Technical Report GIT-CERCS-06-03

The Sleepy Keeper Approach: Methodology, Layout and Power Results for a 4-bit Adder

Se Hun Kim, Vincent J. Mooney III and Jun Cheol Park
Center for Research on Embedded Systems and Technology
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, Georgia, U.S.A.
29 March 2006

1. Introduction

This technical report explains a new approach to low leakage power Very Large Scale Integration (VLSI) design; we name the new approach "sleepy keeper." This report first introduces previous approaches to reduce leakage power consumption and then explains the methodology and findings regarding the sleepy keeper approach. The scope of this report includes test procedures with schematics and layouts for all considered approaches as well as test results such as data on delay plus dynamic and static power. The sleepy keeper results are compared with the previous approaches.

2. Base Case

All layouts and schematics are designed using the North Carolina State University (NCSU) [10] design kit targeting the Taiwan Semiconductor Manufacturing Company (TSMC) 0.18µm process [18]. Transistor sizes are specified as a ratio of Width / Length (W/L). The smallest possible transistor for the TSMC 0.18µm process has a width of 270nm and a length of 180nm, resulting in a ratio of W/L = 270nm / 180nm = 1.5. This ratio of W/L = 1.5 indicates the smallest feasible transistor size throughout this report.

This report evaluates all considered approaches using a 4-bit adder as a test case. The base case for this test circuit is a basic Complementary Metal Oxide Semiconductor (CMOS) implementation [13]. In all approaches, transistors are placed in-between two parallel rows of continuous VDD and GND. For the base case, the 4-bit adder is implemented by using a full adder shown in Figure 1 (repeated here, for convenience, from Figure A.1.a of Appendix A). In Figure 1, *a* and *b* are two inputs, *c* is a carry input, and *Carry* and *Sum* are outputs. Figure 1 also shows the transistor sizing.

3. Prior Static Current Reduction Approaches

In order to compare with the sleepy keeper approach, this section explains several previous leakage reduction approaches: transistor stacking [4][5], source gating via sleep transistors [1][6], selective source gating via alternating sleep transistors (the so-called "zigzag" approach) [7], and a combination of stack and sleep approach called sleepy stack [2][3].

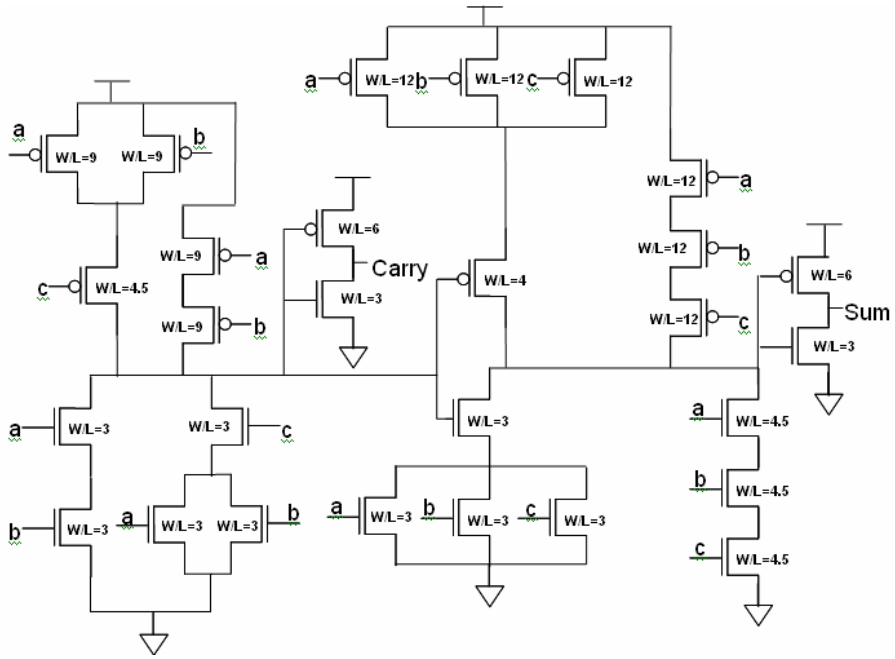


Figure 1. A 1-bit adder schematic for base case

3.1 Stack

For the stack approach, every transistor in the base case network is duplicated with both original and duplicate bearing half the original transistor width as shown in Figure 2. Duplicated transistors cause a slight reverse bias between the gate and source when both transistors are turned off. Because subthreshold current is exponentially dependent on gate bias, a substantial current reduction is obtained [4].

Since all transistors are placed in-between two parallel rows of continuous VDD and GND, stack approach design forces an increase in row length because of an increase in the number of transistors and decrease in transistor width.

3.2 Sleep

For the sleep approach, transistors gating VDD and GND are added to the base case [1][6]. The added transistors cut off supply of power when in sleep mode. Each added transistor is referred to as a "sleep transistor" and takes the width of the largest transistor in the base case. As shown in Figure 3, a PMOS sleep transistor is placed between VDD and the pull-up network, and an NMOS sleep transistor is placed between GND and the pull-down network. The sleep transistors are driven by Sleep (S) and Sleep' (S') signals. Note that the transistor widths in Figure 2 are set to show equal

resistances, and the transistor widths in Figure 3 are set based on the widths shown in Figure 2.

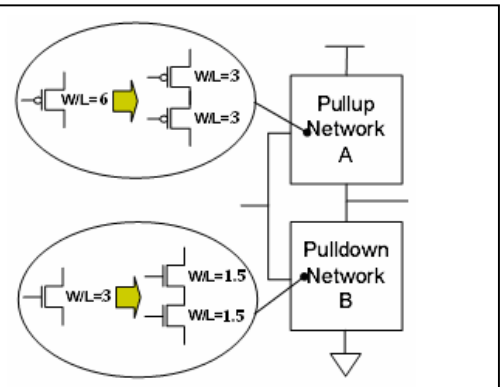


Figure 2. Stack approach.

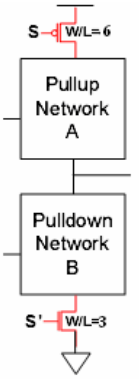


Figure 3. Sleep approach

The sleep transistors disconnect the circuit from VDD and GND when the logic circuit is not in use (i.e., when in sleep mode). By isolating logic circuitry using sleep transistors, the approach reduces subthreshold leakage current but unfortunately also loses state. In addition, time and energy for waking up are necessary. Also, the additional transistors as well as wires for S and S' require an increase in area. Finally, subthreshold leakage current can further be reduced by utilizing high threshold voltage (high-V_{th}) sleep transistors.

3.3 Zigzag

By placement of alternating sleep transistors based on which particular network (pull-up or pull-down) is off given a specific input vector, the zigzag approach reduces wake-up overhead delay caused by sleep transistors [7]. For example in Figure 4, if the output is '1' when input 'a' is asserted to a particular value, then a sleep transistor is placed in the associated pull-down network; if the output is '0', then a sleep transistor is placed in the associated pull-up network. In order to evaluate this approach, the result of static power dissipation for all zero inputs is chosen for comparison with other approaches because reset input values are typically all zeros in most cases. In addition, subthreshold leakage can further be reduced by using high-V_{th} sleep transistors. The reduced number of sleep transistors in this zigzag approach results in smaller increase in area than by using the sleep approach.

3.4 Sleepy Stack

The sleepy stack approach has a structure combining the stack and sleep approaches by dividing every transistor into two transistors of half width and placing a sleep transistor in parallel with one of the divided transistor [2] [3]. As shown in Figure 5, sleep transistors are placed in parallel to the divided transistor closest to VDD for pull-up and in parallel to the divided transistor closest to GND for pull-down.

The sleepy stack approach can have advantages of both the stack approach and the sleep approach. During active mode, the sleepy stack approach results in lower delay than the stack approach because sleep transistors placed in parallel (i) reduce resistance and (ii) are already on. When sleep transistors are turned off, the existence of a path from either VDD or GND prevents floating output. Also, leakage current can further be reduced by applying high-V_{th} on sleep transistors and the transistors in parallel to the sleep transistors (e.g., the slightly shaded/colored transistors in Figure 5).

However, area penalty is significant matter since every transistor is replaced by three transistors and since additional wires are added for S and S', which are sleep signals.

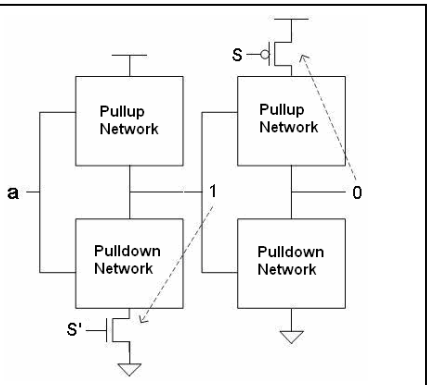


Figure 4. Zigzag approach.

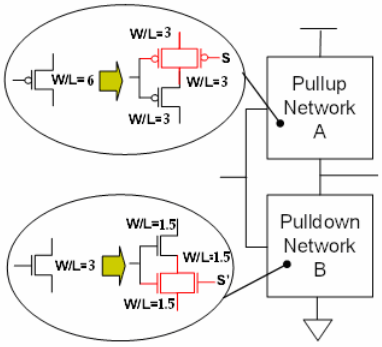


Figure 5. Sleepy-stack approach.

We briefly summarized several prior static current reduction approaches in this section. We will mention these approaches again as we motivate and explain our new approach in following sections.

4. Motivation

Leakage power consists mainly of subthreshold leakage and gate-oxide leakage. A potential solution widely reported for gate-oxide leakage power is the possible use of high-k (high dielectric constant) gate insulators [16]. Currently, subthreshold leakage

power seems to be the majority contributor to total leakage power [17]. In any case, this technical paper targets reduction of the subthreshold leakage component of static power consumption; other approaches (most likely orthogonal to what we propose here in this paper) should be considered for reduction of gate-oxide leakage. Do please note, however, that all results reported in this paper include all sources of leakage power (to the extent that the HSPICE models we use accurately model sources of leakage).

With application of dual V_{th} techniques, the sleep, zigzag and sleepy stack approaches result in orders of magnitude subthreshold leakage power reduction [3]. The major advantage of the sleepy stack approach (see previous section) over the sleep and zigzag approaches is that the sleepy stack approach saves exact logic state. However, the sleepy stack approach carries a nontrivial penalty: each transistor in the original, base case, traditional CMOS design results in three transistors in the sleepy stack equivalent. The goal of our new approach is to achieve the benefits of the sleepy stack approach without the large associated penalties due to the tripled transistor count.

One final comment about motivation is that we assume proper logic design and timing for transition to sleep mode for sleepy keeper VLSI circuits. In particular, we assume that there is a small delay (perhaps a few clock cycles of a gigahertz clock) between the final computation in active mode and the transition to sleep mode. This allows the transition to sleep mode to only require that existing logic state/values be maintained. Finally, we further assume that transition from sleep mode back to active mode also has a few clock cycles of delay between turning sleep transistors back on and beginning to actively calculate new logic values (i.e., beginning to change state again).

5. New Static Current Reduction Approach: Sleepy Keeper

In this section we will describe the new VLSI approach to leakage power reduction proposed in this technical report: the "sleepy keeper" approach. First, we will discuss the structure of the sleepy keeper approach and how it operates. Then, we explain how layouts for the sleepy keeper approach are created.

The basic problem with traditional CMOS is that the transistors are used only in their most efficient, and naturally inverting, way: namely, PMOS transistors connect to VDD and NMOS transistors connect to GND. It is well know that PMOS transistors are not efficient at passing GND; similarly, it is well know that NMOS transistors are not

efficient at passing VDD. However, to maintain a value of '1' in sleep mode, given that the '1' value has already been calculated, the sleepy keeper approach uses this output value of '1' and an NMOS transistor connected to VDD to maintain output value equal to '1' when in sleep mode. For example, when the output is '1' for an inverter designed utilizing the sleepy keeper approach, the current path is shown in Figure 6.

Similarly, to maintain a value of '0' in sleep mode, given that the '0' value has already been calculated, the sleepy keeper approach uses this output value of '0' and a PMOS transistor connected to GND to maintain output value equal to '0' when in sleep mode. For example, when the output is '0' for an inverter implemented using the sleepy keeper approach, the current path is shown in Figure 7.

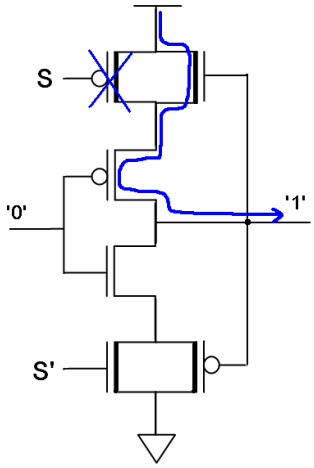


Figure 6. Inverter for sleepy keeper approach (output = '1')

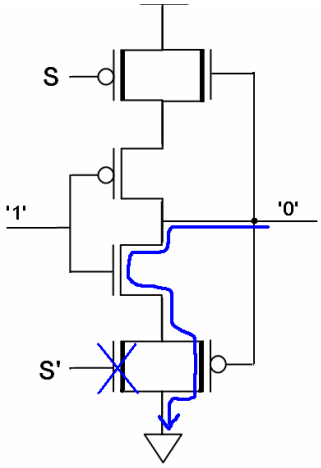


Figure 7. Inverter for sleepy keeper approach (output = '0')

For this sleepy keeper approach to work, all that is needed is for the NMOS connected to VDD and the PMOS connected to GND to be able to maintain proper logic state. This seems likely to be possible as other researchers have described ways to use far lower VDD values to maintain logic state. For example, Flautner et al. propose some significantly reduced VDD values sufficient to maintain state [14].

In any case, we do not investigate – beyond the use of HSPICE [8] simulations – all of the possible side effects due to using PMOS transistors to connect to GND and NMOS transistors to connect to VDD. Instead, we assume that the HSPICE simulations are roughly accurate and report results based on HSPICE.

Consider Figure 8. Note that there is a sleepy keeper PMOS transistor connecting

GND to the pull-down network. When in sleep mode, this PMOS transistor is the only source of GND since the sleep transistor is off. On the other hand, in Figure 8, there is an additional single NMOS transistor connecting VDD to the pull-up network. During sleep mode, this NMOS transistor is the only source of VDD which is the dual case of the PMOS transistor case explained above.

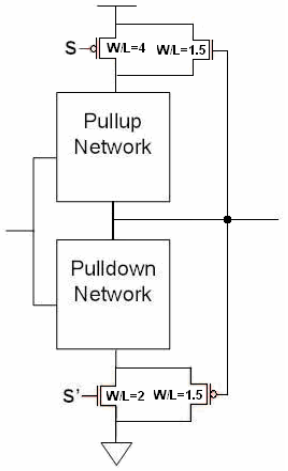


Figure 8. Sleepy keeper approach structure.

We wish to here emphasize that, as explained at the end of Section 4, we emphatically do not use sleepy keeper transistors (the NMOS connected to VDD and the PMOS connected to GND) to dynamically change the output voltage but instead only use them to maintain an already calculated output voltage. Specifically, only a few clock cycles after entering sleep to a few clock cycles prior to exiting sleep do the sleepy keeper transistors acts as the sole connection to keep the output voltage unchanged.

6. Experimental Methodology

In this section, we explain our experimental methods. First, we describe how we create layouts and schematics in preparation for HSPICE simulation. Second, we explain how we obtain estimated results for delay, power consumption and area.

6.1 Layouts, Schematics and HSPICE

Schematics and layouts are designed for all considered design approaches. Schematics are used to obtain netlists corresponding to the test circuit, and the netlists are

used to simulate and test performance for the Berkeley Predictive Technology Model (BPTM) [11] [12] 0.18, 0.13, 0.10, and 0.07µm processes and the TSMC 0.18µm process using HSPICE. Layouts are used to measure and predict area usage. The estimation procedure we use is summarized in Figure 10.

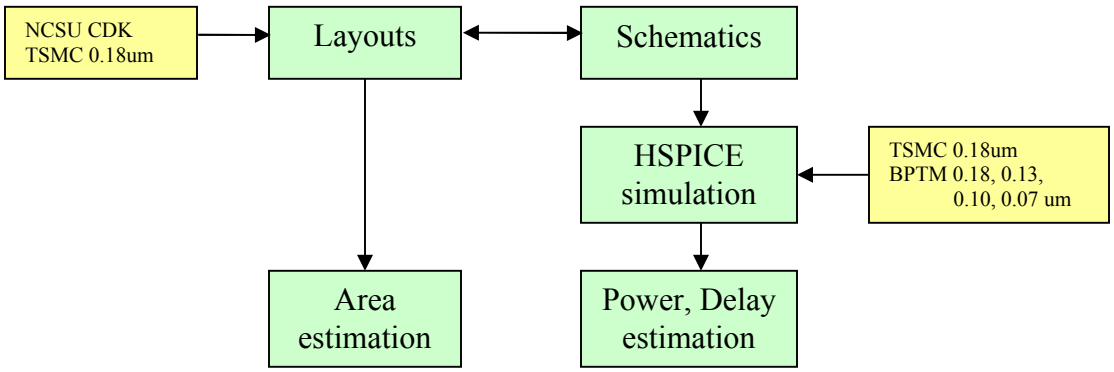


Figure 10. Experimental methodology.

We create schematics of 4-bit adders for all considered approaches using Cadence Virtuoso Schematic Editor [9]. We extract netlists from the schematics by using Cadence Virtuoso Analog Environment [9]. For example, the schematic of an interter with sleepy keeper approach is shown in Figure 11. Since the schematics are designed for the TSMC 0.18µm process, the netlists do not exactly match for the BPTM processes because the netlists include library and parameters for the TSMC 0.18µm process (e.g., see Example 1 on the next page). Since modification of netlists and performing HSPICE simulation include many repetitions of the same or similar procedure, we use an automatic system which generates template netlists and performs HSPICE simulation for the BPTM 0.18, 0.13, 0.10 and 0.07µm processes and the TSMC 0.18µm process. The template netlists are modified from the original extracted netlist as needed so that the netlists can be used for all considered technologies. Some perl scripts are used to make template netlists and run the HSPICE simulations for different technologies. Note that a variety of programming languages can be used to perform this automatic system. Example 1 shows the process of generating a sample template netlist.

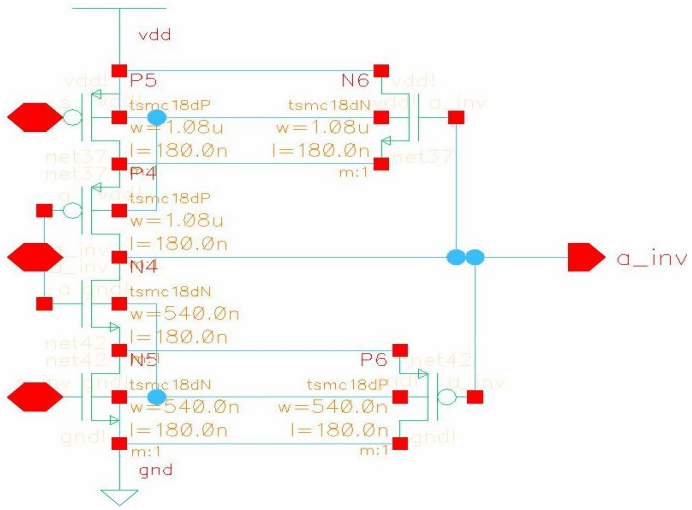


Figure 11. Schematic capture of an inverter with sleepy keeper approach.

```
* # FILE NAME: /HOME/SYNTHESIS/CADENCE/SIMULATION/4ADDER_CHAIN/
* HSPICES/SCHEMATIC/NETLIST/4ADDER_CHAIN.C.RAW
* NETLIST OUTPUT FOR HSPICES

MN2 VDD! A_INV NET28 VDD!  TSMC18DN  L=180E-9 W=1.08E-6 AD=486E-15 AS=486E-
15 +PD=3.06E-6 PS=3.06E-6 M=1
MN1 A_INV A NET9 0  TSMC18DN  L=180E-9 W=540E-9 AD=243E-15 AS=243E-15
+PD=1.98E-6 PS=1.98E-6 M=1           .
               .
               .

.lib "/ncsu/cadence/local/models/hspice/public/publicModel/tsmc18dP" PMOS
.lib "/ncsu/cadence/local/models/hspice/public/publicModel/tsmc18dN" NMOS

.END
```

Figure 12. An example raw netlist (TSMC 0.18µm process).

Example 1 :

Figure 12 shows a sample netlist extracted from the schematic of Figure 11. This netlist includes some information such as a reference to the TSMC 0.18µm process (e.g., “TSMC18DP”), which is not proper content for BPTM processes. First, we keep all listed of connections from this netlist. In Figure 12, “MN2 VDD! A_INV NET28 VDD! TSMC18DN L=180E-9 W=1.08E-6 AD=486E-15 AS=486E-15 +PD=3.06E-6 PS=3.06E-6 M=1” defines a PMOS transistor named MP2 with its drain connected to node 0, gate to SUM_INV, source to NET71 and bulk to node 0. Second, in our template netlists, a variable “length” is used for Length (L), and a variable “pwidth” is used for width (W) for different technologies (e.g., see underlined words in Figure 13). For different technologies, we define proper values for the variables. Lastly, appropriate parameters, test vectors, libraries and reports (e.g., bolded words in Figure 13) are fed into the template netlists for the HSPICE

simulations for different technologies.□

```
* autogenerated netlist file for adder
* based on Keeper-4adder_test.sp

.include /home/hspice/parameters/parameters_#SIZE#u.sp

MN2 VDD! A_INV NET28 VDD! NMOSH  L='length' W='1.5*pwidth'
MN1 A_INV A NET9 0  NMOS  L='length' W='4*pwidth'
                        .
                        .
                        .

.global vdd!
.include "/home/hspice/test_vectors/adder_#TESTTYPE##INPUT#.sp"

.include "/home/hspice/berkeley_models/#LIBRARY#_#SIZE#u.sp"
.include "/home/hspice/reports/adder_#TESTTYPE#_report.sp"
.END
```

Figure 13. An example template netlist.

All six considered approaches are evaluated for performance by using a single threshold voltage (V_{th}) for all transistors. Dual V_{th} technology is applied and tested only for the sleep, zigzag, sleepy stack, and sleepy keeper approaches since applying high-V_{th} to the base case and the stack approach causes dramatic increase of delay (at least 2-5X). For both single V_{th} and dual V_{th} techniques, delay, dynamic power consumption and static power consumption are measured by using HSPICE. In order to measure performance of sleep, zigzag, sleepy stack, and sleepy keeper with dual V_{th} values, every sleep transistor and any transistor parallel to the sleep transistor are configured as high-V_{th} transistors. The high-V_{th} is set to have 2.0 times higher V_{th} than the V_{th} of a normal transistor (low-V_{th}). The "Delvto" option of HSPICE is used to change V_{th}. In order to distinguish two different V_{th} values, "NMOSH" or "PMOSH" is used to indicate high-V_{th} and "NMOS" or "PMOS" is used for low-V_{th}. Figure 12 shows an example of the PMOS case.

6.2 Delay

Worst case propagation delay is measured for each approach. Input vectors and input/output triggers are chosen to measure the delay of a critical path. The propagation delay is measured from the trigger input edge reaching 50% of the supply voltage to the circuit output edge reaching 50% of the supply voltage value.

6.3 Static Power

Static power is measured by asserting sets of input vectors in HSPICE. The input vectors include subsets of possible input combinations. The average power dissipation over the specific subset of input combinations chosen is determined as the static power for the base case, stack, sleep, sleepy stack and sleepy keeper techniques. All sleep transistors are turned off for the HSPICE measurements. As mentioned in static current reduction approaches section, static power for the zigzag approach is determined to be the power dissipation of tested result for inputs all zeros (reset input values).

6.4 Dynamic Power

In order to measure dynamic power, clocked semi-random input vectors for a number of clock cycles are asserted, and average power dissipation during this time reported by HSPICE is considered as estimation of dynamic power consumption. All sleep transistors are turned on for HSPICE measurements.

6.5 Area

Layouts of a 1-bit full adder for all the considered approaches are designed based on TSMC 0.18μm process by using Cadence Virtuoso Layout Editor [9] and NCSU Cadence Design Kit. Layouts are verified with Virtuoso's Design Rule Checker (DRC). Areas for below 0.18μm technology are estimated by scaling the area of each approach layout designed based on TSMC 0.18μm process. The areas are scaled by a ratio of squares with addition of a 10% overhead for nonlinear scaling layers (i.e., metal layers). For example, if an area of 100.00μm^2 is measured for 0.18μm technology, the area for 0.10μm technology would be 100.00μm^2 * (0.10^2 / 0.18^2) * 1.1 = 33.95 μm^2.

6.5 Equations used for comparison

When we compare our results to another result, we often say one is "less than" the other. In particular, "X is n% less than Y" means what Eq. 1 shows:

$$\frac{Y}{X} = 1 + \frac{n}{100} \qquad \text{Eq. 1 [19]}$$

For example, when two propagation delay measurements result in, X is 8.18E-10s and Y is1.23E-09s, n is 50 from calculation using Eq. 1. In this case, we say X is 50% less

delay than Y. This equation is used for all other comparison such as area and power consumption.

7. Test Circuit – 4-bit adder

We use a full adder as an example of a typical complex CMOS gate. Our 4-bit adder is implemented by using four 1-bit full adders. A 1-bit full adder is created from four logic blocks, one block to generate inverted Carry out (Cout'), one block to generate an inverted Sum (Sum') and two inverters as shown in Figure 13. The complex blocks are sized to have an equal rise and fall time. Appendix A.1.a shows the sizing for the base case. In deed, please see Appendix A for exact transistor sizing for all considered VLSI approaches.

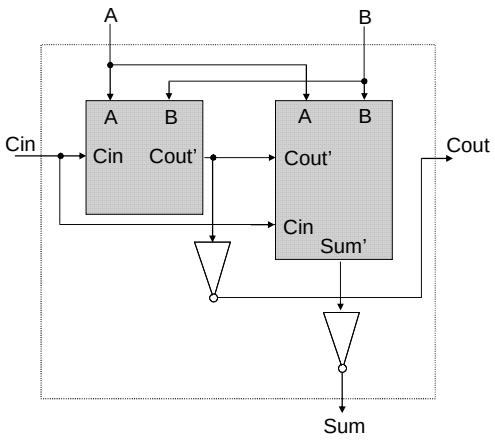


Figure 13. Network of complex gates and inverters composing a 1-bit full adder.

a. Delay

The critical path of our base case 4-bit adder is the path $B_0 - C_{out0} - C_{in1} - C_{out1} - C_{in2} - C_{out2} - C_{in3} - Sum_3$. In order to measure the worst case propagation delay, initial input signals are set as shown in Figure 14. When B_0 is changed to 1, the delay is measure from B_0 to Sum_3.

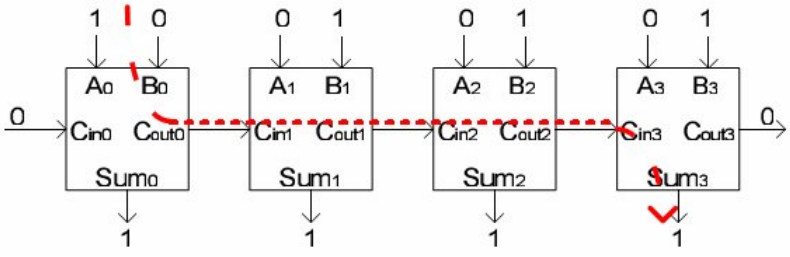


Figure 14. Inputs of 4-bit adder for critical path delay

b. Static Power

Nine input bits (A[3:0], B[3:0], Cin0) provide 2^9(512) possible input combinations. Eight input vectors out of 512 possible input combinations are chosen for the 4-bit adder. Table 1 shows the eight input combinations. The average power dissipation for each input vector during 20ns (per static input vector) is recorded as the static power of each circuit considered.

Table 1. Static power assessment inputs used for 4-bit adder.

C_{in}	A_0	B_0	A_1	B_1	A_2	B_2	A_3	B_3
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0
1	1	0	1	0	1	0	1	0
1	1	1	1	0	1	0	1	0
1	1	1	1	1	1	0	1	0
1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1

c. Dynamic Power

Similar to our approach to static power estimation, to estimate dynamic power we assert input vectors covering eight possible inputs out of 512 possible input combinations. Each input vector is asserted followed by all zero inputs (reset values) except the case when the inputs are all zeros for the first time. The waveform in Figure 15 shows input vectors asserted for each one bit adder, where the input vector changes in every 4ns. The average power dissipation during the 30ns of Figure 15 is recorded as the dynamic power of the circuit.

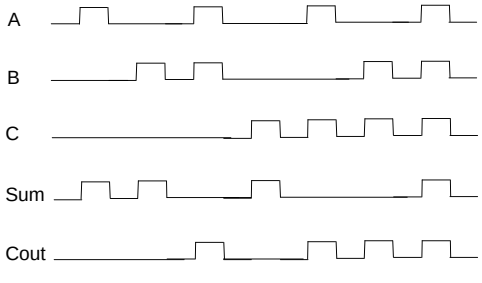


Figure 15. Dynamic power assessment waveform for full adder

d. Area

We create a full transistor-level layout of a 1-bit adder based on TSMC 0.18μm technology. The area for the 1-bit adder is measured; the area for the 4-bit adder is determined as the sum of four 1-bit adders. Area results for the other technologies considered (e.g., 0.07μm) are calculated as explained in Section 6.5.

8. Experimental Results

For the 4-bit adder circuit, propagation delay, static power, dynamic power, and area are shown in Figure 16 and Table 2.

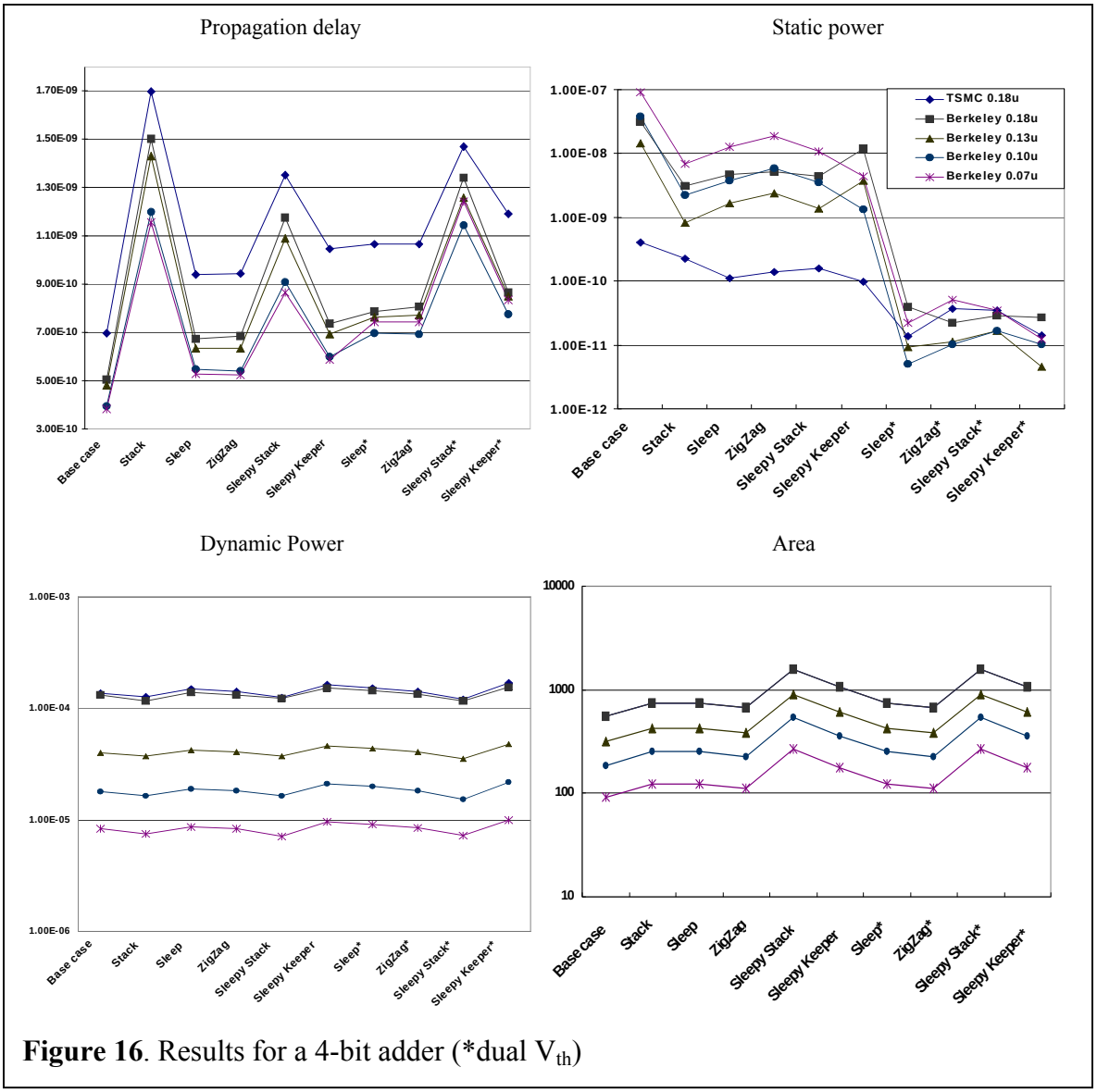


Figure 16. Results for a 4-bit adder (*dual V_{th})

Table 2. Power, delay, area estimation for 0.07μm

Berkeley 0.07μm	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μm^2)
Base case	3.82E-10	8.97E-08	8.28E-06	91.84
Stack	1.16E-09	6.83E-09	7.41E-06	123.76
Sleep	5.29E-10	1.25E-08	8.66E-06	123.76
ZigZag	5.25E-10	1.84E-08	8.37E-06	110.48
Sleepy Stack	8.64E-10	1.08E-08	7.06E-06	263.52
Sleepy Keeper	5.85E-10	4.40E-09	9.62E-06	177.11
Sleep (dual Vth)	7.45E-10	2.23E-11	9.02E-06	123.76
ZigZag (dual Vth)	7.43E-10	5.05E-11	8.46E-06	110.48
Sleepy Stack (dual Vth)	1.24E-09	3.50E-11	7.26E-06	263.52
Sleepy Keeper(dual Vth)	8.33E-10	1.22E-11	9.99E-06	177.11

In 0.07μm technology, the sleepy keeper approach (with dual V_{th}) achieves 7350X leakage reduction over the base case and 560X leakage reduction over the stack approach. The result is similar to the previous best leakage reduction technique with state saving, sleepy stack, but sleepy keeper achieves less delay than sleepy stack. In 0.07μm technology, sleepy keeper results in 48% less delay than sleepy stack with single V_{th} and 49% less delay with dual V_{th}. Sleepy keeper consumes 36% more dynamic power than sleepy stack with single V_{th} and 38% more dynamic power than sleepy stack with dual V_{th}. The dynamic power result is roughly 21% increase of dynamic power over the base case. Finally, area usage of the sleepy keeper is 93% larger than the base case, but it is 49% smaller than area usage of the sleepy stack.

The experimental results for all other considered technologies are available in Appendix B. The Figure 16 and Table 2 are based on the results in Appendix B.

9. Conclusion

Based on the 4-bit adder test results, we have verified that the sleepy keeper approach can result in ultra-low static power consumption with state saving. Furthermore, the sleepy keeper approach is applicable to single and multiple threshold voltages. Compared to the sleepy stack approach, sleepy keeper requires 49% less area as well as achieves smaller propagation delay (up to 49% less). Therefore, sleepy keeper can reduce the main penalties to using the sleepy stack approach, while still achieving the same twin advantages of ultra-low leakage and maintenance of precise logic state in sleep mode. Based on these results, sleepy keeper appears to be the most efficient approach

known to reduce leakage current with the smallest delay and saving state. In terms of area, sleepy keeper is expected to be more attractive for complex logic circuits, because the portion of increased area for the required additional transistors will be smaller for complex logic circuits than for simple logic circuits (e.g., for an inverter).

The sleepy keeper approach causes dynamic power increase which seems to be the main disadvantage of the approach. The increase is most likely due to placing an NMOS transistor in a pull-up network and a PMOS transistor in a pull-down network where the two added transistors are controlled by the output voltage. In order to reduce sleepy keeper dynamic power consumption, additional issues, including solid state circuit issues, should be investigated; this is left as future work.

References

[1] S. Mutoh et al., "1-V Power Supply High-speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 30, No. 8, pp. 847-854, August 1995.

[2] J.C. Park, V. J. Mooney III and P. Pfeiffenberger, "Sleepy Stack Reduction of Leakage Power," *Proceeding of the International Workshop on Power and Timing Modeling, Optimization and Simulation*, pp.148-158, September 2004.

[3] J. Park, "Sleepy Stack: a New Approach to Low Power VLSI and Memory," Ph.D. Dissertation, School of Electrical and Computer Engineering, Georgia Institute of Technology, 2005. [Online]. Available http://etd.gatech.edu/theses /available/etd-07132005-131806/.

[4] Z. Chen, M. Johnson, L. Wei and K. Roy, "Estimation of Standby Leakage Power in CMOS Circuits Considering Accurate Modeling of Transistor Stacks," *International Symposium on Low Power Electronics and Design*, pp. 239-244, August 1998.

[5] S. Nadara, S. Borkar, V. De, D. Antoniadis and A. Chandrakasan, "Scaling of Stack Effect and its Application for Leakage Reduction," *International Symposium on Low Power Electronics and Design*, pp. 195-200, August 2001.

[6] M. Powell, S.-H. Yang, B. Falsafi, K. Roy and T. N. Vijaykumar, "Gated-VDD: A Circuit Technique to Reduce Leakage in Deep-submicron Cache Memories," *International Symposium on Low Power Electronics and Design*, pp. 90-95, July 2000.

[7] K.-S. Min, H. Kawaguchi and T. Sakurai, "Zigzag Super Cut-off CMOS (ZSCCMOS) Block Activation with Self-Adaptive Voltage Level Controller: An Alternative to Clock-gating Scheme in Leakage Dominant Era," *IEEE International Solid-State Circuits Conference*, pp. 400-401, February 2003.

[8] Synopsys Inc., http://www.synopsys.com/.

[9] Cadence Design Systems, http://www.cadence.com/.

[10] NC State University Cadence Tool Information, http://www.cadence.ncsu.edu/.

[11] Berkeley Predictive Technology Model (BPTM), http://www.eas.asu.edu/~ptm/.

[12] Y. Cao, T. Sato, D. Sylvester, M. Orshansky and C. Hu, "New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design," *Proc. of IEEE Custom Integrated Circuits Conference*, pp. 201-204, June 2000.

[13] N. Westel, and K. Eshraghian, *Principles of CMOS VLSI Design*. Santa Clara, California: Addision Wesley, 1992.

[14] K. Flautner, N. S. Kim, S. Martin, D. Blaauw and T. Mudge, "Drowsy Caches: Simple Techniques for Reducing Leakage Power," *Proceedings of the International Symposium on Computer Architecture*, pp. 148-157, May 2002.

[15] MOSIS, http://www.mosis.org/.

[16] G. Ribes, J. Mitard, M. Denais, S. Bruyere, F. Monsieur, C. Parthasarathy, E. Vincent and G. Ghibaudo, "Review on High-k Dielectrics Reliability Issues," *IEEE Transactions on Device and Materials Reliability*, Vol. 5, Issue 1, pp. 5-19, March 2005.

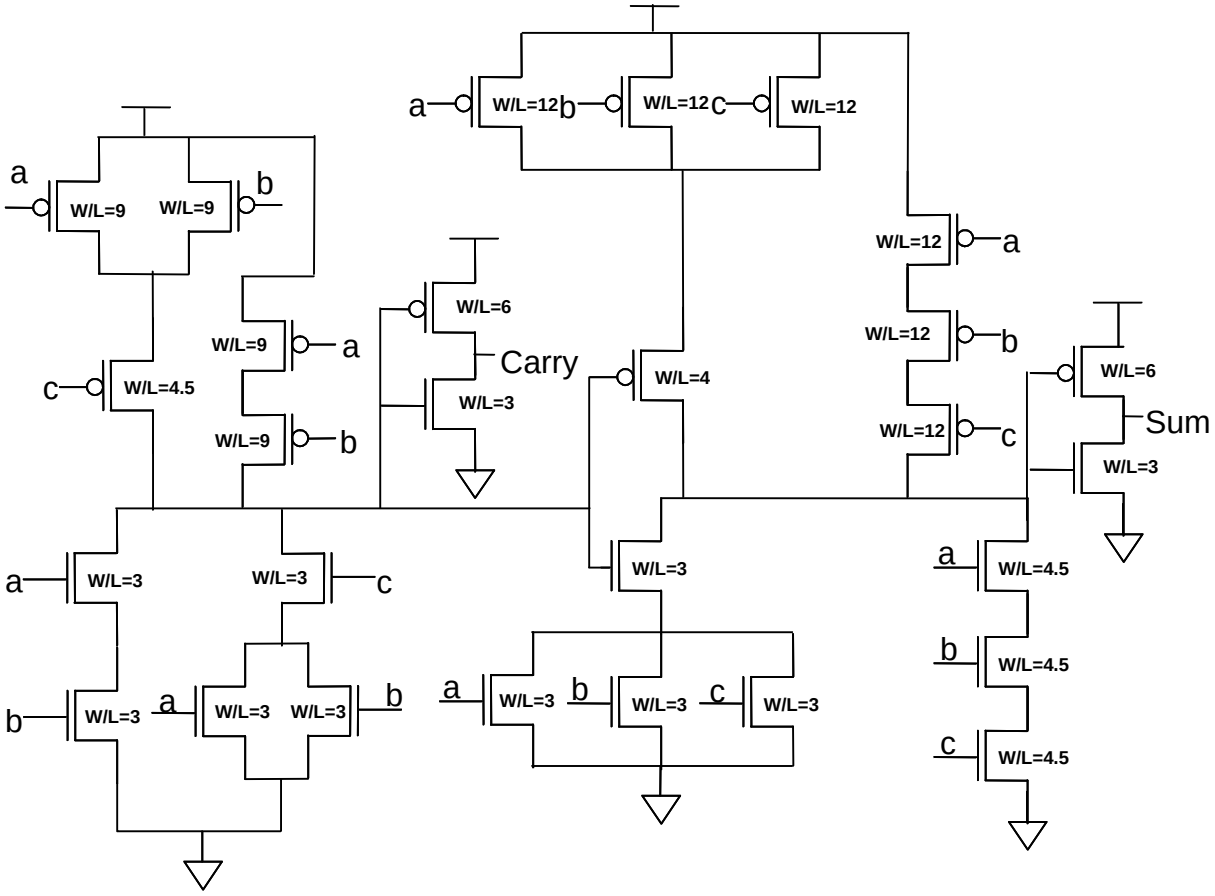
[17] A.B. Kahng, S. Muddu and P. Sharma, "Defocus-aware Leakage Estimation and Control," *International Symposium on Low Power Electronics and Design,* pp. 263-268, Aug. 2005.

[18] Taiwan Semiconductor Manufacturing Company, http://www.tsmc.com/.

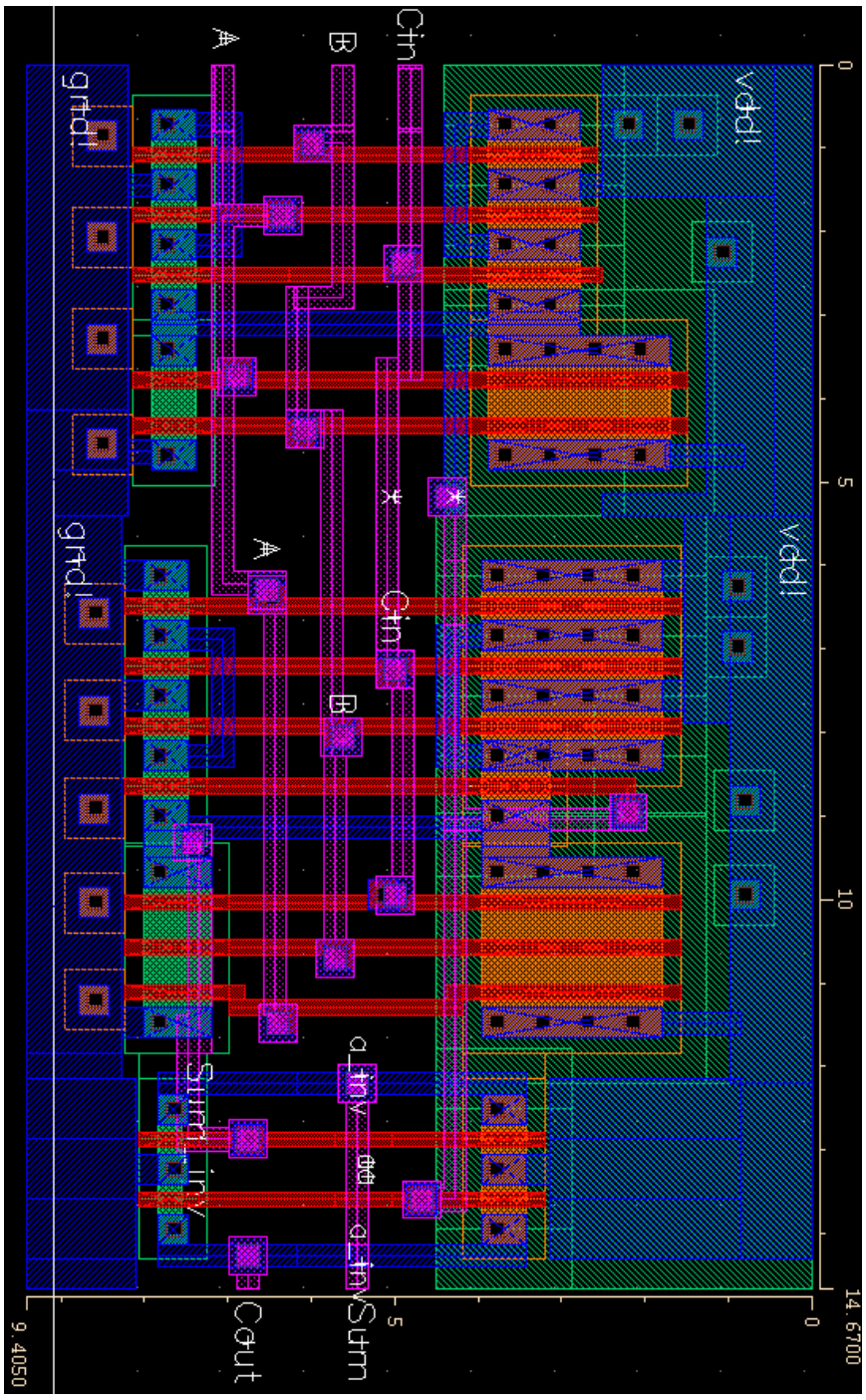
[19] D. Patterson and J. Hennessy, *Computer Architecture: A Quantitative Approach*. Palo Alto, California: Morgan Kaufmann Publishers, pp. 5-7, 1990.

Appendix A: Schematics and Layouts

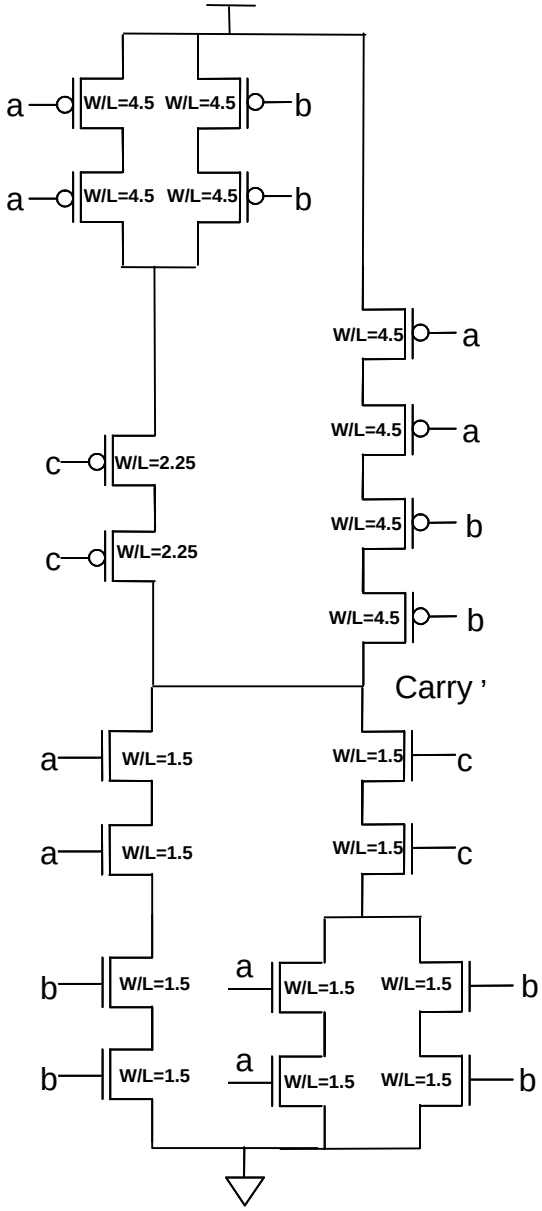
1) Base approach
    a) Schematic
    b) Layout
2) Stack approach
    a) Schematic
        (i) Cout'
        (ii) Sum'
    b) Layout
3) Sleep approach
    a) Schematic
        (i) Cout'
        (ii) Sum'
    b) Layout
4) Zigzag approach
    a) Schematic
        (i) Cout'
        (ii) Sum'
    b) Layout
        (i) Cout'
        (ii) Sum'
        (iii) Full Adder
5) Sleepy stack approach
    a) Schematic
        (i) Cout'
        (ii) Sum'
    b) Layout
        (i) Cout'
        (ii) Sum'
        (iii) Full Adder
6) Sleepy keeper approach
    a) Schematic
        (i) Cout'
        (ii) Sum'
    b) Layout
        (i) Cout'
        (ii) Sum'
        (iii) Full Adder



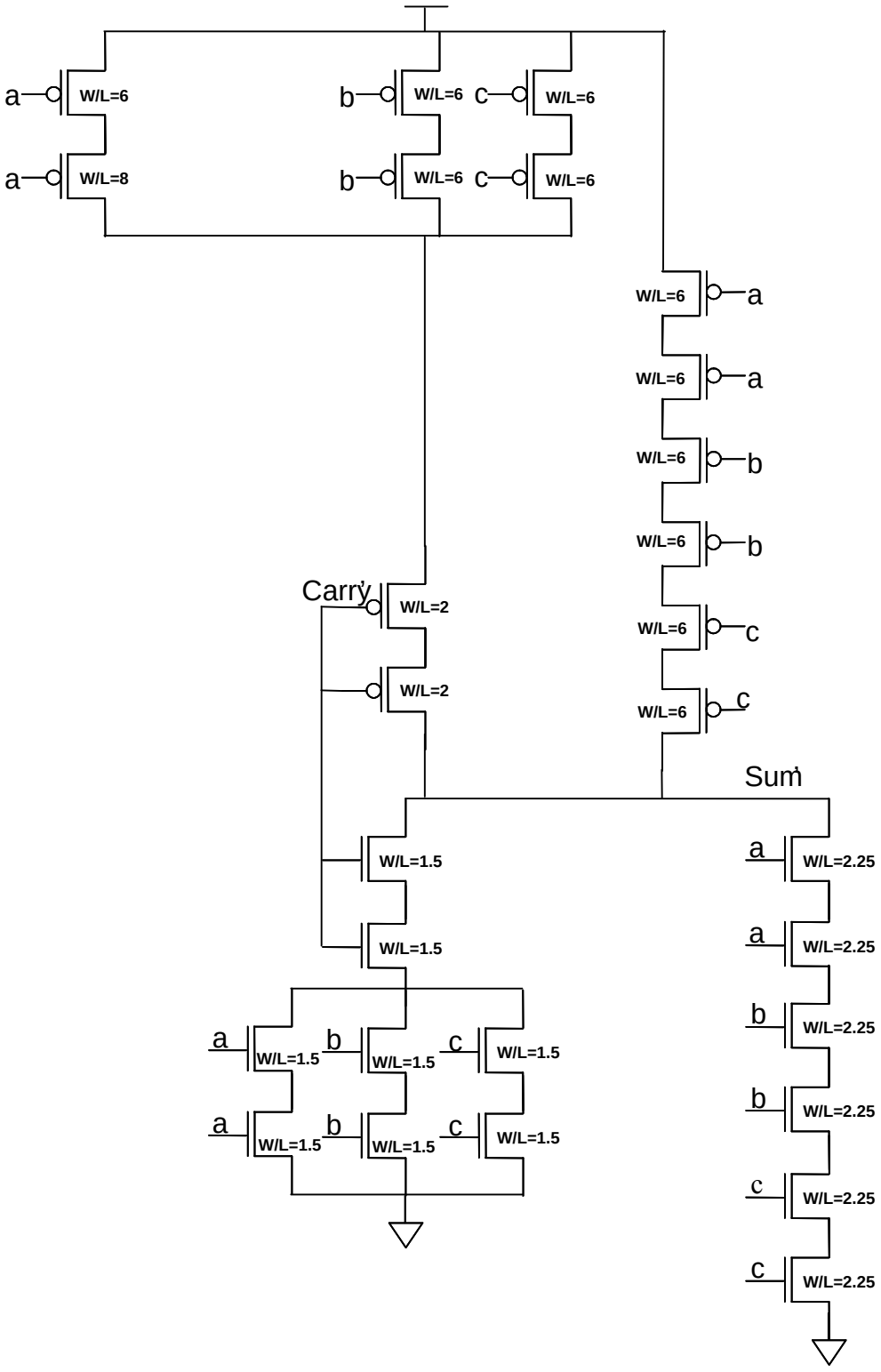
A.1.a. Base case full adder schematic



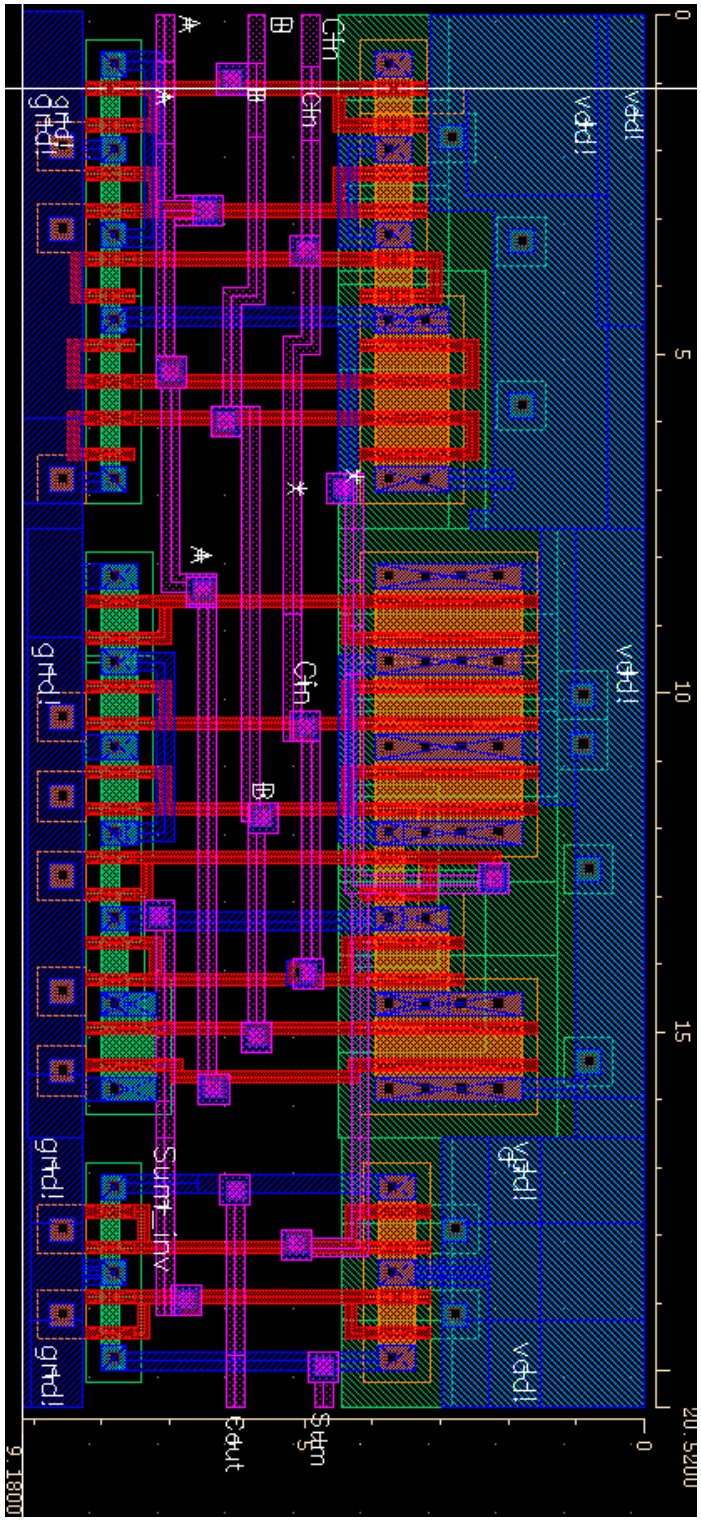
A.1.b. Base case full adder layout



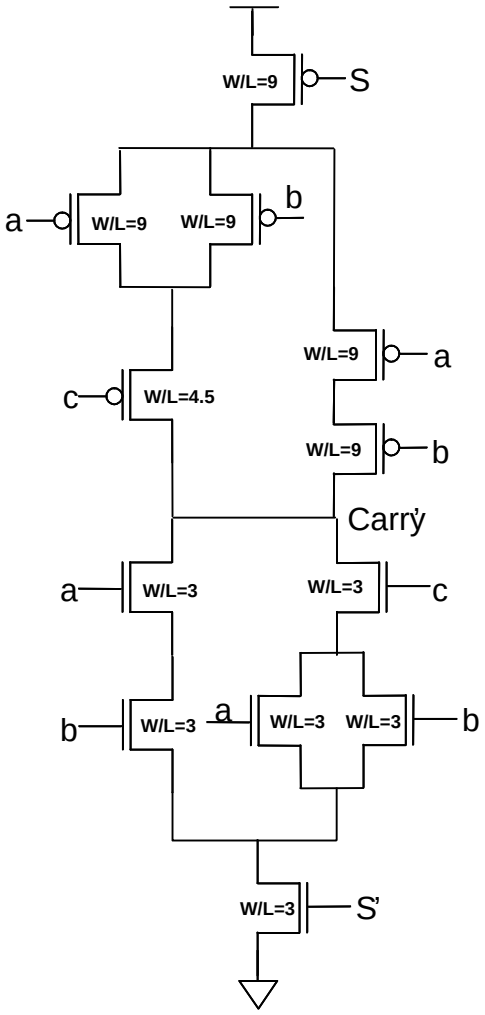
A.2.a.i. Stack approach Full Adder Cout' schematic



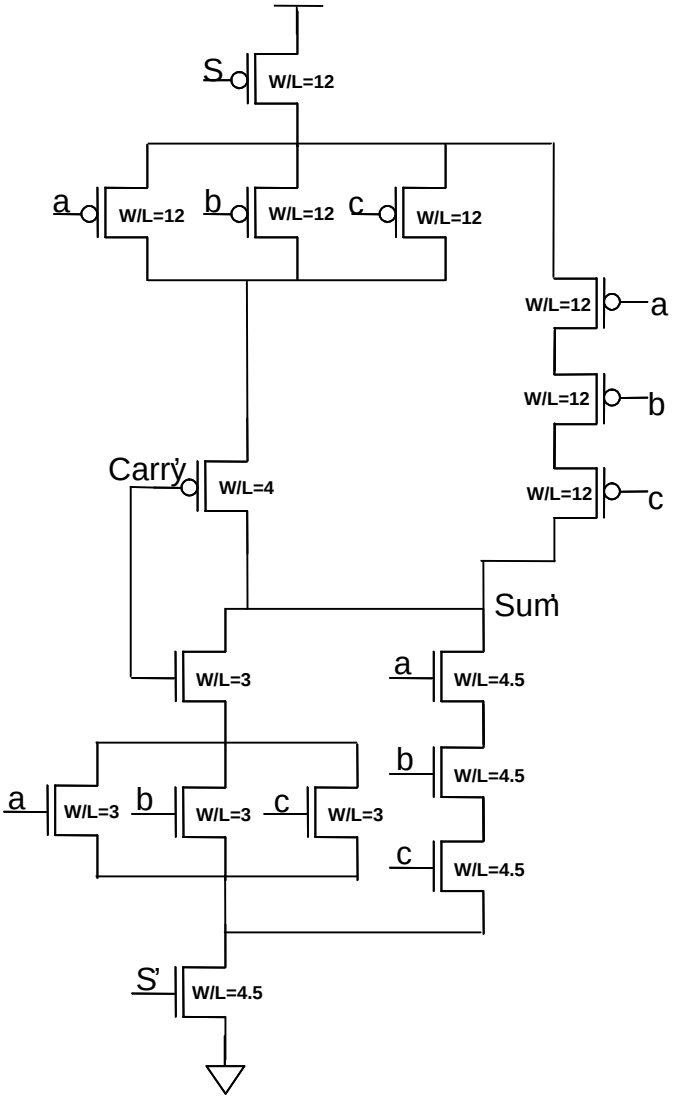
A.2.a.ii. Stack approach Full Adder Sum' schematic



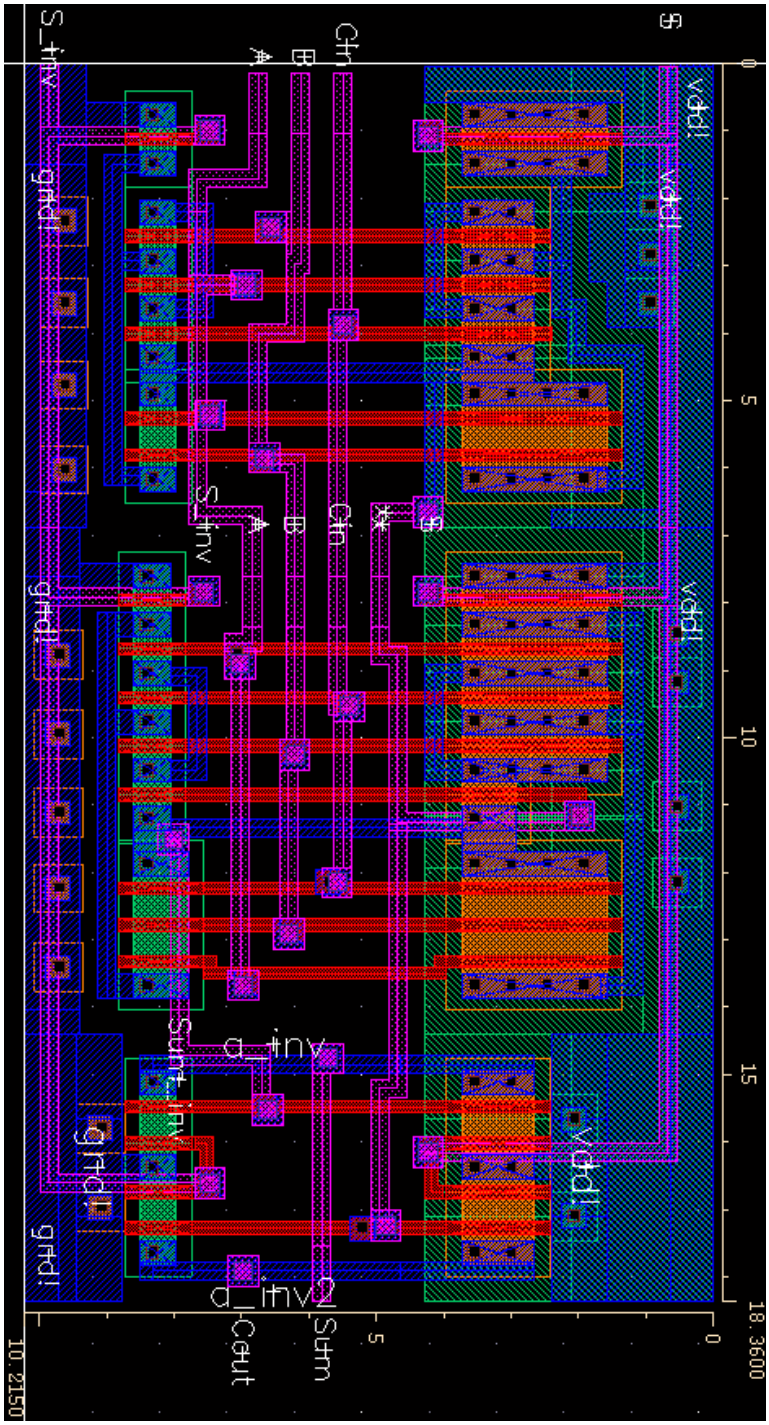
A.2.b. Stack approach Full Adder layout



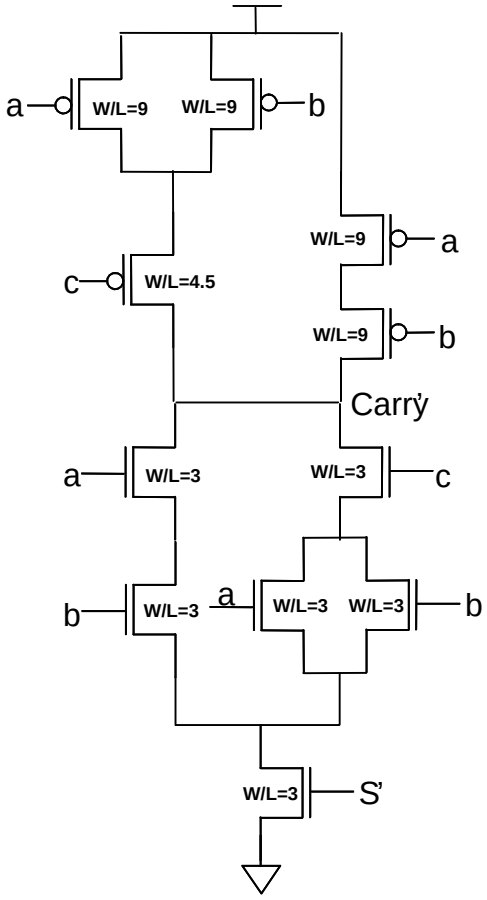
A.3.a.i. Sleep approach Full Adder Cout' schematic



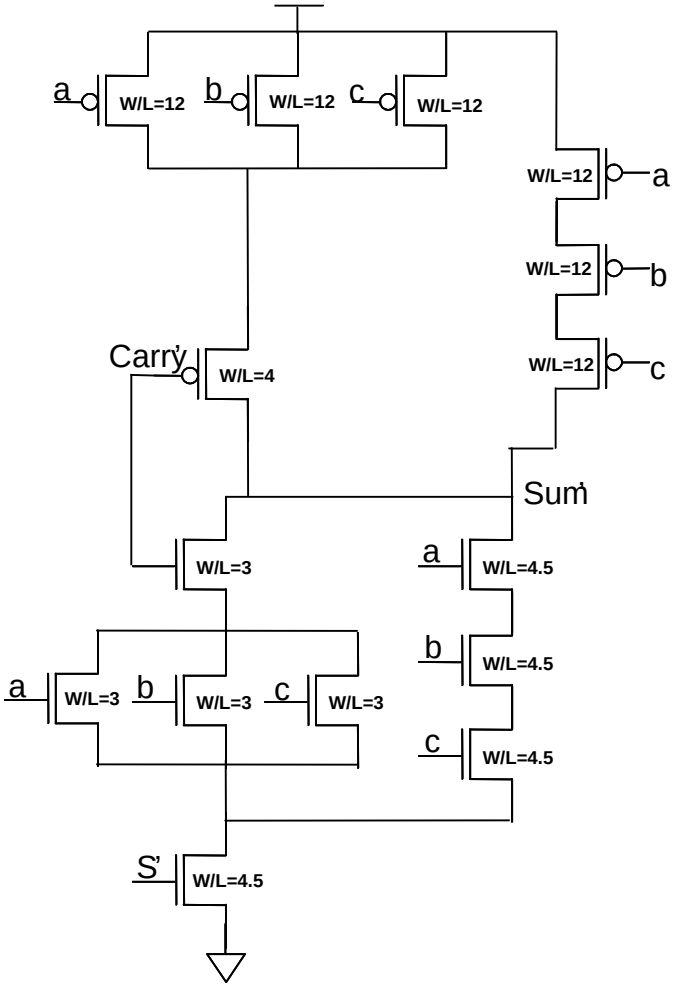
A.3.a.ii. Sleep approach Full Adder Sum' schematic



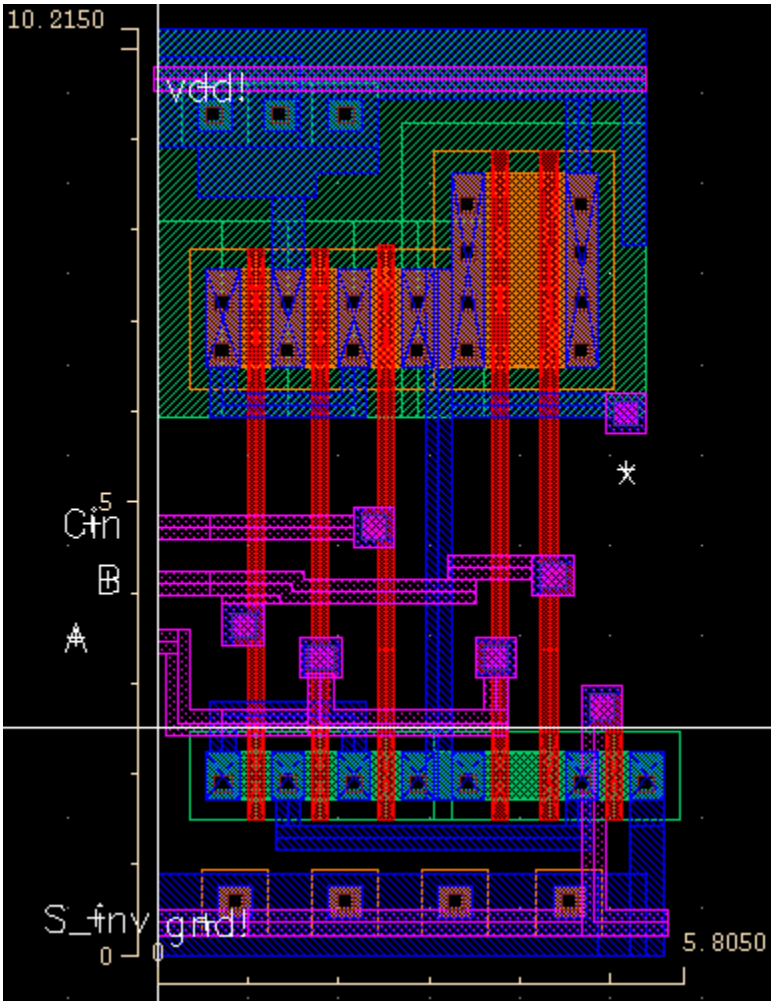
A.3.b. Sleep approach Full Adder layout



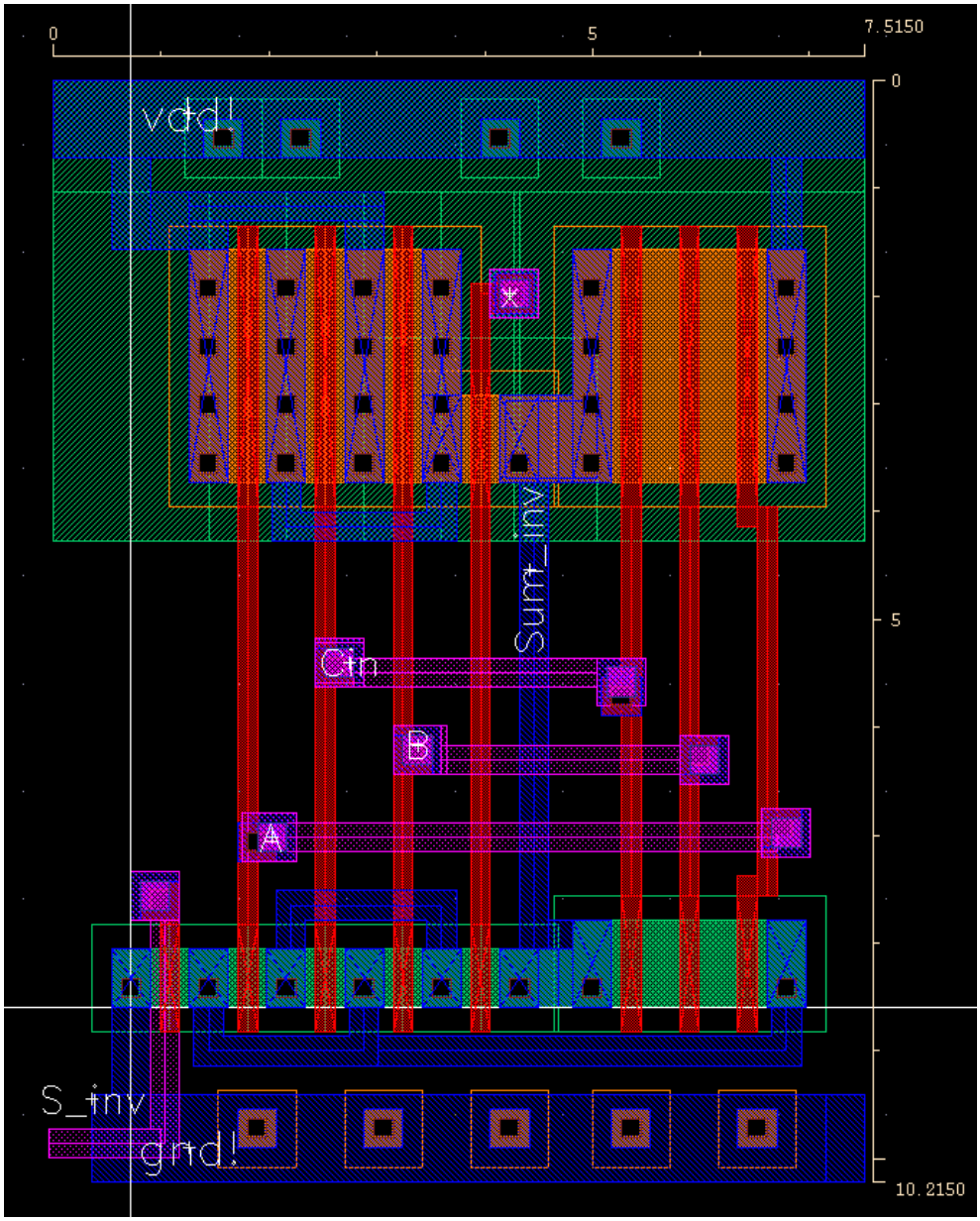
A.4.a.i. Zigzag approach Full Adder Cout' schematic.



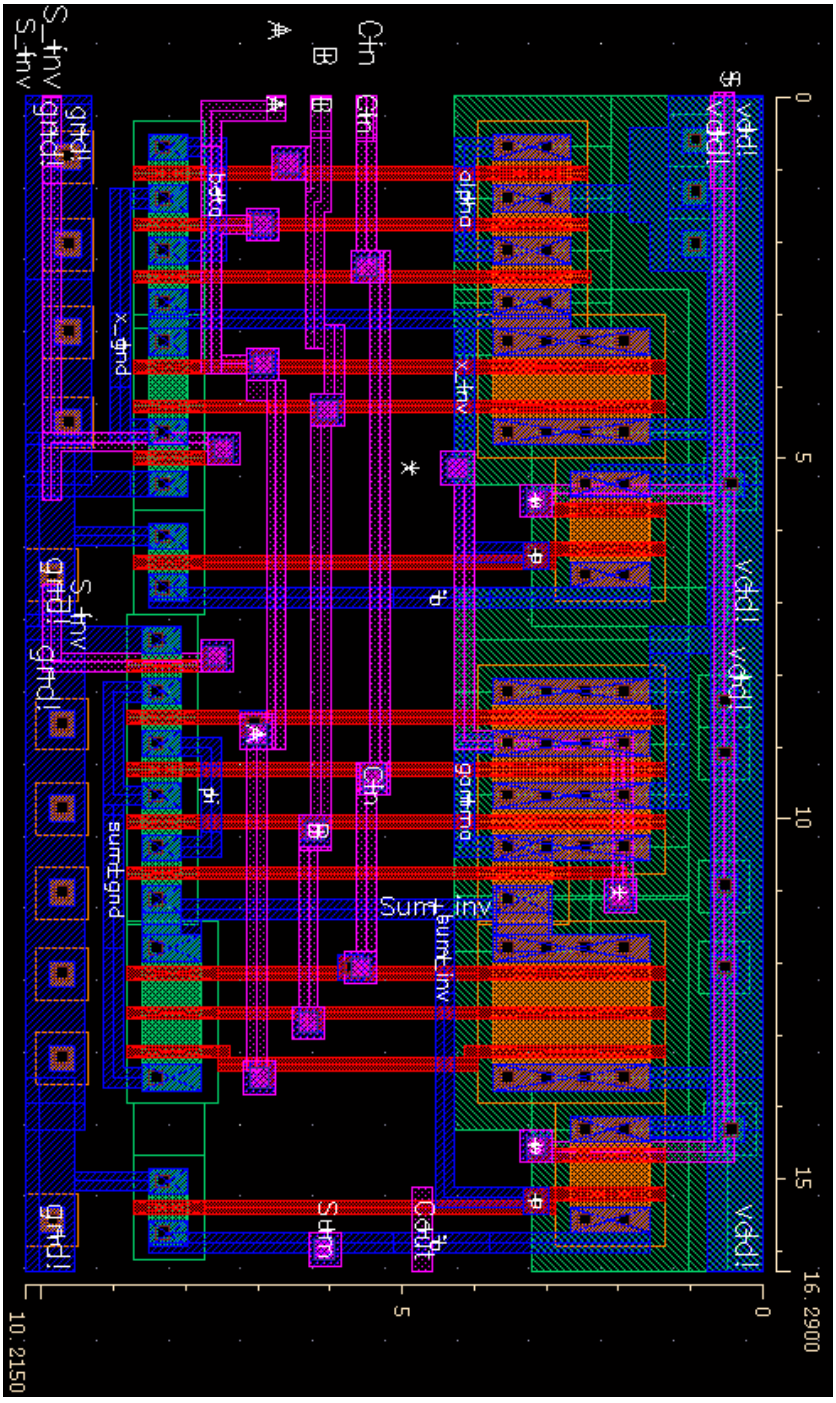
A.4.a.ii. Zigzag approach Full Adder Sum' schematic



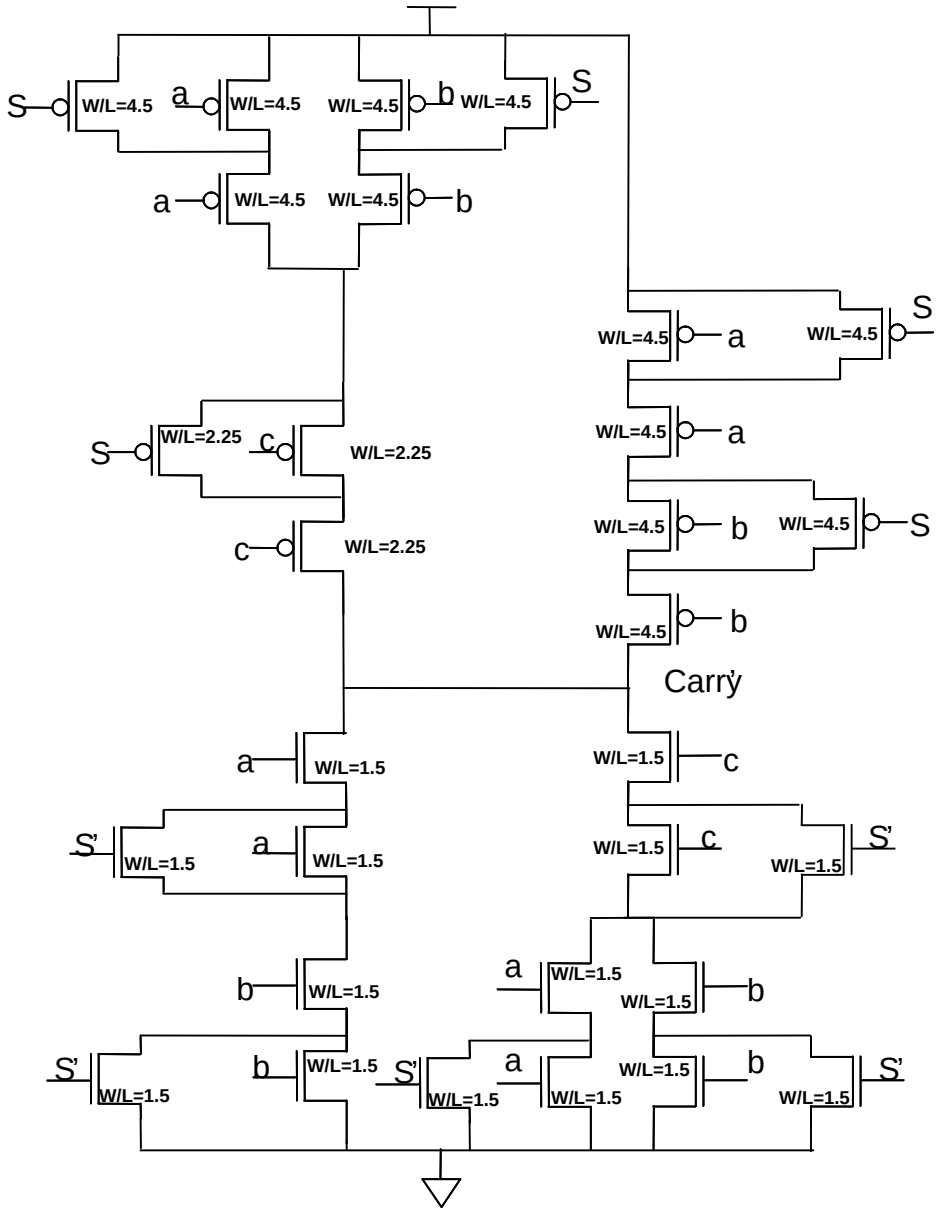
A.4.b.i. Zigzag approach Full Adder Cout' layout



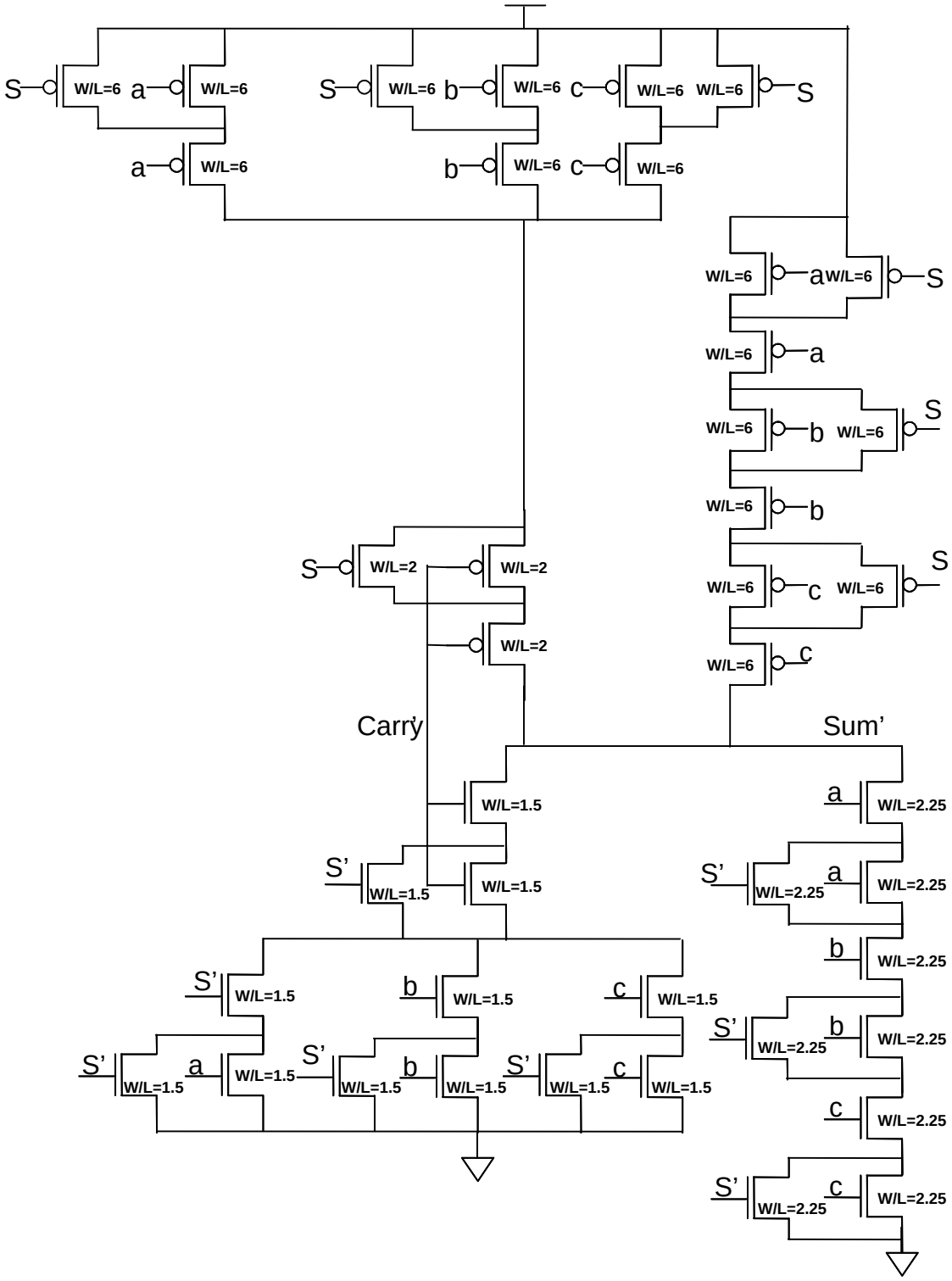
A.4.b.ii. Zigzag approach Full Adder Sum' layout



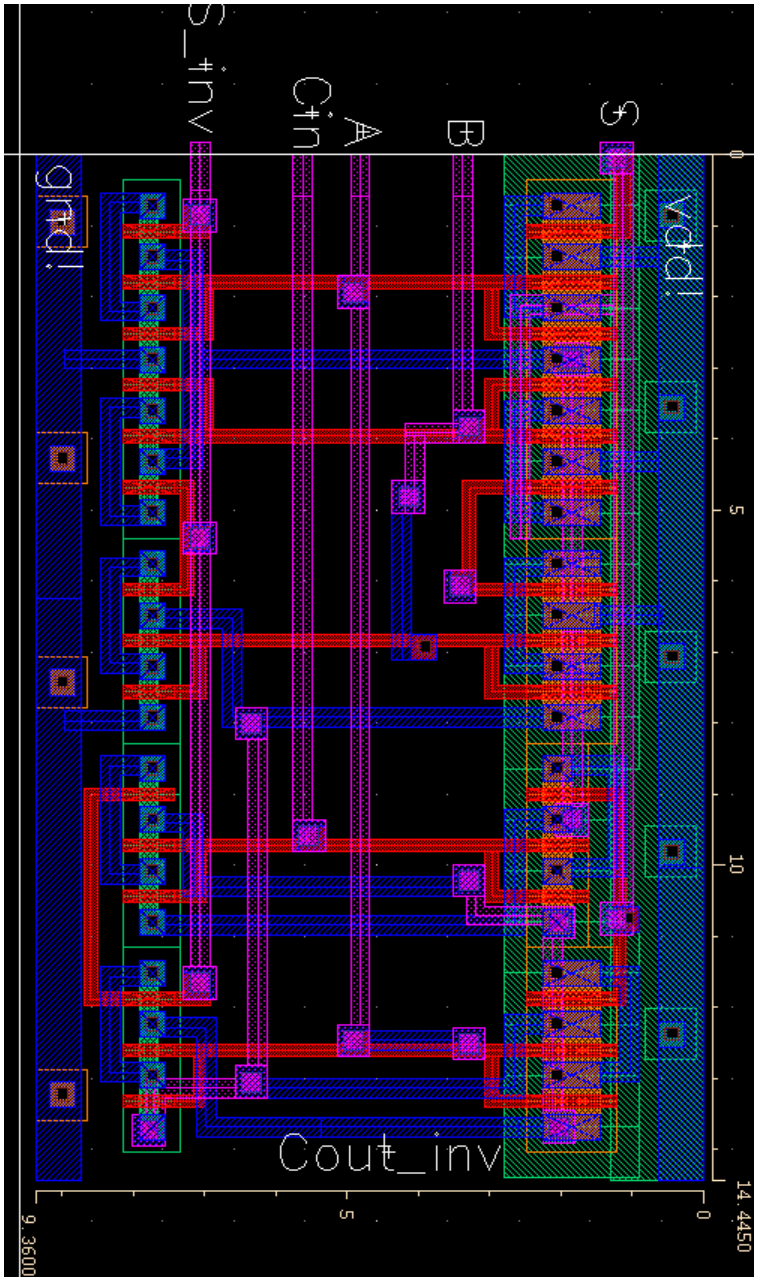
A.4.b.iii. Zigzag approach Full Adder layout



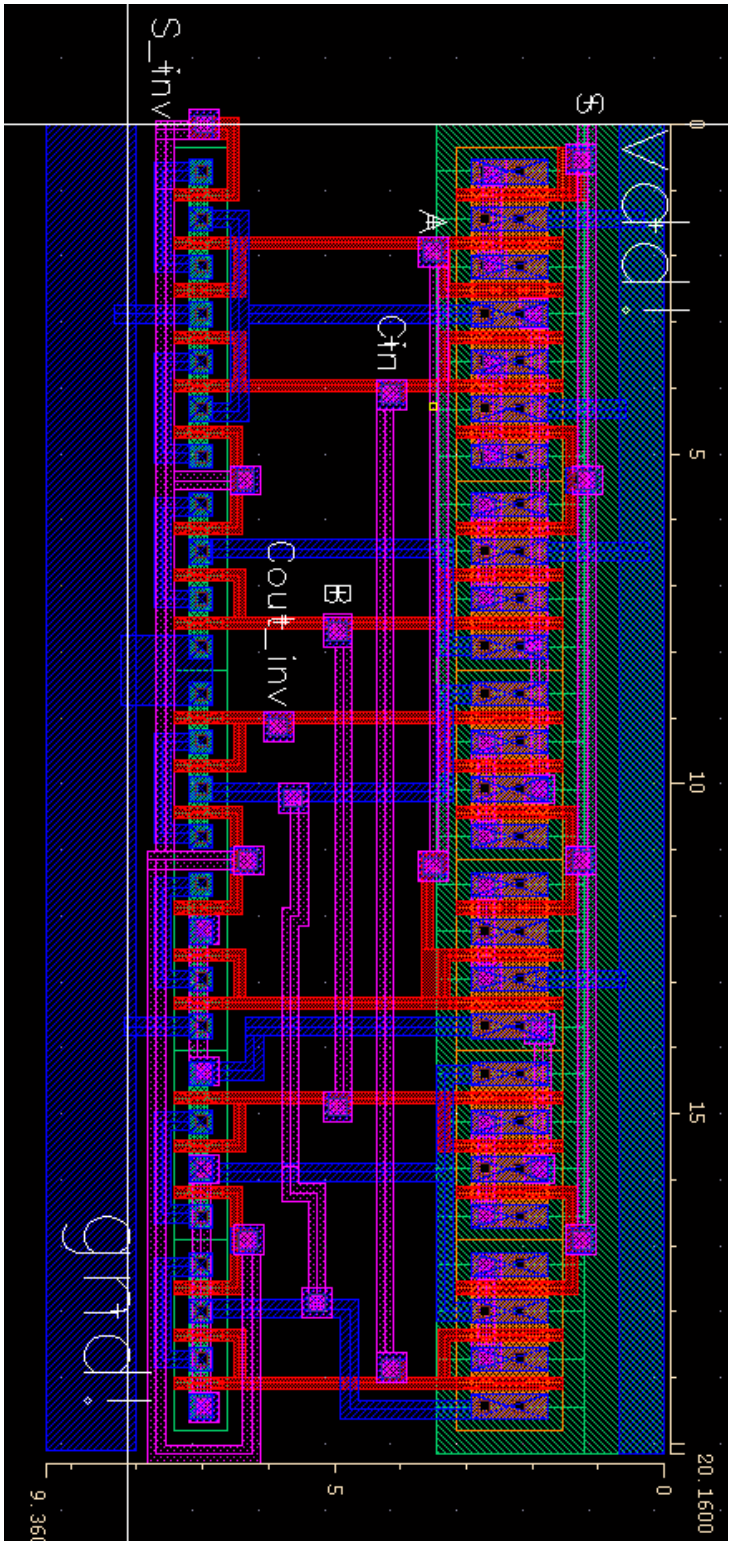
A.5.a.i. Sleepy stack approach Full Adder Cout' Schematic



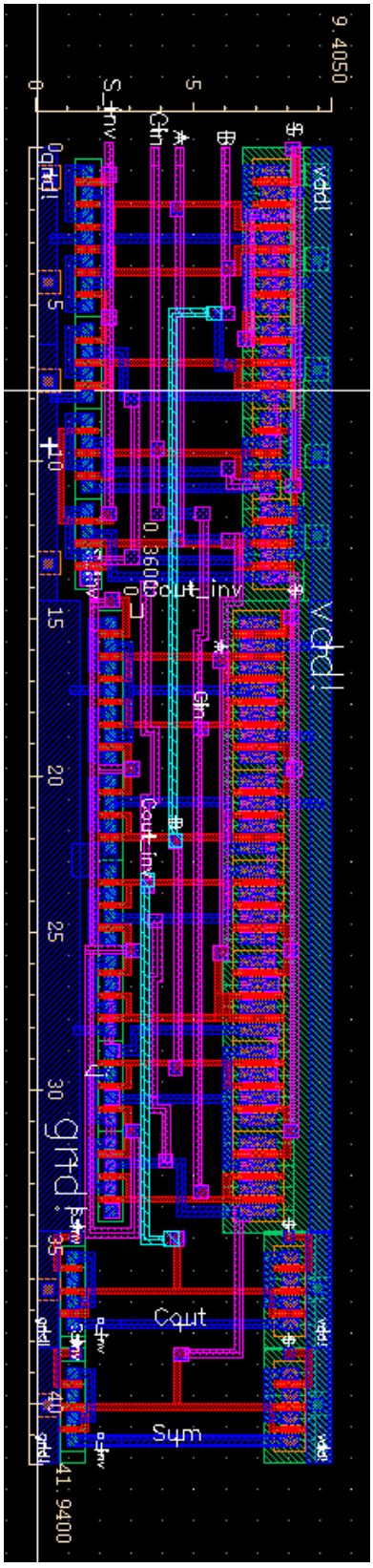
A.5.a.ii. Sleepy stack approach Full Adder Sum' Schematic



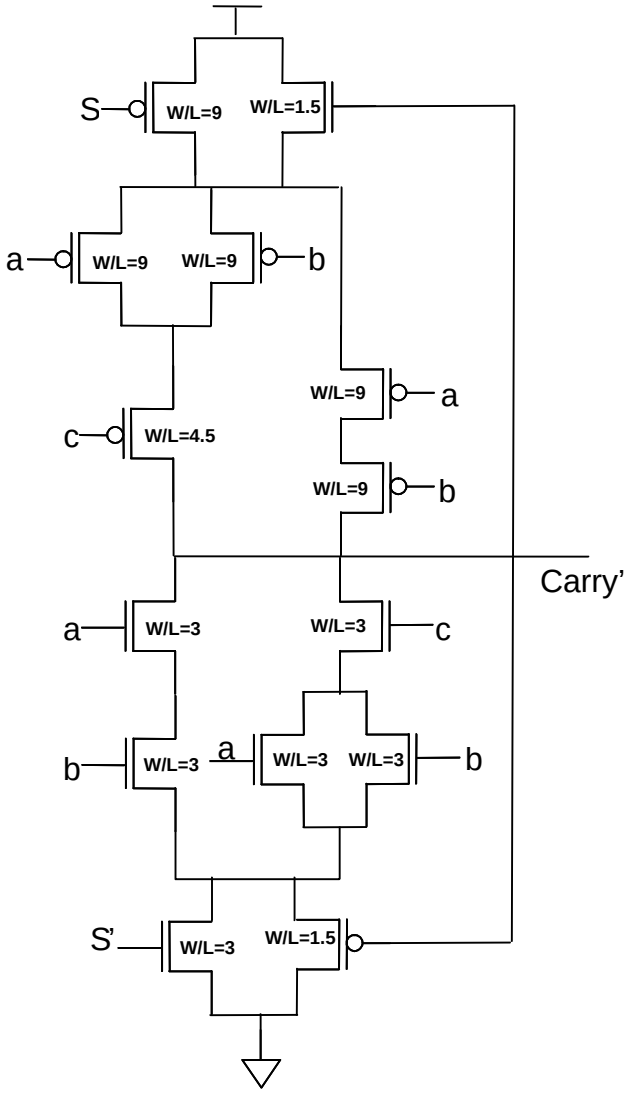
A.5.b.i. Sleepy stack approach Full Adder Cout' Layout



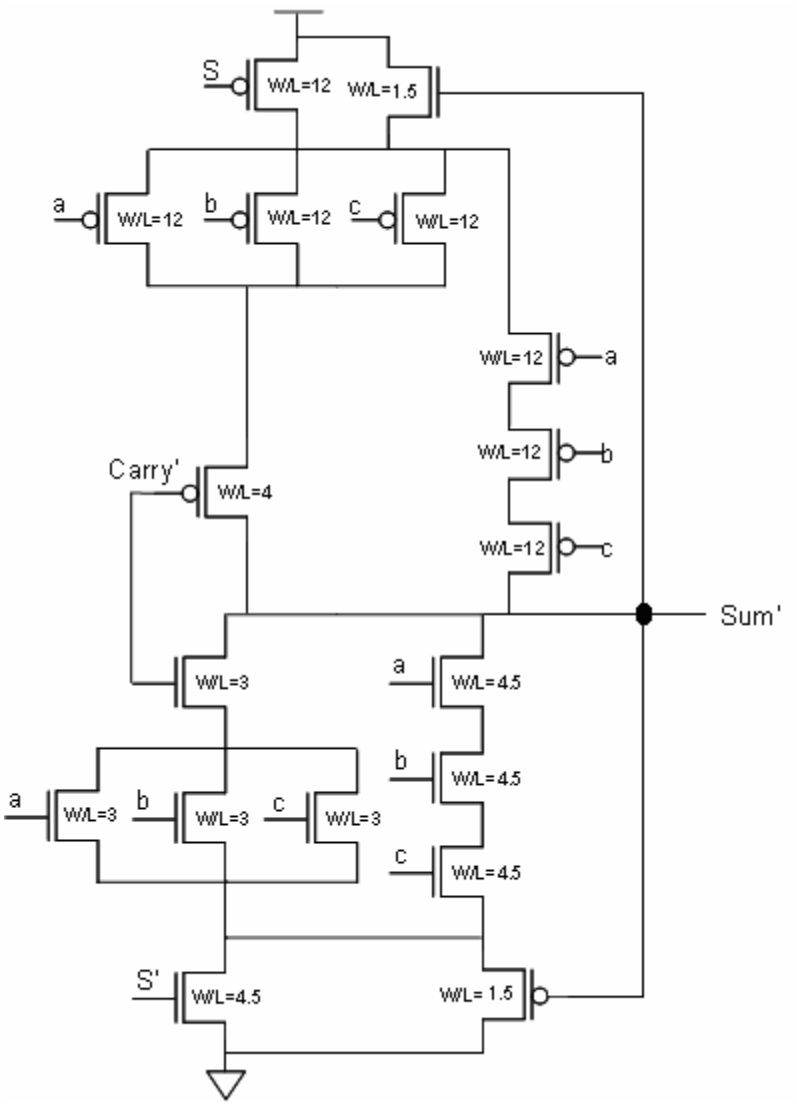
A.5.b.ii. Sleepy stack approach Full Adder Sum' Layout



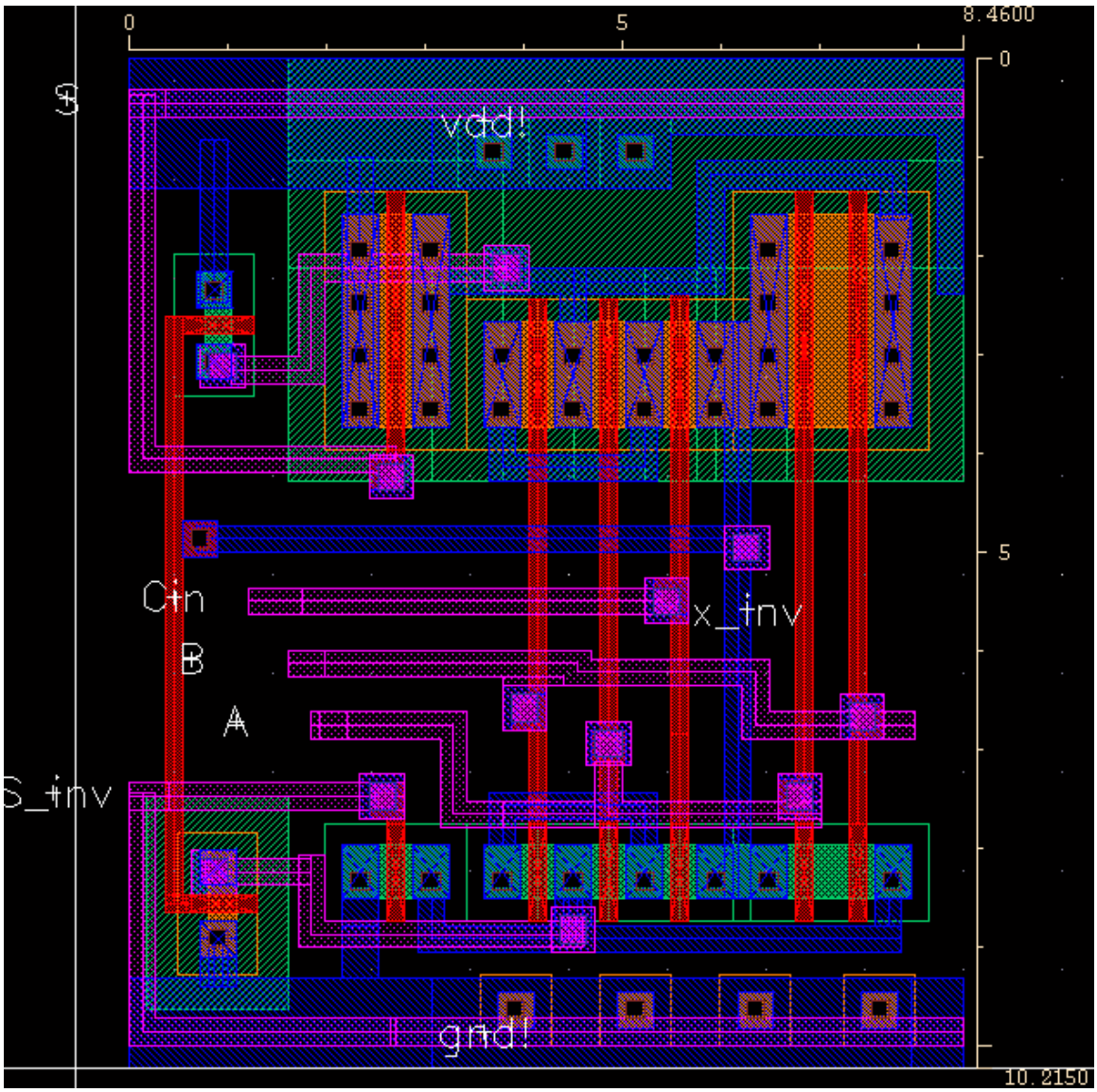
A.5.b.iii. Sleepy stack approach Full Adder Layout



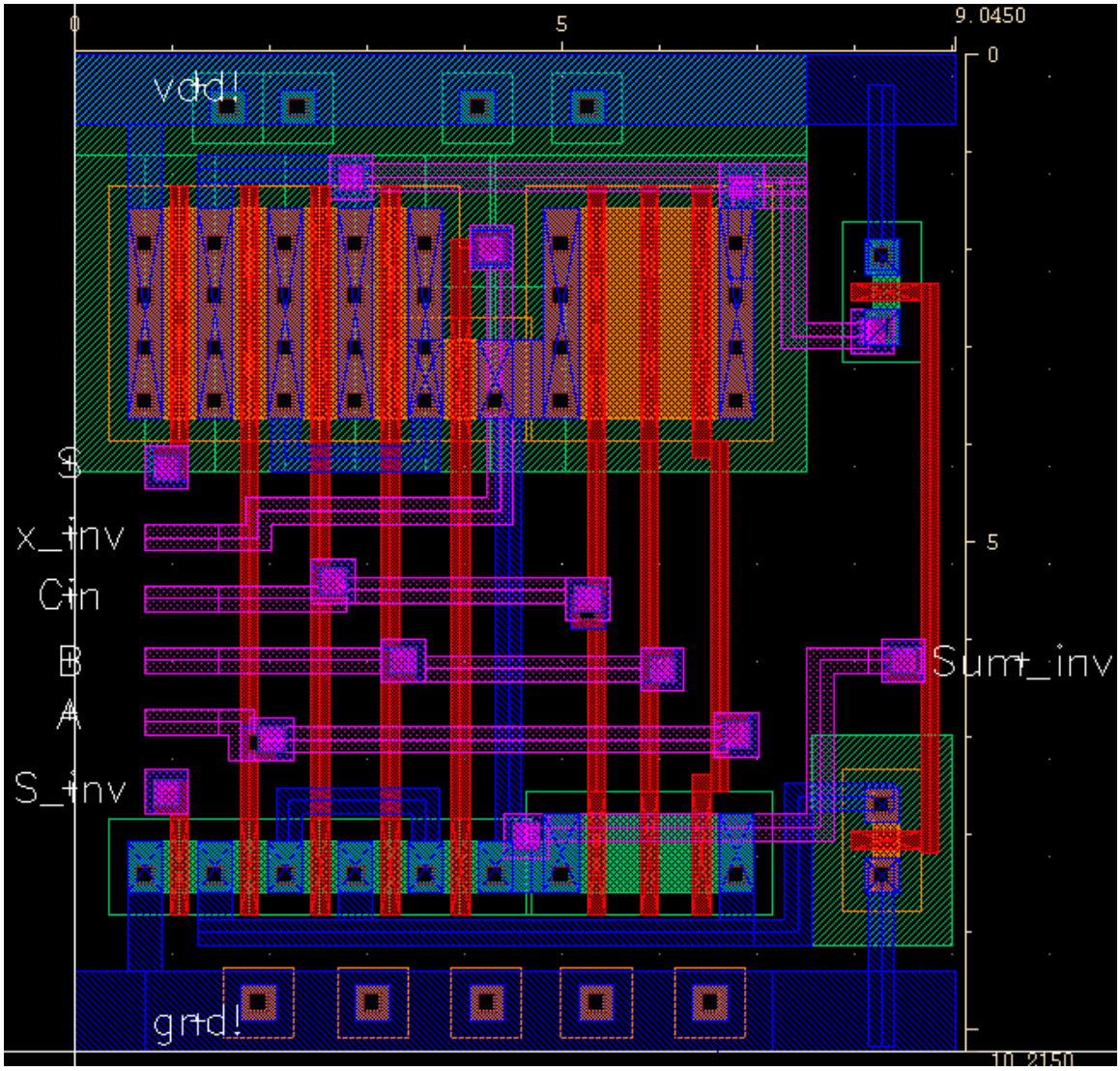
A.6.a.i. Sleepy keeper approach Full Adder Cout' schematic



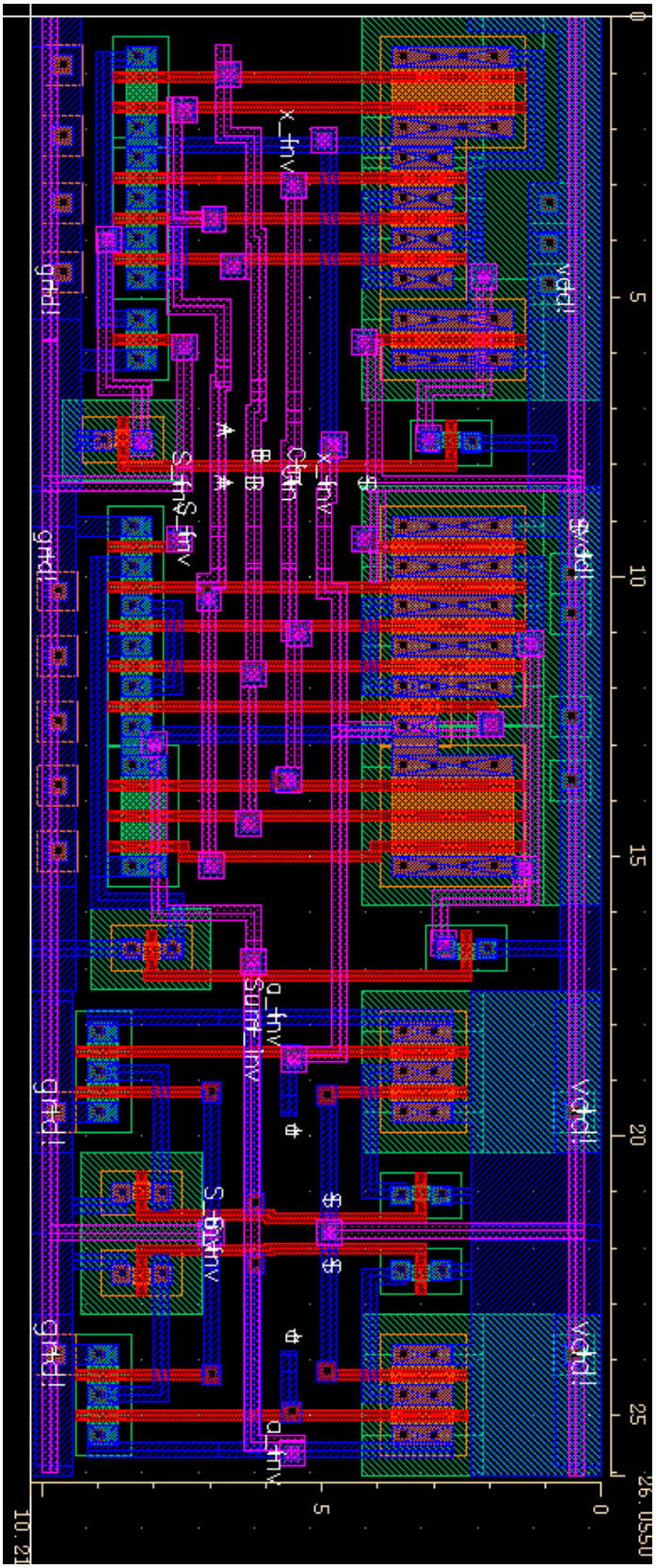
A.6.a.ii. Sleepy keeper approach Full Adder Sum' schematic



A.6.b.i. Sleepy keeper approach Full Adder Cout' Layout



A.6.b.ii. Sleepy keeper approach Full Adder Sum' Layout



A.6.b.iii. Sleepy keeper approach Full Adder Layout

Appendix B: 4-bit adder results

TSMC 0.18µm	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ^2)
Base case	7.05E-10	3.89E-10	1.48E-04	552.06
Stack	1.70E-09	2.23E-10	1.27E-04	743.94
Sleep	9.49E-10	1.15E-10	1.49E-04	743.94
ZigZag	9.42E-10	5.49E-11	1.40E-04	664.11
Sleepy Stack	1.35E-09	1.77E-10	1.25E-04	1584.05
Sleepy Keeper	1.02E-09	1.16E-10	1.66E-04	1064.61
Sleep (dual Vth)	1.07E-09	3.70E-11	1.50E-04	743.94
ZigZag (dual Vth)	1.07E-09	1.21E-11	1.40E-04	664.11
Sleepy Stack (dual Vth)	1.47E-09	3.44E-11	1.19E-04	1584.05
Sleepy Keeper(dual Vth)	1.20E-09	2.78E-11	1.68E-04	1064.61

Berkeley 0.18µm	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ^2)
Base case	5.06E-10	3.08E-08	1.38E-04	552.06
Stack	1.50E-09	3.05E-09	1.17E-04	743.94
Sleep	6.78E-10	4.73E-09	1.39E-04	743.94
ZigZag	6.83E-10	2.51E-09	1.32E-04	664.11
Sleepy Stack	1.18E-09	4.43E-09	1.22E-04	1584.05
Sleepy Keeper	7.39E-10	4.57E-09	1.51E-04	1064.61
Sleep (dual Vth)	7.96E-10	4.01E-11	1.43E-04	743.94
ZigZag (dual Vth)	8.07E-10	8.12E-12	1.33E-04	664.11
Sleepy Stack (dual Vth)	1.34E-09	2.89E-11	1.16E-04	1584.05
Sleepy Keeper(dual Vth)	8.73E-10	3.86E-11	1.55E-04	1064.61

Berkeley 0.13µm	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ^2)
Base case	4.86E-10	1.45E-08	4.25E-05	316.75
Stack	1.43E-09	8.13E-10	3.71E-05	426.85
Sleep	6.32E-10	1.77E-09	4.28E-05	426.85
ZigZag	6.33E-10	1.07E-09	4.07E-05	381.04
Sleepy Stack	1.09E-09	1.35E-09	3.71E-05	908.88
Sleepy Keeper	6.98E-10	1.50E-09	4.68E-05	610.84
Sleep (dual Vth)	7.77E-10	2.67E-11	4.36E-05	426.85
ZigZag (dual Vth)	7.71E-10	2.82E-12	4.08E-05	381.04
Sleepy Stack (dual Vth)	1.26E-09	1.66E-11	3.49E-05	908.88
Sleepy Keeper(dual Vth)	8.53E-10	2.73E-11	4.77E-05	610.84

Berkeley 0.10µm	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ^2)
Base case	4.00E-10	3.74E-08	1.90E-05	187.43
Stack	1.20E-09	2.23E-09	1.63E-05	252.57
Sleep	5.46E-10	4.29E-09	1.92E-05	252.57
ZigZag	5.38E-10	2.33E-09	1.83E-05	225.47
Sleepy Stack	9.10E-10	3.52E-09	1.64E-05	537.80
Sleepy Keeper	5.95E-10	4.27E-09	2.11E-05	361.44
Sleep (dual Vth)	7.05E-10	1.52E-11	1.96E-05	252.57
ZigZag (dual Vth)	6.93E-10	4.71E-12	1.83E-05	225.47
Sleepy Stack (dual Vth)	1.15E-09	1.66E-11	1.53E-05	537.8
Sleepy Keeper(dual Vth)	7.91E-10	1.46E-11	2.17E-05	361.44

Berkeley 0.07µm	Propagation delay (s)	Static Power (W)	Dynamic Power (W)	Area (μ^2)
Base case	3.76E-10	8.90E-08	8.63E-06	91.84
Stack	1.16E-09	6.83E-09	7.41E-06	123.76
Sleep	5.38E-10	1.36E-08	8.77E-06	123.76
ZigZag	5.25E-10	9.09E-09	8.37E-06	110.48
Sleepy Stack	8.64E-10	1.08E-08	7.39E-06	263.52
Sleepy Keeper	5.90E-10	1.30E-08	9.71E-06	177.11
Sleep (dual Vth)	7.52E-10	3.65E-11	9.03E-06	123.76
ZigZag (dual Vth)	7.43E-10	2.19E-11	8.46E-06	110.48
Sleepy Stack (dual Vth)	1.24E-09	3.50E-11	7.06E-06	263.52
Sleepy Keeper(dual Vth)	8.30E-10	3.89E-11	1.00E-05	177.11