Design and Development of Logical Effort Based Automated Transistor Width Optimization Methodology

Sapthi Chandra Tiwari, Kunwar Singh and Maneesha Gupta

1Department of ECE N.S.I.T.(University of Delhi), Delhi, India
2Department of EE, Delhi Technological University, Delhi, India
3Department of ECE N.S.I.T.(University of Delhi), Delhi, India

Abstract: Optimization and characterization of transistor widths has been always a very important task in digital design process. The actual potential of a circuit cannot be fully unleashed unless it is fairly optimized and a better circuit may give poor results. Moreover, as the loading conditions vary it becomes necessary to re-optimize the circuit for that load. LM algorithm and Logical Effort theory are generally used for the optimization purpose. For transistor width optimization the most reliable method is Logical Effort theory. Logical Effort theory is loosely based on hand calculation hence it is tedious, time consuming and error prone. The paper presents a new automated transistor width optimization methodology for SoC. The methodology is based on Logical Effort theory. The proposed methodology is completely automation based and uses different procedural blocks written in TCL (tool command language). The methodology takes SPICE netlist as input and optimizes transistor widths for minimum delay. Both sequential (flip-flop) and combinational (basic logic gates) logic blocks were optimized successfully using the proposed methodology. Moreover the paper also presents comparison between LM and LE theory.

Key words: Flip-Flop • Low power • VLSI • Optimization • Logic Effort

INTRODUCTION

In today’s world there is extensive demand of mobile devices which are faster, better and having less power consumption as compared to their predecessors. Technology scaling has reached up to 16nm and further scaling has many issues like cross talking, tunneling, leakage etc. Now to proceed further we need to better optimize the present circuits at available technologies. Optimization of a circuit directly involves transistor width sizing/optimization. Transistor width optimization has been always a very critical and tedious task in front of Digital IC designers. In modern era, the transistor width optimization is required in library cell characterization. Moreover multicore ICs are in existence and the density has increased many folds i.e. same chip area now contains two to four times more logic cells. Also there is increased need of on chip communication in multicore ICs. Hence it can be said that the cell library is very heavily loaded and its characterization requires a huge amount of time. Until now the optimization is performed using either Levenberg-Marquardt algorithm or Logical Effort theory. Several research papers have been proposed in past related to both, but none of them proposed a completely automated methodology. Both the algorithms have their own drawbacks. The optimization methodologies based on LM algorithm, is a mathematical non-linear optimization technique and while optimizing it does not take into account electronics principles. Whereas the optimization methodology based on LE theory is not embedded in SPICE, hence it involves time consuming and error prone pen paper calculations [1,2]. Since LE theory is not embedded in SPICE automated optimization is a distant dream. Whereas even though LM algorithm is embedded in SPICE, the previously proposed methodologies utilize iterative procedure for optimization hence creating requirement of automation [3,4]. The paper proposes two completely automated optimization algorithms based on both LE theory and LM algorithm. Results were obtained, for both sequential (flip-flop) and combinational blocks (basic logic gates), using the proposed algorithms.

Corresponding Author: Sapthi Chandra Tiwari, Department of ECE N.S.I.T. (University of Delhi), Delhi, India
The paper is organized as follows: Section II presents
detailed analysis LM theory. Section III illustrates detailed
analysis of automated optimization methodology based
on LE theory. The results and analysis is presented in
section IV. Finally, a conclusion is presented in the last
section.

Levenberg-Marquardt Algorithm: LM algorithm
embedded in SPICE was originally proposed by K.
Levenberg [5] and D.W. Marquardt [6]. The algorithm is
based on iterative procedure that finds minimum of a
multivariable function, that is expressed as sum of squares
of non-linear real valued functions (It is commonly used
for non-linear real valued functions) [7-9].

LM algorithm is largely a combination of steepest
descent and Gauss-Newton method. If the current
solution is far from the correct value, the algorithm
behaves like steepest descent method; hence it is slow,
but guaranteed to converge. If the current solution is
closer to correct solution it behaves like Gauss-Newtonian
method. Consider 'f' to be an assumed functional solution
which maps a parameter $P \in \mathbb{R}^n$ to an estimated
measurement vector $X = f(P)$, $X \in \mathbb{R}^n$. An initial value
parameter estimate $P_0$ and a measured vector 'X' are
provided and it is desired to find vector $P^+$ that best
satisfies the functional relation $f$, so as to minimize
squared distance etc. where

$$e = X - \hat{X}$$  \hspace{1cm} (1)

The very basis of LM algorithm is a linear
approximation to 'f' in the neighbourhood of 'P'. The in
depth solution of LM algorithm can be obtained from
[7-9].

Theory Based Optimization Methodology

Logical Effort Theory: Logical effort theory is based on
delays caused by the capacitive loads that the logic gate
drives and by the topology of the logic gate. It is well
known as the load increases, the delay increases and
again the delay is also dependent on logical function of
gate. Logical effort theory uses inverters as basic block
and compares the driving capabilities of other gates with
it. Hence a logic gate which requires some transistors in
series will be slow as compared to inverter having similar
transistor width and loading conditions. Therefore NAND
gate will have more delay as compared to inverter [10].

Logical effort theory expresses the absolute delay as
the product of unit less delay $(d)$ of the gate and the basic
delay unit $(t)$ characterized by particular fabrication
process. 't' can also said to be delay of transistor at that
process.

d_{ns} = d^* t \hspace{1cm} (2)

typically $t$ is about 50ps for 0.6um process [10].

Now again $d$ can be divided into two parts:

- Fixed delay i.e. parasitic delay.
- Delay due to load on gates output (called effort delay
  or stage effort 'f').

$$d = f + p \hspace{1cm} (3)$$

The effort delay 'f' is dependent on load and
properties of logic gate driving that load.

$$f = g^* h \hspace{1cm} (4)$$

Where 'g' is the logical effort and 'h' is the electrical
effort.

The logical effort 'g', hence represents how much
worse the gate is in producing output current as
compared to inverter, given that all other parameters are
same. Electrical effort 'f' defines the effect of electrical
environment of logic gate on performance and effects of
size of transistors on load driving capability.

$$h = c_{out}/c_{ns} \hspace{1cm} (5)$$

Where $c_{out}$ is the output load capacitance and $c_{ns}$ is
the capacitance presented by logic gate at one of its input
terminal. Hence,

$$d = g^* h + p \hspace{1cm} (6)$$

The backbone of complete logical effort theory is the
calculation of logical effort 'g'. Its calculation is based on
the fact that it gives inverter a logical effort equal to one.
Logical effort 'g' is unit less quantity i.e. all the delays are
measured relative to delay of simple inverter. The logical
effort equal to one for an inverter is based on the
following equation by Ivan Sutherland.

"The logical effort of a logic gate is defined as the
ratio of its input capacitance to that of an inverter that
delivers equal output current"

$$g_i = C_i / C_{inv} \hspace{1cm} (7)$$

Where, $g_i$ is logical effort of input 'b'; $C_i$ is the input
capacitance of every signal in input 'b'; and $C_{inv}$ is the
input capacitance of inverter having same driving
capability as the logic gate. Capacitance of transistor gate
is proportional to width 'w' and so its ability to produce
output current. Since mobility of electrons is more as compared to holes, in CMOS, pull up transistors must be wider as compared to pull down transistors to have same conductance. In case of inverter for simplicity the ratio of PMOS to NMOS transistor width is chosen equal to two. So the total sum of widths of PMOS and NMOS becomes equal to three. Hence by definition:-

$$g = \frac{3}{3} = 1$$ (8)

Similarly two input NAND gate will have $g = \frac{4}{3}$ for both pins and two input NOR gate will have $g = \frac{5}{3}$ for each input. Hence for an inverter having ‘c’ as its input load and ‘4c’ as its output load the delay can be calculated as:

- $h = 4c/c = 4$; $p = 1$ (for 0.6u process [10]);
- $g = 1$ for inverter
- $d = gh + p = 5$; since $t = 50ps$ (for 0.6u process)

Automated Optimization Methodology Based on Logical Effort: As it is evident from the previous section that the whole logical effort theory is dependent on calculation of ‘g’ which in turn is dependent on transistor widths, the proposed methodology optimizes transistor widths as per logical effort theory and follows the flow chart shown in Fig. 1. Several research papers have been proposed in the past which utilized logical effort theory for transistor widths optimization [11,1,2]. Since there is no algorithm based on LE theory embedded in SPICE hence they are dependent only on tedious hand calculations. The paper proposes automated optimization algorithm based on LE theory as shown in flow the chart shown in Figure 5.

The main objective of automation algorithm besides delay is the calculation of logical effort ‘g’ which in turn dependent on PMOS and NMOS transistor widths as said earlier. Once the optimized widths were obtained and hence ‘g’, the delay can easily be obtained by using equation (6). The complete automation has been implemented using different procedural code modules written in TCL (tool command language) which are finally accessed by top procedural module [12].

Algorithm Formulation: Logical Effort theory as discussed earlier is mainly dependent on calculation of logical effort ‘g’ and electrical effort ‘h’. Once the values of logical effort and electrical efforts are calculated it becomes very easy to calculate the optimized transistor widths for minimum delay. The logical effort ‘g’ for any input is dependent on following equation:-

---

Fig. 1: CMOS inverter

Fig. 2: Series combination of PMOS transistors.

Fig. 3: Series and parallel combination of PMOS transistors.
Logical effort = total PMOS transistor widths

Total NMOS transistor widths/3  \( (9) \)

Now the PMOS and NMOS widths can only be calculated when the topology of circuit or stage is known. For this we require a methodology by which we can be able to figure out the series and parallel connected transistors (i.e. number of PMOS transistors connected to vdd with series and parallel information and similarly for NMOS). We tackled this problem by taking SPICE netlist as our input (since SPICE netlist has the connectivity information as per our requirement). This can be more clearly understood by following example:-

Above shown is the schematic of inverter drawn on Mentor Graphics IC station. Now the netlist of the same is given below:-

- .CONNECT GROUND 0
- .Global VDD GROUND
- M2 VOUT VIN VDD VDD P L=0.18u W=1.4u M=1
- M1 VOUT VIN GROUND GROUND N L=0.18u W=1.4u M=1
- .END

We know that the order of terminals in SPICE are:-
- Transistor name Drain Gate Source Body L W
- Hence we can have the connectivity information from the netlist and based on that we can define the PMOS and NMOS transistor widths.

- Initial width calculation to obtain ‘g’

When we summarize the logical Effort theory in simple lines we will come to following conclusion:-

If there are three transistors connected in series there widths can be calculated by following formula:-

\[ \frac{1}{x} + \frac{1}{x} + \frac{1}{x} = \frac{1}{2} \]  \( (10) \)

\[ x = 6 \]

Similarly if there are N numbers of transistors connected in series there widths will be calculated as:-

\[ \frac{1}{x} + \frac{1}{x} + \ldots \ldots + \frac{1}{x} = \frac{1}{2} (N \text{ times}) \]  \( (11) \)

Similar is the case for NMOS transistors except the fact that the addition will be equal to one instead of two.

**Initial Transistor Width Fixation:** As mentioned in the earlier sections SPICE netlist is taken as input for the calculation of logical effort of circuit/stage. The SPICE
netlist will have the connectivity information of both the
PMOS and NMOS transistors. With the help of an
automation script written in TCL the NMOS and PMOS
transistors are stored in NMOS.txt and PMOS.txt
respectively which are connected to the output or
specified terminal. The transistors which are connected to
output terminal as well as vdd are moved to
Pout1+vdd.txt. Transistors that are connected to output
terminal and not with vdd are moved to Pout1.txt. Now
the widths of the transistors which are present in
Pout1+vdd.txt are made equal to two. For the transistors
present in Pout1.txt an automation script checks out the
name of their source terminal, suppose it comes out to be
‘y’ now an automation script checks for PMOS transistors
whose drain terminal is connected to ‘y’. Suppose a
transistor M7 was found whose drain is connected to ‘y’
now again the script checks out the name of M7’s source
terminal. If its source terminal is connected to vdd then
this transistor is moved to Pout2+vdd.txt and widths of
both the transistors are made equal to four. If M7’s source
terminal is not connected to vdd it is moved to Pout2.txt.
Similarly the process is repeated again and again unless
each of the PMOS transistors are covered. Moreover
similar process is repeated for NMOS transistors. At the
end all the Pout+vdd.txt and Nout+vdd.txt are
concatenated together in a single netlist file.

Special Cases: There are certain cases which are
uncovered by aforementioned mentioned methodology.
Those cases are:-

When there are parallel as well as series connection
of transistors in a design.

For these cases we need to mark node names in
netlist itself. Now when they will be processed with the
proposed methodology it will process them as:-

Now at the end we will have some transistors which
are present more than once. An automation script will
remove them. Hence every case can be covered with
the proposed methodology. Initially we have developed it for
a series of three transistors (which can be manipulated for
larger number with a slight alteration in script).

- Calculation of ‘g’ for individual inputs

After initial fixation of widths the netlist file will have
all the widths fixed according to the topology. Now to
calculate the logical effort of individual inputs we have
developed an automation script. It takes name of the
desired input terminal and separates all the transistors
(both NMOS and PMOS) whose gates are connected to
this terminal to a temporary file. Now all the width values
of transistors present in this file are added and divided by
three. Hence individual logic effort of all the input
terminals is obtained. This can be more clearly understood
from the following example:-

Suppose we need to calculate logical effort of an
input ‘VIN1’. Now a script separates out those transistors
whose gate terminals are connected to ‘VIN1’.

- M3 Z VIN1 VDD VDD P L=0.18u W=4 M=1
- M2 Y VIN1 Z VDD N L=0.18u W=2 M=1
- M1 X VIN1 Y VDD P L=0.18u W=8 M=1
- g = 4+2+8/3=4.67

Now width of all the transistors are added and
divided by three to obtain the logical effort.

Calculation of delay is directly based on the
equations (2)-(7).

Automation Flow: The whole automation process can be
summarized in following steps:

Step (1): SPICE net list is taken as input along with
input output node name information of each
stage (for multiple stage design) in critical path.
Step (2): The PMOS and NMOS transistors are
separated (using automation script) and stored
in different files for further processing.
Step (3): For PMOS file, transistors whose source is
connected to first output are stored in different
file. Now this file may contain several
transistors, their drain terminals are identified
one by one. If the drain terminal is connected to
V_{ds} its width value is fixed as two and stored in
final file else a counter is incremented and drain
terminal name along with this transistor is
stored.
Step (4): PMOS file is checked for transistor having
source connected to stored drain terminal
name, if found its drain terminal is checked. If
its drain terminal of this transistor is connected to
V_{ds} width of both transistors (previously
stored and the current one) are made equal to
four each. If not the counter is again incremented
and name of both transistors are
stored along with drain terminal name.
Step (5): The process above is repeated in loop in similar way until the drain terminal is not found connected to $V_{dd}$ and widths are fixed and stored to a different out file. The transistors whose width are fixed were removed from PMOS file. Hence in the end, PMOS file becomes empty and hence the loop stops.

Step (6): Similar process is repeated for NMOS file, while the only difference is that Gnd terminal is checked instead of $V_{dd}$ connected with source.

Step (7): When both NMOS and PMOS files have been processed, the output of both are concatenated in single out file.

Step (8): Since all the transistor widths are fixed, logical effort ‘$g$’ is calculated for each pin by adding the transistor widths connected to that pin and dividing by three. Other parameters such as electrical effort and ‘$p$’ are already known to the designer, hence $d$ can be calculated from eq. (6). And finally the absolute delay is obtained by $d_{ma} = d^*t$, where ‘$t$’ is obtained from process file.

RESULTS

The LM algorithm as explained earlier has requirement of minimum ($W_{min}$) and maximum ($W_{max}$) transistor widths for optimization algorithm. These values are determined by designer depending on area and delay trade off i.e. as $W_{max}$ increases the circuit delay decreases and vice versa. Moreover in case of logical effort theory the width values are dependent on input and output capacitive loads. Once the input and output capacitive values are fixed, the LE theory will give only a single unique solution for width values to achieve minimum delay.

Simulation Parameters
Transistor Width Range for Optimization: $W_{min} - W_{max}$ is the Width Range for LM Algorithm. The paper utilizes LE theory for the fixation of $W_{min}$ and $W_{max}$ values to be used in LM algorithm for optimization. The $W_{min}$ and $W_{max}$ are fixed as follows:

$$W_{max} = 2u$$
$$W_{min} = (2 \times \text{maximum transistor width obtained by LE theory})$$

Input and Output Capacitances:
For LM Algorithm

$$C_{in} = 10\text{F}$$
$$C_{out} = 20\text{F}$$

For LE algorithm

$$C_{in} = 10\text{ units}$$
$$C_{out} = 20\text{ units}$$

Technology: 180nm BSIM 3v3 model parameter (for LM algorithm), whereas $P = 1$ (for LE algorithm). The paper chooses both sequential and combinational logic gates for optimization and characterization using proposed algorithms (Fig. 2. and Fig. 3.)

<table>
<thead>
<tr>
<th>Circuit/Gate</th>
<th>LE algorithm automation</th>
<th>LM algorithm automation; When o/p is falling</th>
<th>LM algorithm automation; When o/p is rising</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2MOS Flip-Flop</td>
<td>$s_1 = 4; s_2 = 4;$</td>
<td>$S_1 = 1u; S_2 = 1u; S_3 = 1u;$</td>
<td>$S_1 = 3.999u; S_2 = 1u;$</td>
</tr>
<tr>
<td></td>
<td>$s_3 = 2; s_4 = 2;$</td>
<td>$S_4 = 1.579u; S_5 = 1u; S_6 = 1u;$</td>
<td>$S_3 = 4u; S_4 = 1u;$</td>
</tr>
<tr>
<td></td>
<td>$s_5 = 4; s_6 = 4;$</td>
<td>$S_7 = 3.69u; S_8 = 3.132u;$</td>
<td>$S_5 = 2.76u; S_6 = 4u;$</td>
</tr>
<tr>
<td></td>
<td>$s_7 = 2; s_8 = 2;$</td>
<td>$\text{Delay} = 127.79 \text{ps}$</td>
<td>$S_7 = 1u; S_8 = 1u;$</td>
</tr>
<tr>
<td></td>
<td>$\text{Delay} = 66.568542495 \text{units}$</td>
<td>$\text{Delay} = 117.74 \text{ps}$</td>
<td>$\text{Delay} = 117.74 \text{ps}$</td>
</tr>
<tr>
<td>NOR 2X1</td>
<td>$S_1 = 4; S_2 = 4; S_3 = 1; S_4 = 1;$</td>
<td>$S_1 = 4u; S_2 = 2u; S_3 = 4u; S_4 = 4u;$</td>
<td>$S_1 = 8u; S_2 = 8u;$</td>
</tr>
<tr>
<td></td>
<td>$\text{Delay} = 4.6666 \text{units}$</td>
<td>$\text{Delay} = 8.3 \text{ps}$</td>
<td>$S_3 = 2u; S_4 = 2u;$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{Delay} = 105.05 \text{ps}$</td>
<td>$\text{Delay} = 105.05 \text{ps}$</td>
</tr>
<tr>
<td>XOR 2X1</td>
<td>$S_1 = 2; S_2 = 1; S_3 = 4; S_4 = 4;$</td>
<td>$S_1 = 3u; S_2 = 2u; S_3 = 2u;$</td>
<td>$S_1 = 2u; S_2 = 2.5u; S_3 = 2u;$</td>
</tr>
<tr>
<td></td>
<td>$S_5 = 4; S_6 = 4; S_7 = 2; S_8 = 1;$</td>
<td>$S_4 = 2u; S_5 = 2u; S_6 = 2u;$</td>
<td>$S_4 = 2.9u; S_5 = 2u; S_6 = 2u;$</td>
</tr>
<tr>
<td></td>
<td>$S_9 = 9; S_{10} = 2; S_{11} = 2; S_{12} = 2;$</td>
<td>$S_7 = 8u; S_8 = 8u; S_9 = 6u;$</td>
<td>$S_7 = 3.73u; S_8 = 2u; S_9 = 2u;$</td>
</tr>
<tr>
<td></td>
<td>$\text{Delay} = 5 \text{units}$</td>
<td>$S_{10} = 2u; S_{11} = 2u; S_{12} = 2u;$</td>
<td>$S_{10} = 2u; S_{11} = 2u; S_{12} = 2u;$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{Delay} = 93.95 \text{ps}$</td>
<td>$\text{Delay} = 62ps$</td>
</tr>
<tr>
<td>NAND 2X1</td>
<td>$S_1 = 2; S_2 = 2; S_3 = 2; S_4 = 2;$</td>
<td>$S_1 = 2u; S_2 = 2u; S_3 = 4u; S_4 = 4u;$</td>
<td>$S_1 = 4u; S_2 = 4u; S_3 = 2u; S_4 = 2u;$</td>
</tr>
<tr>
<td></td>
<td>$\text{Delay} = 3.6666 \text{units}$</td>
<td>$\text{Delay} = 8.8916 \text{ps}$</td>
<td>$\text{Delay} = 22 \text{ps}$</td>
</tr>
</tbody>
</table>
Fig. 6: CMOS Flip-Flop [13].

Fig. 7: NAND and NOR gates with variable transistor widths.

It can be seen that the results obtained by LE automation algorithm are in agreement with electronics fundamentals i.e. the width of PMOS transistors are more as compared to those of NMOS. Whereas the LM algorithm has random width values i.e. in case of NOR 2X1 transistors S3 and S4 have widths equal to 4, whereas S2 has width equal to 2. Moreover there is difference in transistors widths and delay time when optimized for rising/falling output signals (since during rising output signal the algorithm optimizes PMOS transistor widths while for falling output signals the algorithm optimizes NMOS transistor widths). Hence it requires additional number of iterations to achieve same delay values for both rising and falling output signals.

DISCUSSION AND CONCLUSION

The paper presented an algorithm based on Logical Effort theory for optimization of transistor widths for minimum delay. Step by step procedure of its development and implementation is also reported using examples and scripts. Moreover comparison of developed algorithm with LM algorithm embedded in SPICE is also presented along with examples. Since logical effort theory is based on electronics fundamentals, the obtained results are near to correct values with only +− 5% error. Whereas the results obtained from LM algorithm based automation are correct but not efficient. The transistor widths values obtained with LM algorithm gives desired results but obtained results are not in accordance with electronics fundamentals. Both the LE theory and LM algorithm have their own advantages and limitations. The algorithm based on LE theory is very efficient for working with gate driven logics but it is inefficient in working with pass transistor logic (PTL). Whereas the algorithm based on LM theory optimizes both gate driven and PTL logic but the results are inefficient. The shortcomings of both the algorithms can be removed by introducing limitations in LM algorithm as per electronics principles and by introduction of source/drain capacitance effects in LE theory.

REFERENCES


10. Ivan, E. Sutherland, Bob F. Sproull and David L. Harris, 2004. “Logical Effort: Designing fast CMOS circuits, Morgan Kaufmann Publishers,