Application of Logical Effort on Design of Arithmetic Blocks

Xiao Yan Yu*,**, Vojin G. Oklobdzija*,**
*ACSEL Laboratory
Electrical and Computer Engineering
Department University of California, Davis

William W. Walker**
**Advanced LSI Research
Fujitsu Laboratory of America
Sunnyvale, California

Abstract

In this paper, we review the logical effort model presented in [1]. Based on the HSPICE simulation results using 0.18μm, CMOS technology as applied to logic blocks used in arithmetic circuits; we analyze the efficiency of the model and also present modifications that include modeling of wire delay. We propose a new model for logical effort that will better fit the behavior of these blocks. The results are applicable for evaluation of arithmetic units as well as for development of new arithmetic algorithms. Our ultimate objective is to close the gap between arithmetic algorithms and their performance in VLSI CMOS.

1. Introduction

Sutherland and Sproull [1] presented a simple logical effort (LE) delay model $d = \tau(g \times h + P_{inv})$, where τ is the intrinsic delay of an inverter, g is the logical effort of the gate or how well the gate drives current in compare to a minimal sized inverter, which is the ratio of the on-resistance of the gate to the on-resistance of an inverter with the same input capacitance, h is the electrical effort of the gate, which is the ratio of load capacitance to input capacitance, and τP_{inv} is the parasitic delay of the gate. After unnormalizing, the delay model for LE can be seen to be simply $d = t_0 + R \times C_{load}$ where t_0 is the intrinsic delay of the gate, which is equivolent to τP_{inv}, R is its on-resistance value and C_{load} is the load capacitance. Table 1 provides a list of theoretical logical effort values for NAND and NOR gates assuming the optimal ratio of PMOS transistor size to NMOS transistor size is 2 for an inverter.

Table 1: Theoretical LE Values for NAND and NOR gates

Gate Type	Logical Effort	Formula	N=2, pn=2	N=3, pn=2	N=4, pn=2
NAND	Total	N(n+pn)/(1+pn)	8/3	5	8
	Per input	(n+pn)/(1+pn)	4/3	5/3	2
NOR	Total	N(1+n*pn)/(1+pn)	10/3	7	12
	Per input	(1+n*pn)/(1+pn)	5/3	7/3	3

The theoretical values for parasitic delays of NAND and NOR gates, is i x P_{inv}, where i represents the number of inputs to the logic gate as presented in [1]. We simulated the 2-input, 3-input, 4-input NAND gates and 2-input NOR gates in HSPICE and comparing the result with the theoretical value presented in [1]. Table 2 shows our simulation results in compare with the theoretical values using 0.18μm, 1.8V CMOS technology. There are two cases to simulate for each logic gate, inner controlled and outer controlled. The inner controlled case is which the input to the transistor that is near the output switches and the outer controlled case is which the input to the transistor that is furthest to the output switches.

From the table, it can be seen that the results shown does not really match the theoretical value. This is because the model did not include the effects on the delay times by the transition times at the input. The delay time should increase as the output load increases and decreases as the output load decreases due to the rise and fall time impact. In order to improve accuracy, we proposed a new model that will more accurately estimate the delay in arithmetic blocks, which includes the input transition time factors.

Table 2: Logical Effort Values from theory and simulation

	Logical Effort (g)		
	Theoretical	Our Simulation	
		Inner Controlled	Outer Controlled
NAND2	1.33	1.3	1.1
NAND3	1.67	1.4	1.3
NAND4	2.00	1.8	1.5
NOR2	1.66	1.5	1.4
	Parasitic Delay (Pinv)		
	Theoretical	Our Simulation	
		Inner Controlled	Outer Controlled
NAND2	2.00	1.3	2.4
NAND3	3.00	2.3	4.7
NAND4	4.00	2.4	7.5
NOR2	2.00	1.9	3

2. Modified Logical Effort Model

The LE model is good for insight and quick hand calculations, but its accuracy is limited. To improve the accuracy, we add a linear input transition time dependence to the LE model, and call it the extended logical effort model (XLE). XLE expresses delay as $d = t_0 + RC_{load} + K_{tran}t_{tran}$ where t_{tran} is the 10-90% input transition time and K_{tran} is a dimensionless model coefficient. An additional equation $t_{tran} = ttran_0 + R_tC_{load}$ is used to model the output transition time, where $ttran_0$ and R_t are model coefficients. Separate equations and coefficients for up and down transitions are used, as well as separate equations for each pin. This addition is beneficial because delays are a strong function of transition times and input pin. The transition times depend on the output loading directly. Therefore, rise and fall times for same logic gates are different in a given arithmetic circuit because the loading on different nodes differs. The LE model does not include impact on delay time from variations of transition times. This then, compare to the new XLE model is less accurate. The XLE model is a subset of the Synopsys CMOS2 Delay Model [3]. To size a path using the XLE model, we first derive the model coefficients for a set of arbitrary unit size gates (using SPICE simulation and linear regression) then assign scale factors, α_i, i=1..n, to each of the n gates in the path. When the size of a gate increases by a scale factor α_i, the resistances in the model scale down by α_i and the load capacitances scale up by α_i. Optimization proceeds by selecting the set α_is that minimize the path delay using any appropriate algorithm.

3. Simulation Results:

Table 3 shows average percentage error of delay time by using the regular absolute percentage error formula, $\left|\frac{ActualDelay - ModelDelay}{ActualDelay}\right| \times 100\%$, between model and HSPICE simulation result. Pin name "a" represents delay from bottom transistor of the transistor stack; "b" represents delay from second to bottom transistor of the transistor stack and so on. The simulations results are the delay time from the inputs of the logical blocks to the outputs by varying input transition times and the output capacitive load. Since the logical effort (LE) model did not include the influence on delay time by variation in input transition delays, the overall average error of it is about 23.3%. Our extended logical effort model (XLE) is about 2.2%. The results show that our method is almost 10 times more accurate than the original LE model. This accuracy will provide us with better optimization solutions.

Table 3: Average Timing Error from HSPICE results of LE and XLE

		Average Percent Errors	
Logical Block Name	**Pin Name**	**LE**	**XLE**
INV	A	33.5%	4.1%
NAND2	A	30.5%	3.7%
	B	24.3%	2.4%
NAND3	A	27.5%	2.5%
	B	21.6%	1.8%
	C	16.9%	1.4%
NAND4	A	25.9%	1.9%
	B	19.9%	1.3%
	C	15.5%	1.2%
	D	12.9%	1.1%
NOR2	A	27.8%	2.9%
	B	22.7%	1.9%
Overall Average Error		**23.3%**	**2.2%**

4. Wiring Delay:

The wiring RC delay is becoming much more important as technology features scales down bellow 0.1u. However, in [1], the model did not provide any estimation of the wiring delay. Therefore, it is very important to include this parameter in the delay equation. Two models available to model the wires are shown in Fig. 1(a) and 1(b) and also shown in Fig. 8.31 in [3].

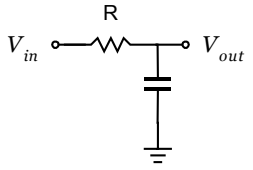

Fig. 1(a). Lumped Model

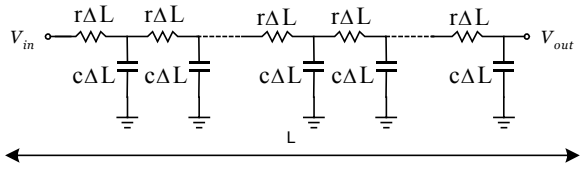

Fig. 1(b). Distributed Model

If we assume that the size of the floorplan of the input arithmetic circuit is minimal, then we can model the wire as an extra load capacitor on each node for simplicity. When we scale up the arithmetic block by scale factor of α, assume the width of the wire does not change, the modeled wire capacitance, or the extra load capacitor will need to be increased by α.

5. Some guidelines on arithmetic circuit design

Previously, the critical path delay of an arithmetic circuit is measured by gate delays assuming identical gate delay for each type of gate. Now, the delay of a gate or block is determined by its number of fan-in, size and how much loading it has to drive. To design an arithmetic circuit, a detailed floor plan with initial sizing is necessary. From the floor plan, the wiring capacitance can be estimated to be included as extra load capacitors in the delay model for delay analysis.

To optimize the circuit for higher performance, one can use the delay model proposed in this paper to properly size all the transistors in the circuit to minimize delays on the paths. The critical path delay will be the sum of the delays of the blocks that are on the critical path. Fig. 2 shows the theoretical critical path for a simple 8-bit Carry Skip Adder based gate delay measures. However, as mentioned before, the gate delay measures are not acurate. Therefore, the critical path delay should results from simulation with logical blocks properly sized on every path. To size the blocks, an iterative approach is required. Fig. 3 shows an algorithm to optimize the circuit.

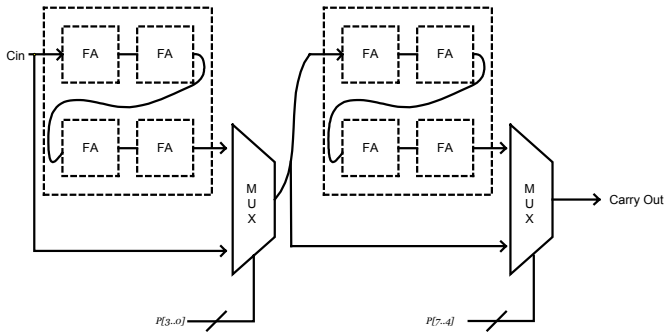

Fig. 2. A regular 8-bit Carry Skip Adder

```
CurrentCritPath = FindCriticalPath();
PrevCritPath = CurrentCritPath;
While( TRUE ) {
   EstimateSideCap( CurrentCritPath );
   SizeCriticalPath(CurrentCritPath);
   CurrentCritPath = FindCriticalPath();
   if( PrevCritPath == CurrentCritPath ) {
      BreakOutOfLoop;
   }
   else {
      PrevCritPath = CurrentCritPath;
   }
}
```

Fig. 3. A suggested algorithm for circuit optimization

The algorithm suggested in Fig. 3 relies on static timing analysis to find the critical path of a given arithmetic circuit. Once the critical path is provided, then some estimation of the off-path capacitances for each node that are on the critical path is needed. Using the load capacitance on each node, then we can express total delays of the path as variable sizes for each block that is on the critical path. Using linear regression method, then we can find the solution to the sizes that minimizes the delay time. With the new sizes, the static timing analyzer will be able to produce another critical path. If the new critical path is the same as the old one, then we know we have an optimized circuit because we cannot optimize the critical path further. With this algorithm, we were able to improve our 8-bit adder timing by 30%.

Conclusion

The original logical effort model has been studied and tested using 0.18 μm CMOS technology. An extended model has been introduced and compared against the original one. It improves by including the effect on delay time from transition times at the input. The simulation results show that the new model exceeds the original model on delay accuracy by 10 times. This result is very beneficial to circuit optimization since a more accurate model will bring closer to the best implementation of a particular arithmetic circuit. At the end of this paper, some guidelines on designing and optimizing arithmetic circuits have been introduced.

References

[1] I.E. Sutherland and R.F. Sproull, "Logical Effort: Designing for Speed on the Back of an Envelope", in C.H. Sequin, Ed., *Advanced Research in VLSI*. Cambridge, MA: MIT Press, 1991.

[2] V. G. Oklobdzija, *"High-Performance System Design: Circuits and Logic",* IEEE Press, February 1999.

[3] Jan Rabaey, "Digital Integrated Circuits: A Design Perspective", Prentice Hall, 1996.

[4] Synopsys Library Documentation.

[5] H. Dao and V. G. Oklobdzija, "*Comparative Delay Analysis of Representative Adders Using Logical Effort Technique*", 35th Asilomar Conference on Signals, Systems and Computers, 2001.

[6] H. Dao and V. G. Oklobdzija, "*Application of Logical Effort on Delay Analysis of 64-bit Static Carry-Look-ahead Adder*", 35th Asilomar Conference on Signals, Systems and Computers, 2001.