Low Power Arithmetic Units using Pass Logic

Damu Radhakrishnan
Division of Computer Engineering, School of Applied Science
Nanyang Technological University, Nanyang Ave., Singapore 639798
e-mail: asdrkrishnan@ntu.edu.sg

Abstract

There is a tremendous need for miniaturization as well as low power consumption in many processor architectures, especially when they are used in portable equipment. In this context, the design of low power DSP processors receives utmost importance. A clever means of achieving this is by use of pass logic circuits in these designs. But switching activity is found to be a major contributor in dynamic power consumption. Hence, a formal technique for calculating the switching activity in CMOS pass networks is presented in this paper, which is validated using various designs of CMOS full adder cells. This technique can therefore be used for the design of CMOS pass networks, with reduced switching activity, suitable for use in arithmetic units for the design of smaller and energy-efficient DSP processors.

1. Introduction

With the fast improvements in technology, the cost of special purpose processors capable of performing DSP tasks has dropped significantly to the point where such a processor can be used in consumer products and other cost-sensitive systems. As a result, more and more products have begun using DSP processors, fueling demand for faster, smaller, cheaper and more energy-efficient chips. These chips open the door for a new wave of products to implement signal-processing capabilities.

One of the most popular arithmetic operation in DSP is multiply-accumulate (MAC). The first major architectural modification that distinguished DSP processors from early general-purpose processors was the addition of specialized hardware that enabled single cycle multiplication. The design of energy-efficient DSP chips must therefore be viewed with ways of reducing the power dissipation of the MAC hardware, especially its dynamic power dissipation. The dynamic power is caused by the switching activity of the circuit. The most significant source of dynamic power dissipation is due to the charging and discharging of internal parasitic capacitances in transistors and interconnect capacitances in integrated circuits [1,3]. The only way to reduce this power is by analyzing and optimizing various circuit topologies, which is by itself quite a complex problem to deal with. When individual logic gates are available which consume minimum power, they have to be interconnected to perform the given logical function in such a way as to obtain the smallest possible number of switchings for particular gates. The number of logic gate switchings is directly related to the consumed power. Therefore, by way of making the logical synthesis of a circuit with activities of particular gates taken into account, one can obtain the circuit with better power and thermal parameters. In this connection, we shall be interested in the total switching activity $A_T$ on the total circuit power and in the individual gate activity $A_i$ on even temperature distribution in the circuit.

Recently, pass logic circuits have been identified as a clever means to reduce power in CMOS designs. Many combinational and sequential logic networks implemented with pass logic have been claimed to reduce the Si area considerably, while reducing the node capacitances to a minimum, thereby providing much higher performance per unit area. Even though many excellent circuits using pass logic have been reported in the literature, their switching activities in relation to power have not been studied so far. Hence in this paper we give an analytical technique for calculating the switching activity in a CMOS pass network. Various designs of pass logic adder cells have been analysed using this technique and are found to exhibit very good correlation between power and switching activity.

2. Pass networks

Detailed analysis and synthesis procedures for pass networks are presented in [4-7]. A pass transistor is an nMOS/pMOS transistor with the signal input fed to the source and the signal output taken from drain. A pass network is an interconnection of a number of pass transistors to achieve a particular switching function. The propagation of the signal through the transistor is controlled by a signal applied to its gate. In the case of an nMOS transistor, a logic ‘1’ at the gate passes the input from source to drain and a logic ‘0’ opens the source to drain path. A pMOS transistor exhibits similar behaviour with a control signal of logic level 0. If signals $X/X$ and $Y$ are connected to the gate and source of an nMOS/pMOS
transistor respectively, then it is denoted as \( X(Y) \) and read as ‘X passing Y’. The variable \( X/X' \) is called a control variable and \( Y \) is called a pass variable. When both an nMOS and a pMOS transistor are connected in parallel to pass the signal \( Y \), the circuit is referred to as a CMOS transmission gate. The logic symbols and the pass logic expressions for the above three types of pass gates are shown in Figure 1 [6]. To avoid any confusion, the same pass logic expression is used to represent both nMOS and pMOS networks, but the control signals applied to the gates of the pMOS transistors are inverted during implementation.

A series connection of a number of nMOS/pMOS transistors passes the input to the output when all control signals are high/low. This is represented by the expression \( \overline{X_1} \overline{X_2} \ldots \overline{X_n} \), where \( X_1, X_2, \ldots, X_n \) are the control variables applied to the gates of the nMOS transistors \( (X_1, X_2, \ldots, X_n) \) are the control variables applied to the gates of the pMOS transistors) and \( V \) is the pass variable.

![Figure 1. Pass transistor symbols for X(Y)](image)

A product term \( P = X_1X_2 \ldots X_n \) passing an input signal \( V \), is defined as the pass implicant and is denoted by \( P(V) \). A pass function is formed by the logical sum of a number of pass implicants and is given by \( f = \sum_{i=1}^{n} P_i(V_i) \). In a minimal pass function, all pass implicants belong to the set of pass prime implicants.

In the pass logic expression, whenever the pass variable in a pass implicant is a 0/1, a purely nMOS/pMOS transistor chain can pass it to the output without any signal degradation. But when a variable is passed (other than 0 or 1) each individual transistor is replaced by a transmission gate, but at the expense of additional inverters for complementing each control variable.

### 3. Switching activity in pass networks

The switching activity for a logic gate refers to the total number of 0 to 1 and 1 to 0 transitions occurring at the output of the gate while all possible two pattern input sequences are applied at the input of the gate. The switching activity for combinational circuits realized using gate logic is presented in [2]. The total switching activity for a complete circuit is equal to the sum of the switching activities at the individual nodes of the circuit. To calculate the switching activity, two sets \( F \) and \( R \) are used. \( F \) is a set with minterms corresponding to the 1’s of the function, and \( R \) is a set corresponding to the 0’s of the function. The number of switchings for any gate is given as a doubled product of the power of these two sets that are referenced to the output of the gate. If we assume that 0 to 1 and 1 to 0 transitions at a node are equivalent, then the total switching activity for a logic circuit is given by:

\[
A_f = \sum_i A_i = \sum_i |F_i||R_i|
\]

where \( A_i \) corresponds to the switching activity at the \( i \)th node, and \(|F_i|\) and \(|R_i|\) corresponds to the cardinality of the sets \( F_i \) and \( R_i \), respectively.

The switching activity for pass networks can be defined in a manner exactly similar to gate networks. For a combinational pass network the total switching activity is equal to the sum of the switching activities at the individual nodes of the circuit. But the calculation of switching activity for pass networks is not as direct as in gate networks. This is due to the inherent nature of pass networks in which the voltage level at an intermediate node may also be dictated by the voltage level at the output node. Thus to find the logic level at each internal node, both the forward path from the input node as well as the feedback path from the output node to the node in question must be considered [8]. But while calculating the switching activity, the type of network considered (nMOS, pMOS or CMOS) is immaterial. Hence the switching activity for a pass network remains the same for all possible choices of transistors used in the network as long as the structural pattern is maintained the same for all the circuits.

Now consider a non-redundant, \( r \) variable pass network as shown in Figure 2 with the pass function given as:

\[
f = \sum_{i=1}^{r} P_i(V_i)
\]

For this network all the internal nodes are formed by exactly two branches (two transistors) connected to each other. But in certain cases, the pass function given above can be factored further to reduce the number of transistors in its implementation. When this happens, some of the internal nodes in the pass network may have more than two transistors connected to each other. These cases are shown in Figures 3(a) and 3(b). In the following the calculation of switching activity at a node \( x \) within the path represented by the pass implicant \( P_i(V_i) \) for the three cases shown in Figures 2, 3(a) and 3(b) are treated differently.

By inspection of the network in Figure 2 let there be \( k \) control variables in forming the product term \( P_i \), and \( m \) control variables involved in connecting the variable \( V_i \) to node \( x \) in the forward path. Obviously, \( m \leq k \leq r \). When \( m = k \), node \( x \) becomes the output node \( f \). Let \( P_{x_{i}} \) denote the product term such that \( V_{x_i} = V_i \) when \( P_{x_{i}} \) = 1, where \( V_{x_i} \) denotes the logic level at node \( x \). Also, let \( P_{x_{sf}} \) denote the product term such that \( f = V_{x_i} \) when \( P_{x_{sf}} \) = 1. Obviously \( P_i = P_{x_{i}} P_{x_{sf}} \). The set \( F \) corresponding to node \( x \) can now be generated using Procedure 1 as follows:
Procedure 1:
1) Initialize $F = \emptyset$ (the null set).
2) If $V_i \neq 0$, include the product term $P_i V_i$ in $F$.
3) Find all product terms $P_j$ for values of $j$ from 1 to $n$, but $j \neq i$ and $V_j \neq 0$ such that $f = 1$ for all such $P_j V_j$ while $P_{xf} = 1$. Include all such product terms $P_j V_j P_{xf}$ in $F$.
4) Expand all implicants in $F$ to their minterm values.

Procedure 2 generates the set $R$ in a similar manner.

Procedure 2:
1) Initialize $R = \emptyset$ (the null set).
2) If $V_i \neq 1$, include the product term $P_i V_i$ in $R$.
3) Find all product terms $P_j$ for values of $j$ from 1 to $n$, but $j \neq i$ and $V_j \neq 1$ such that $f = 1$ for all such $P_j V_j$ when $V_j \neq 0$, or $f = 0$ when $V_j = 0$, while $P_{xf} = 1$. Include all such product terms $P_j V_j P_{xf}$ in $R$.
4) Expand all implicants in $R$ to their minterm values.

In Figure 3(a), the product term forming the network path connecting the input node $V_i$ to $x$ is $P_{ix}$, and those connecting $x$ to output node $f$ are $P_{xf}$, $P_{x2f}$, ..., and $P_{xm,f}$. In this case to generate the set $F$, Step 3 of Procedure 1 has to be modified as follows:
Modified Step 3 (Procedure 1): Find all product terms $P_j$ for values of $j$ from 1 to $n$, but $j \neq i$ and $V_j \neq 0$ such that $f = 1$ for all $P_j V_j$ corresponding to each $P_{xf} = 1$ for $k = 1,2, \ldots, m$. Include all such product terms $P_j V_j P_{xf}$ in $F$.

Similarly, to generate the set $R$, modify Step 3 of Procedure 2 as follows:
Modified Step 3 (Procedure 2): Find all product terms $P_j$ for values of $j$ from 1 to $n$, but $j \neq i$ and $V_j \neq 1$ such that $f = 1$ for all such $P_j V_j$ when $V_j \neq 0$, or $f = 0$ when $V_j = 0$, corresponding to each $P_{xf} = 1$ for $k = 1$ to $m$. Include all such product terms $P_j V_j P_{xf}$ in $R$.

In Figure 3(b), the product terms forming the network paths connecting the input node $V_i$ (a set of $m$ nodes $V_{i1}, V_{i2}, \ldots, V_{im}$) to $x$ are $P_{i1}, P_{i2}, \ldots, P_{im}$ and that connecting $x$ to output node $f$ is $P_{xf}$. In this case to generate the set $F$, Step 2 of Procedure 1 has to be modified as follows:
Modified Step 2 (Procedure 1): For every $V_{ik} \neq 0$, include all product terms $P_{ik} V_{ik}$ in $F$, for all values of $k$ from 1 to $m$. Finally to generate the set $R$, Step 2 of Procedure 2 has to be modified as:
Modified Step 2 (Procedure 2): For every $V_{ik} \neq 1$, include all product terms $P_{ik} V_{ik}$ in $R$, for all values of $k$ from 1 to $m$.

It may be noted at this point that, contrary to the gate networks, the sum of the elements in the two sets $F$ and $R$ may not always add up to $2^m$ for the pass network, where $m$ denotes the number of variables in the switching function.

4. Pass logic full adder cells

A structured approach for designing low power adder cells were presented in [9]. This is done by partitioning the full adder cell into three independent submodules as shown in Figure 4.

In Figure 3(a), the product term forming the network path connecting the input node $V_i$ to $x$ is $P_{ix}$, and those connecting $x$ to output node $f$ are $P_{xf}$, $P_{x2f}$, ..., and $P_{xm,f}$. In this case to generate the set $F$, Step 3 of Procedure 1 has to be modified as follows:
Modified Step 3 (Procedure 1): Find all product terms $P_j$ for values of $j$ from 1 to $n$, but $j \neq i$ and $V_j \neq 0$ such that $f = 1$ for all $P_j V_j$ corresponding to each $P_{xf} = 1$ for $k = 1,2, \ldots, m$. Include all such product terms $P_j V_j P_{xf}$ in $F$.

Similarly, to generate the set $R$, modify Step 3 of Procedure 2 as follows:
Modified Step 3 (Procedure 2): Find all product terms $P_j$ for values of $j$ from 1 to $n$, but $j \neq i$ and $V_j \neq 1$ such that $f = 1$ for all such $P_j V_j$ when $V_j \neq 0$, or $f = 0$ when $V_j = 0$, corresponding to each $P_{xf} = 1$ for $k = 1$ to $m$. Include all such product terms $P_j V_j P_{xf}$ in $R$.

In Figure 3(b), the product terms forming the network paths connecting the input node $V_i$ (a set of $m$ nodes $V_{i1}, V_{i2}, \ldots, V_{im}$) to $x$ are $P_{i1}, P_{i2}, \ldots, P_{im}$ and that connecting $x$ to output node $f$ is $P_{xf}$. In this case to generate the set $F$, Step 2 of Procedure 1 has to be modified as follows:
Modified Step 2 (Procedure 1): For every $V_{ik} \neq 0$, include all product terms $P_{ik} V_{ik}$ in $F$, for all values of $k$ from 1 to $m$. Finally to generate the set $R$, Step 2 of Procedure 2 has to be modified as:
Modified Step 2 (Procedure 2): For every $V_{ik} \neq 1$, include all product terms $P_{ik} V_{ik}$ in $R$, for all values of $k$ from 1 to $m$.

It may be noted at this point that, contrary to the gate networks, the sum of the elements in the two sets $F$ and $R$ may not always add up to $2^m$ for the pass network, where $m$ denotes the number of variables in the switching function.
and $C_{\text{out}} = \overline{A} + HC_{\text{in}}$, where $H = A \oplus B$. Six independent designs for Module 1 (XOR, XNOR combination), four for Module 2 and one for Module 3 were given in [9]. They are reproduced here in Figures 5, 6 and 7 respectively using our transistor symbols given earlier. A total of 25 different choices of adders (including the conventional CMOS adder) by taking all possible combinations of the above were simulated and tabulated in the order of their normalized average power consumption. The switching activities for all of the above designs are calculated and are tabulated in Table 1, together with the power values given in [9]. The different full adder cells in Table 1 are identified by an ordered 2 letter code, with the 1\textsuperscript{st} letter representing the cell taken from module 1 (the XOR-XNOR cell) and the second letter representing the cell taken from module 2 (the XOR cell). As seen from Table 1, a good match is observed between switching activity in these cells and the power consumed. In a few cases the orderings did not match. These can be accounted for due to the reduced voltage swings occurring at their internal nodes. Even though this could reduce the power dissipation at such nodes they do not reflect in the switching activity of those nodes. Also, the change in node capacitance at certain nodes was responsible for modifying the total power dissipation, while not reflecting on the switching activity.

5. Conclusions

Switching activity in CMOS circuits is a good measure of the overall power dissipation, particularly the dynamic power dissipation. The calculation of switching activity is more involved in pass networks as compared to gate networks because of the feedback paths in pass networks. In this paper a formal procedure is given to calculate the switching activity in CMOS pass networks.

Table 1. Switching activity for 1-bit full adder

<table>
<thead>
<tr>
<th>#</th>
<th>Cell</th>
<th>Avg. Power Norm.</th>
<th>Switching Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DB</td>
<td>1.00</td>
<td>160</td>
</tr>
<tr>
<td>2</td>
<td>DA</td>
<td>1.03</td>
<td>176</td>
</tr>
<tr>
<td>3</td>
<td>DD</td>
<td>1.08</td>
<td>176</td>
</tr>
<tr>
<td>4</td>
<td>CA</td>
<td>1.14</td>
<td>160</td>
</tr>
<tr>
<td>5</td>
<td>BA</td>
<td>1.15</td>
<td>176</td>
</tr>
<tr>
<td>6</td>
<td>CB</td>
<td>1.2</td>
<td>144</td>
</tr>
<tr>
<td>7</td>
<td>EA</td>
<td>1.22</td>
<td>176</td>
</tr>
<tr>
<td>8</td>
<td>CD</td>
<td>1.25</td>
<td>160</td>
</tr>
<tr>
<td>9</td>
<td>EB</td>
<td>1.27</td>
<td>160</td>
</tr>
<tr>
<td>10</td>
<td>BB</td>
<td>1.27</td>
<td>160</td>
</tr>
<tr>
<td>11</td>
<td>BD</td>
<td>1.31</td>
<td>176</td>
</tr>
<tr>
<td>12</td>
<td>ED</td>
<td>1.31</td>
<td>176</td>
</tr>
<tr>
<td>13</td>
<td>DC</td>
<td>1.35</td>
<td>192</td>
</tr>
<tr>
<td>14</td>
<td>FA</td>
<td>1.41</td>
<td>208</td>
</tr>
<tr>
<td>15</td>
<td>AA</td>
<td>1.41</td>
<td>208</td>
</tr>
<tr>
<td>16</td>
<td>CC</td>
<td>1.46</td>
<td>176</td>
</tr>
<tr>
<td>17</td>
<td>FB</td>
<td>1.49</td>
<td>192</td>
</tr>
<tr>
<td>18</td>
<td>FD</td>
<td>1.52</td>
<td>208</td>
</tr>
<tr>
<td>19</td>
<td>AD</td>
<td>1.53</td>
<td>208</td>
</tr>
<tr>
<td>20</td>
<td>EC</td>
<td>1.56</td>
<td>192</td>
</tr>
<tr>
<td>21</td>
<td>CMOS</td>
<td>1.56</td>
<td>224</td>
</tr>
<tr>
<td>22</td>
<td>BC</td>
<td>1.57</td>
<td>198</td>
</tr>
<tr>
<td>23</td>
<td>AB</td>
<td>1.58</td>
<td>198</td>
</tr>
<tr>
<td>24</td>
<td>FC</td>
<td>1.71</td>
<td>224</td>
</tr>
<tr>
<td>25</td>
<td>AC</td>
<td>1.79</td>
<td>224</td>
</tr>
</tbody>
</table>
The switching activity calculated for various designs of full adder cells compares well against their power dissipation, thus justifying its use in CAD tools, for the analysis and synthesis of CMOS pass logic circuits suitable for the design of low power arithmetic units.

6. References


