06	330.58	378.36	427.90	478.93	530.79	582.75
120	253.68	299.22	347.71	398.41	450.86	504.29	558.35
90	218.77	263.50	311.72	362.32	415.11	469.04	523.56
60	177.40	219.44	265.32	314.37	365.63	418.59	472.37
30	145.37	185.87	231.11	280.03	331.56	385.06	439.75
0	118.05	157.51	202.72	252.40	305.34	360.36	416.84
-30	91.24	128.83	173.76	224.22	278.63	335.68	394.63
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Temperature	Dynamic Ripple Carry Adder Frequency (MHz) Typi-Typi						
(C) 150	140.96	169.72	201.21	235.07	271.00	308.55	347.22
120	117.73	145.12	175.53	208.51	243.84	280.82	319.08
90	92.51	117.54	146.05	177.49	211.51	247.52	285.06
60	74.52	98.23	125.87	157.04	191.13	227.43	265.60
30	58.34	80.45	107.23	138.14	172.38	209.25	248.39
0	43.20	63.17	88.50	118.74	153.00	190.40	230.41
-30	29.37	46.45	69.54	98.52	132.40	170.15	211.01
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Temperature	Dynamic Ripple Carry Adder Frequency (MHz) Slow_Slow						
(C) 150	48.38	62.19	78.55	97.37	118.54	141.74	166.61
120	36.51	48.78	63.73	81.43	101.80	124.41	148.94
90	27.40	38.27	52.08	68.97	88.81	111.33	135.91
60	19.60	28.89	41.36	57.21	76.39	98.62	123.33
30	13.04	20.52	31.25	45.68	63.94	85.62	110.20

0	7.86	13.38	22.00	34.55	51.28	72.05	96.25
-30	4.16	7.77	14.03	24.12	38.79	58.04	81.43
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Figure 9: Dynamic ripple carry Shmoo plots

Temperature	Static Ripple Carry Adder Frequency (MHz) Fast_Fast						
(C) 150	215.10	243.07	271.96	301.20	331.02	361.66	393.24
120	199.04	227.48	257.14	287.44	318.88	350.39	382.70
90	182.48	211.46	241.66	272.93	305.34	337.84	371.89
60	164.69	194.48	225.48	257.53	291.21	324.68	359.71
30	146.16	176.18	207.64	240.50	274.95	310.66	347.10
0	126.37	156.08	188.29	222.52	257.60	294.64	332.12
-30	104.59	133.96	166.42	201.17	237.76	276.01	315.86
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Temperature	Static Ripple Carry Adder Frequency (MHz) Typi_Typi						
(C) 150	118.25	138.31	159.57	181.95	205.21	229.20	250.06
120	104.43	124.47	145.94	168.69	192.49	213.86	234.58
90	90.33	110.14	131.72	154.73	176.37	196.89	218.05
60	75.93	95.33	116.66	138.08	158.03	178.79	200.40
30	61.31	79.74	100.00	118.61	138.43	159.34	181.39
0	46.69	63.65	80.06	98.04	117.44	138.37	160.72
-30	32.49	45.13	59.59	76.22	94.88	115.55	138.08
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Temperature	Static Ripple Carry Adder Frequency (MHz) Slow_Slow						
(C) 150	49.80	63.33	78.86	96.06	111.93	132.01	147.12
120	40.05	52.77	67.75	84.75	99.70	118.62	133.69
90	30.88	42.50	56.69	73.15	87.72	104.53	119.57
60	22.50	32.70	45.77	61.39	74.46	89.77	104.70
30	15.21	23.62	35.11	49.43	61.61	74.46	88.97
0	9.29	15.63	25.08	37.48	48.71	58.51	72.41
-30	4.98	9.15	16.14	26.13	35.97	42.19	55.04
VDD (V)	1.62	1.68	1.74	1.8	1.86	1.92	1.98

Figure 10: Static ripple carry adder Shmoo plots

The Shmoo plots above clearly demonstrate the supremacy of the carry lookahead implementation over the others in terms of speed, since there is a significantly larger range of values over which the design exceeds the 200MHz mark. It is interesting to note the extremely large range of possible values that the maximum frequency takes on in these plots. For instance, a carry lookahead adder with a fast_fast process, temperature of 150°C, and VDD of 1.98V has a maximum frequency of over 900MHz, while one with a slow_slow process, temperature of -30°C, and VDD of 1.62 only has a maximum

frequency of about 9MHz. This is a difference of roughly 100 fold from corner to corner. The other designs also show this wide variation.

Perhaps an even more lucid visualization of the differences in speed between the various implementations is given by the graphs in Figure 11, Figure 12, and Figure 13, which show how the maximum frequency varies with process, temperature, and VDD.

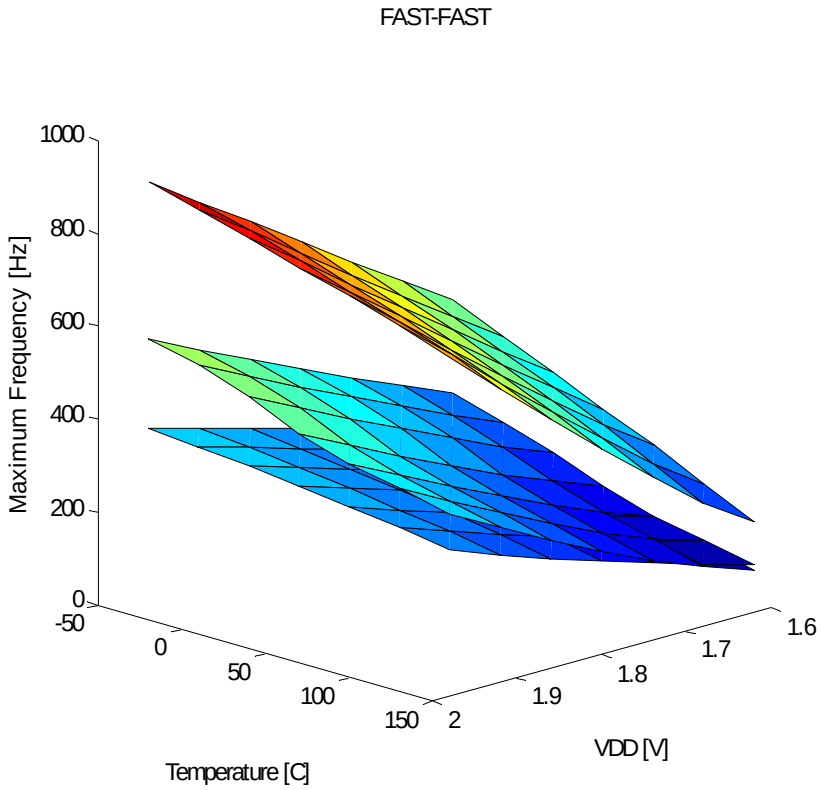

Figure 11: Max frequency of designs for fast_fast process

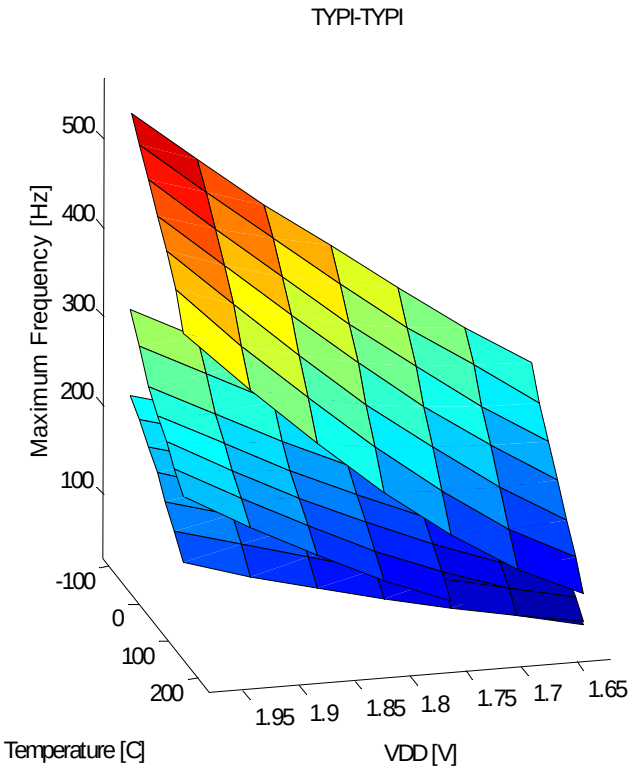

Figure 12: Max frequency of designs for typi_typi process

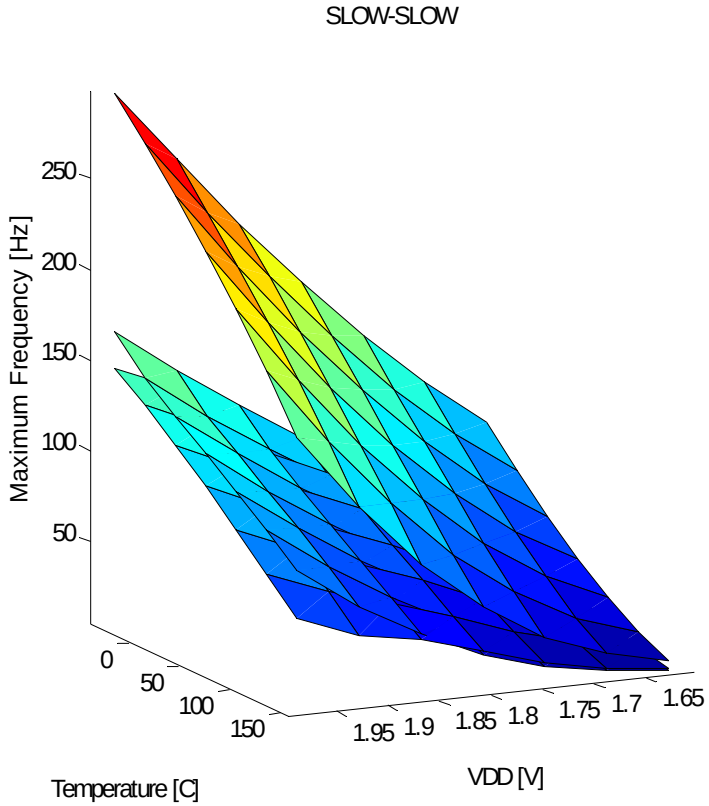

Figure 13: Max frequency of designs for slow_slow process

The surface floating far above the other two surfaces represents the carry look-ahead data, the surface in between the other two represents the dynamic ripple carry data, and the bottom surface represents the static ripple carry data. From these graphs, not only can it be seen that the carry look-ahead is the fastest design, but the general trend of increasing speed with increasing VDD and temperature can be seen as well.

Power

Another important concern is how the designs compare to one another in terms of power dissipation. Figure shows values of average power and energy per transition for

(not finished)

Conclusions

Through the comparison of the three distinct adder architectures, the carry look-ahead adder was shown to be vastly superior in terms of circuit speed over virtually all testing conditions. Power analysis subsequently showed that the carry look-ahead dissipated the most power and took up the most chip area, while the dynamic ripple-carry design was the most efficient in terms of power dissipation and chip area. However, as shown by the Shmoo plots and the data presented, the carry look-ahead architecture is the only design out of those presented that will consistently be able to operate at frequencies greater than 200MHz. By default this must eliminate all other candidates. The carry look-ahead adder is thus the most suitable design.

References

[1] Baker, R. Jacob (2005). CMOS Circuit Design, Layout, and Simulation (Second Edition). p368

[2] Baker, R. Jacob (2005). CMOS Circuit Design, Layout, and Simulation (Second Edition). P410