A Low-Power Synthesis of Submicron Interconnects with Time and Area Constraints

A. Mahdoum, R. Benmadache, A. Chenouf, M. L. Berrandjia

Abstract — Technology scaling has resulted in interconnect delay increasing significantly. Buffer-insertion is a well-known technique to reduce wire delays of critical signal nets in a circuit. However, the power consumption of buffers has become a critical concern with the increase of the number of buffers. In this paper, it is shown that this problem is not polynomial in time. Thus, we developed a genetic-based algorithm that provides optimal or near optimal solutions for reducing the power dissipation while meeting the time and area constraints.

Keywords — Submicron interconnections, buffer insertion, low-power design, area and time constraints, genetic-based algorithm.

I. INTRODUCTION

With the advent of new semiconductor technologies, it is possible today to integrate multiple systems on a single chip (SOC). This tight integration offers several advantages, but is certainly not without problems: hybrid systems (digital, analog, mixed RFs) that are present on the same chip require proper and complicated design (e.g. consistent interfacing and communication protocols ...). Compared to older systems, there are other problems due to electro thermal phenomena, coupling ... Among these problems, it is one that is no less important: energy consumption. This problem arises in two ways: i) a strong energy dissipation resulting in an increase in temperature, which could affect the reliability of the system; ii) there exist on the current market many portable systems (PDAs, mobile phones, notebook PCs, etc ...) and for which the operating time of batteries is limited. Obviously, the same problem can arise for the systems on board satellites (the stored energy during the day should be sufficient to operate the system during the night). These are all reasons that lead to a need to low-power circuit designs. Thus, the power dissipation problem is tackled at each level of abstraction either to propose diverse and varied methods estimating this parameter or to design circuits with low power consumption [1]-[24].

In past technologies, gate delay was the major concern. Today, with submicron technologies, this is no longer true. Indeed, wire delay has become a critical concern. Buffer insertion and wire sizing are two interesting techniques to deal with the interested problem. The reader may found many interesting works that addressed this problem ([25]-[29]). In this paper, we show that buffer insertion is not a polynomial in time problem. We then present our genetic-based algorithm that features a twofold purpose: solution search processed in polynomial time while targeting the most interesting (near optimal) solutions. Because power consumption is also a critical problem in modern technologies, our buffer insertion is processed with power (and area) constraints. Our paper is organized as follows. In the next section we present the models we used. In section 3 we give details of our buffer insertion technique. Section 4 presents some obtained results. Finally, we conclude the paper in section 5.

II. MODEL DEFINITIONS

A. Delay Model

Let us consider “Fig.1” in which 1 and 6 are respectively source and sink nodes while 2–5 are candidate positions for buffer insertion. Because a precise delay model for an inverter exists in literature (e.g. [30]), we implement buffers with inverters. Equation (1) shows the delay model for an inverter. \( D_i \) is the delay for the buffer inserted at node \( i \) (2 \( \leq i \leq 5 \)), \( C_{\text{Load}(i)} \) is the capacitance at the output node of the \( i \)th buffer, \( L_i \) and \( W_i \) are the transistor sizes of the NMOS transistor (a similar model as that shown in (1) can be given for the PMOS transistor of the inverter). \( V_{dd(i)} \) and \( V_{th(i)} \) are respectively the supply voltage and the threshold voltage of the NMOS (PMOS) transistor of the \( i \)th inverter. Equation (2) is a delay model of the wire portion between nodes \( i \) and \( j \) ([28]). \( C_{Wij} \) and \( r_{Wij} \) are the capacitance and the resistance of the wire portion between nodes \( i \) and \( j \), respectively. \( l_{Wij} \) is the length of this wire portion.

\[
D_i = \frac{C_{\text{Load}(i)} \times L_i}{\mu C_{\text{Ox}} W_i \left( V_{dd(i)} - V_{th(i)} \right) \times \left[ \frac{2V_{th(i)}}{V_{dd(i)} - V_{th(i)}} + \ln \frac{4(V_{dd(i)} - V_{th(i)})}{V_{dd(i)}} - 1 \right]} 
\]

\[
d_{ij} = \frac{1}{2} \left( r_{Wij} C_{Wij} l_{Wij}^2 \right) + r_{Wij} l_{Wij} C_{Wij} \]

The total delay \( D_{ij} \) between nodes \( i \) and \( j \) (as shown in “Fig.1”) is then:

\[
D_{ij} = D_i + d_{ij}
\]
B. Area Model

The area consumed by the buffers is merely estimated as the sum of the transistor sizes of the inserted inverters.

C. Power Model

The switching power dissipation is given by (4). However, because we target dual Vdd dual Vth circuit designs, we transform it as shown in (5).

\[
P_{sw} = 0.5 \times V_{dd}^2 \times f \sum_{i=1}^{Nb\,gates} C_{Gi} \times N_{Gi} \tag{4}
\]

\[
P_{sw} = 0.5 \times f \left[ V_{dd,L}^2 \sum_{i=1}^{E_i} C_{Gi} \times N_{Gi} + V_{dd,H}^2 \sum_{i=1}^{E_i} C_{Gi} \times N_{Gi} \right] \tag{5}
\]

\(V_{dd}\) and \(f\) are respectively the supply voltage and the frequency. \(C_{Gi}\) is the load capacitance of the \(i\)th logic gate while \(N_{Gi}\) is the number of times \(C_{Gi}\) is charged or discharged under some input sequence. \(V_{dd,L}\) and \(V_{dd,H}\) are the lower and the higher supply voltages, respectively. \(E_i\) (\(E_{Gi}\)) is the set of the logic gates that are fed with \(V_{dd,L}\) (\(V_{dd,H}\)).

The leakage power dissipation is given in BACPAC (Berkeley Advanced Chip Performance) by (6).

\[
P_{leak} = 0.2813 \times V_{dd} \times K \times N_{trans} \times W_{avg} \times L^{\frac{\alpha}{\alpha_V}} \tag{6}
\]

\(W_{avg}\), \(L\), \(N_{trans}\) and \(V_t\) are the average transistor width, the transistor length (in \(\mu m\)), the total number of transistors in the circuit and the threshold voltage, respectively. \(K=10 \ \mu A/\mu m\), \(\alpha_V=0.095 \ V\).

Again, because we are dealing with low-power circuits, we transform it by (7) so that dual \(V_{dd}\) dual \(V_t\) design methodology could be possible.

\[
P_{leak} = 0.2813 \times K \times W_{avg} \times L \times \sum_{i=1}^{Nb\,gates} \left[ N_{bN,i} \times 10^{-\frac{V_{th}}{\alpha_V}} + N_{bP,i} \times 10^{\frac{V_{th}}{\alpha_V}} \right] V_{dd,i} \tag{7}
\]

\(Nb_{N,i}\) (\(Nb_{P,i}\)) is the number of NMOS (PMOS) transistors of the buffers, \(V_{th}(V_{thP})\) is the threshold voltage of the NMOS (PMOS) transistors in the \(i\)th buffer and \(V_{dd,i}\) is the supply voltage of the \(i\)th buffer.

III. BUFFER INSERTION

Let \(N\) be the maximal number of buffers to insert between the source and sink nodes (see “Fig.1”). In order to reduce the wire delay while meeting power and area constraints, an obvious way is to consider all the cases (inserting 1, 2, …, or \(N\) buffers) then to pick the best solution. But to insert only a single buffer, we have \(N\) possibilities: placing it at node 2, 3, …, or \((N+1)\). For inserting \(m\) (\(m \neq 1\)) buffers, the number of possibilities is much larger. The total number of possibilities is \(\sum_{k=1}^{N} C_N^k = \sum_{k=1}^{N} N!/(k!(N-k)!)\), which is a huge number of possibilities. Like many other problems that are intractable [31], this obvious buffer insertion is computationally infeasible, which led us to develop a genetic-based algorithm that features a reasonable CPU time while insuring near optimal solutions.

Before describing our method, notice that our genetic-based algorithm handles a single individual at each generation. This is due to the following reasons:

- starting with the most interesting one, namely with the one that meets the time and area constraints while consuming the lowest power
- in case one or both constraints are not met with the current individual, the next one is generated from it with making few modifications: if the obtained solution will meet the constraints, it will be near optimal since it is generated from the best candidate(s)

Notice also that at each generation, the individual is generated in a deterministic way for the following reasons:

- to keep it not too far from the most interesting solutions (but that did not meet the constraints)
- to guarantee that already explored solutions are not again generated (the CPU time is only consumed to explore new solutions)
- to avoid falling in a cyclic scenario (i.e. the same explored solutions are periodically generated)

Such advantages are explained in details in our book review [32].

For each interconnection in each equipotential, our main algorithm determines, if possible, the buffer positions such that the time and area constraints are met while minimizing the power dissipation.

\texttt{Determine_buffer_positions()} includes three main parts. For each combination (i.e. for some number of buffers and their positions) among \(M\) ones, it generates the ideal individual, namely the one with which the power dissipation is the lowest one. In case the time and area constraints are met, the search process continues with another combination. Else, the procedure tries to find an individual (belonging to the same combination) that meets the constraints with carefully tuning (to keep the solution not too far from the ideal one that did not meet the constraints) the characteristics of the current individual (supply voltage, threshold voltage, size transistors, …): this is the part \(k=2\) in the procedure \texttt{Generate_Individual()}. In case the previous individual met the constraints but it is not the ideal one, \texttt{Generate_Individual()} enhances it in order to reach a lower power dissipation that is possible without violating the constraints: This is the part \(k > 2\).

\texttt{Select_configuration()} returns, if the combinational problem is solvable, three possible solutions:

- \texttt{E_{can}} (the set that includes the positions of the buffers whose electrical parameters are stored in \texttt{E_{buffer}}) and \texttt{E_{buffer}} (the set that includes the solutions that meet the time constraint while consuming both the less power and the less area)
We give hereafter the details of our algorithms with necessary comments:

BEGIN /* main algorithm */
for each equipotential
  {Determine all the interconnections belonging to this equipotential, then sort them in the decreased length; /* in order to first satisfy the constraints for the longest interconnections*/
    for each interconnection
      {Determine G = (V, E); /* V={nodes in the wire, including the source and sink ones},
        E={(vi,vj); vi \in V \forall i \neq j} –see “Fig.1”- */
        Determine_buffer_positions(); /* determine the number of buffers and their positions */
        Select_configuration (); /* in case of many candidate solutions that infer the same wire delay, select the one that best suits the application – power and/or area is the most critical parameter for the interested application - */
      }
  }
END

Determine_buffer_positions()
Smin=\infty; Pmin=\infty; Ecand_1=\emptyset; Ecand_S=\emptyset; Ecand_P=\emptyset;
Ebuffer_1=\emptyset; Ebuffer_S=\emptyset; Ebuffer_P=\emptyset; /* Smin (Pmin) is the minimal area (power) of the buffer configuration that meets the time and area constraints
Ebuffer_1 is the set that includes the solutions that meet the time constraint while consuming both the less power and the less area
Ecand_1 is the set that includes the positions of the buffers whose electrical parameters are stored in Ebuffer_1
Ebuffer_S (Ebuffer_P) is the set that includes the solutions that meet the time constraint while consuming the less power (area)
Ecand_S (Ecand_P) is the set that includes the positions of the buffers whose electrical parameters are stored in Ebuffer_S (Ebuffer_P)*/
for i=1 to M /*M is the number of explored combinations; M \leq \sum_{k=1}^{N} C_{N}^{k} */
  {Generate_Ideal_Individual(); /* Assign W_L, V_{NH}, V_{PL}, V_{DD} for all the buffers in the current combination */
    An ideal individual is a number n of buffers (n \leq N) such that each one is designed with W_L, V_{NH}, V_{PL} and V_{DD}, i.e. the individual that better maximizes the power reduction; subscripts L and H stand to Low and High, respectively*/
    k=1;
  }
LABEL: D=delay(); // calculate the wire delay
S=estimate_area(); // calculate the area of the buffers
if |D - T_f| \leq \epsilon and |S - S_f| \leq \epsilon / T_f and S_f are the time and area constraints, respectively*/
  then {P=Power(); /* calculate the power due to the current buffer insertion */
    if S < S_{min} and P < P_{min}
      then {Ebuffer_i=\{combination i\}; /* combination i stores the electrical parameters W, Vdd, Vth of the m buffers (1 \leq m \leq N) */
        Ecand_i=\{less costly path that is found\}; /* this path includes the source and the sink nodes, and m buffers */
        S_{min}=S; P_{min}=P; }
    else {if S < S_{min}
              then {Ebuffer_S =\{combination i\};
                Ecand_S=\{less costly path that is found\};
              } else {Ebuffer_S=Ebuffer_S \cup \{combination i\};
                Ecand_S=Ecand_S \cup \{less costly path that is found\};
              end if
            }
        if P < P_{min}
          then {Ebuffer_P =\{combination i\};
            Ecand_P=\{less costly path that is found\};
          } else {Ebuffer_P=Ebuffer_P \cup \{combination i\};
            Ecand_P=Ecand_P \cup \{less costly path that is found\};
          end if
        end if
    end if
  }
  if k=1 // ideal case
    then continue; /* stop generating individuals for the current combination, then continue with another one */
  end if
  k++;
  if k \leq nb_individuals
    then {Generate_Individual(D, S, P); goto LABEL;}
  end if
end for

Generate_Individual(D, S, P)
{ if k > 2
    then {i=1;
while $|D - T_f| \leq \varepsilon$ and $i \leq \text{nb\_buffers}$
do if $W_i = W_{H_i}$ /* minimize power and area while meeting the time constraint */
  then {$W_i = W_{L_i}$; calculate $D$; }
  endif
  $i++$;
end
de if $|D - T_f| > \varepsilon$ and $i > 1$
  then {$i--; W_i = W_{H_i}$; }
endif
// Begin process with VthN
$i=1$;
while $|D - T_f| \leq \varepsilon$ and $i \leq \text{nb\_buffers}$ in the current combination
do if $V_{th_i}$ = $V_{th_{NH}}$ /* minimize the leakage current in the NMOS transistors */
  then {$V_{th_i} = V_{th_{NL}}$; calculate $D$; }
  endif
  $i++$;
end
// Begin process with VthN
$i=1$;
while $|D - T_f| \leq \varepsilon$ and $i \leq \text{nb\_buffers}$ in the current combination
do if $V_{dd_i}$ = $V_{dd_{L_i}}$ /* minimizing the supply voltages of the buffers */
  then {$V_{dd_i} = V_{dd_{H_i}}$; $j=j+$; }
  endif
  $i++;$
end
// Begin process with VthN
// Begin process with VthP
Use process with VthN, replacing: VthN with VthP, VthNL with VthPL, VthNH with VthPH
// End process with VthP
// Begin process with Vdd
Use process with VthP, replacing: VthP with Vdd, VthPL with VddH, VthPH with VddL
// End process with Vdd
#else
  // $k=2$: Generate an individual from the ideal one that did not meet the time constraint
  $i=1$;
  while $|D - T_f| > \varepsilon$ and $i \leq \text{nb\_buffers}$ in the current combination
do if $W_i = W_{L_i}$
  then {$W_i = W_{H_i}$; /* Attempting to meet the time constraint with enlarging the sizes of the transistors */
    $S_1 = S$; calculate $S$;
    if $|S - S_f| \leq \varepsilon$
      then calculate $D$;
    else 
      $W_i = W_{L_i}$; $S=S_1$;
    endif
  }
  endif
  $i++;$
end
// Begin process with VthN
$i=1$;
while $|D - T_f| > \varepsilon$ and $i \leq \text{nb\_buffers}$ in the current combination
do if $V_{th_{NH,i}}$ = $V_{th_{NL,i}}$ /* minimizing the leakage current in the NMOS transistors */
  then {$V_{th_{NH,i}} = V_{th_{NL,i}}$; calculate $D$; }
  endif
  $i++;$
end
// Begin process with VthN
// Begin process with VthP
Use process with VthN, replacing: VthN with VthP, VthNL with VthPL, VthNH with VthPH
// End process with VthP
// Begin process with Vdd
Use process with VthP, replacing: VthP with Vdd, VthPL with VddH, VthPH with VddL
// End process with Vdd
#endif
// Select configuration()
if $E_{buffer\_1} = \emptyset$ and $E_{buffer\_P} = \emptyset$ and $E_{buffer\_S} = \emptyset$
  then Write "$No solution for this problem: Too hard constraints$";
  exit();
endif
if $E_{buffer\_1} \neq \emptyset$ /* this set includes solutions that minimize both the power and the area while meeting the time constraint */
  then use $E_{buffer\_1}$ and $E_{cand\_1}$ for buffer insertion;
else if $E_{buffer\_S} \neq \emptyset$ and $E_{buffer\_P} \neq \emptyset$
  then {select 1 combination $\in E_{buffer\_S}$ (resp. $\in E_{buffer\_P}$)
    /* in case the area constraint (resp. the power constraint) has the highest priority for the interested application */
    use ($E_{buffer\_S}$ and $E_{cand\_S}$) or ($E_{buffer\_P}$ et $E_{cand\_P}$) for buffer insertion;
  } else 
    {select 1 combination among those included in the non-empty set ;
    use ($E_{buffer\_S}$ and $E_{cand\_S}$) (resp. ($E_{buffer\_P}$ and $E_{cand\_P}$) for buffer insertion; /* according to $E_{buffer\_S} \neq \emptyset$ (resp. $E_{buffer\_P} \neq \emptyset$) */
    }
  endif
}
IV. RESULTS

Many results were obtained for different wire lengths and time and area constraints targeting the 0.18µm CMOS technology. We present some of them.

TABLE I

<table>
<thead>
<tr>
<th>Without Buffer Insertion</th>
<th>Tool Power (µWatts)</th>
<th>Wire delay (ps)</th>
<th>Area (µm²)</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12.05</td>
<td>2.60</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Heuristic-Based Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>path : 0 2 4 7</td>
<td>0.85</td>
<td>2.49</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 5 7</td>
<td>0.85</td>
<td>2.41</td>
<td>0.237600</td>
<td>4</td>
</tr>
<tr>
<td>path : 0 3 5 7</td>
<td>0.59</td>
<td>2.41</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 4 7</td>
<td>0.85</td>
<td>2.49</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 5 7</td>
<td>0.83</td>
<td>2.52</td>
<td>1.069200</td>
<td>1015</td>
</tr>
<tr>
<td>path : 0 3 5 7</td>
<td>0.59</td>
<td>2.41</td>
<td>0.237600</td>
<td></td>
</tr>
</tbody>
</table>

NA: Not Applicable

Assuming that $V_{ddL}=1.8V$, $V_{ddH}=3.3V$, $V_{thNL}=0.45V$, $V_{thNH}=0.55V$, $V_{thPL}=-0.55V$, $V_{thPH}=-0.45V$, $W_L=0.22µm$ and $W_H=1.76µm$, Table I shows the obtained results for inserting buffers in a wire whose length is equal to 750µm with time and area constraints equal to 2.60ps and 5.7024µm², respectively. The heuristic-based method was able to output the exact solution (inserting 2 buffers at nodes 3 and 5. Note that 0 and 7 are source and sink nodes, respectively). The total power, wire delay and area are obtained with the following parameters:

- Buffer3: $V_{dd}=3.30V$, $V_{thN}=0.55V$, $V_{thP}=-0.55V$, $W_N=0.22µm$, $W_P=0.44µm$
- Buffer5: $V_{dd}=3.30V$, $V_{thN}=0.55V$, $V_{thP}=-0.55V$, $W_N=0.22µm$, $W_P=0.44µm$

$V_{dd}$ is the supply voltage feeding the inverter, $V_{thN}$ and $V_{thP}$ are respectively the threshold voltages of the NMOS and PMOS transistors of the inverter. $W_N$ and $W_P$ are respectively the widths of the NMOS and PMOS transistors of the buffer. Due to an exhaustive search, the CPU time consumed by the exact method was much larger than that of the heuristic-based method (4 s VS 1015 s). Note that this buffer insertion leads to 95% (100 - 95/1.05) reduction in power dissipation against wire design without buffer insertion (0.59 µW VS 12.05 µW) while meeting the time and area constraints.

Table II shows the obtained results for inserting buffers in a wire whose length is equal to 900µm with time and area constraints equal to 3.73 ps and 7.60 µm², respectively. Again, our heuristic-based method was able to output the exact solution in a shorter CPU time (23 s) with respect to the exact method (21529 s). The best solution was achieved with inserting 2 buffers at nodes 6 and 8 (the results in Table I - that are obtained for another wire length and other constraints - show that the buffer insertion concerns nodes 3 and 5 instead of nodes 6 and 8) and assigning the following values for the different parameters:

- Buffer6: $V_{dd}=3.30V$, $V_{thN}=0.55V$, $V_{thP}=-0.55V$, $W_N=0.22µm$, $W_P=0.44µm$
- Buffer8: $V_{dd}=3.30V$, $V_{thN}=0.55V$, $V_{thP}=-0.55V$, $W_N=0.22µm$, $W_P=0.44µm$

<table>
<thead>
<tr>
<th>Without Buffer Insertion</th>
<th>Total Power (µWatts)</th>
<th>Wire delay (ps)</th>
<th>Area (µm²)</th>
<th>CPU Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>14.46</td>
<td>3.73</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>path : 0 1 3 5</td>
<td>1.19</td>
<td>3.59</td>
<td>0.475200</td>
<td></td>
</tr>
<tr>
<td>path : 0 1 3 9</td>
<td>1.23</td>
<td>3.33</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 1 4 9</td>
<td>1.13</td>
<td>3.17</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 1 6 9</td>
<td>1.23</td>
<td>3.18</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 1 7 9</td>
<td>1.23</td>
<td>3.38</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 3 5</td>
<td>1.04</td>
<td>3.59</td>
<td>0.475200</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 3 9</td>
<td>1.09</td>
<td>3.67</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 4 5</td>
<td>1.04</td>
<td>3.59</td>
<td>0.475200</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 4 6</td>
<td>1.04</td>
<td>3.52</td>
<td>0.475200</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 4 8</td>
<td>1.14</td>
<td>3.24</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 4 9</td>
<td>0.98</td>
<td>2.99</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 5 9</td>
<td>0.98</td>
<td>2.91</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 6 9</td>
<td>0.98</td>
<td>2.82</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 2 7 9</td>
<td>0.89</td>
<td>2.84</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 3 4 9</td>
<td>1.00</td>
<td>3.03</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 3 6 9</td>
<td>0.75</td>
<td>3.17</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 3 7 9</td>
<td>0.75</td>
<td>2.91</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 3 8 9</td>
<td>0.75</td>
<td>2.84</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 4 5 9</td>
<td>0.75</td>
<td>2.94</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 4 6 9</td>
<td>0.60</td>
<td>3.18</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 4 7 9</td>
<td>0.60</td>
<td>3.02</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 5 6 9</td>
<td>0.60</td>
<td>3.02</td>
<td>0.237600</td>
<td></td>
</tr>
<tr>
<td>path : 0 5 7 9</td>
<td>0.60</td>
<td>3.02</td>
<td>0.237600</td>
<td></td>
</tr>
</tbody>
</table>
Finally, note that this buffer insertion leads to 96% (100 - 51/14.46) reduction in power dissipation against wire design without buffer insertion (0.51 µW VS 14.46 µW) while meeting the time and area constraints. Our CAD tool was developed with C++ language and qt tool under Linux operating system. Our GUI shows that our tool is user-friendly and enables the user to perform different tasks: “Fig.2” shows the window that allows him to do some common tasks (editing a file, saving it, …) while both “Fig.3” and “Fig.4” serve to capture the parameters of the technology process and the time and area constraints. The windows in “Fig.3” and “Fig.4” serve to perform the exact method and the heuristic-based one, respectively.

Fig. 2 The main window of our CAD tool (editing a file, saving it, …)

Fig. 3 Editing the constraints then launching the exact method
V. CONCLUSION

In this paper, we have presented our genetic-based technique for low-buffer insertion in order to reduce the power dissipation in submicron wires while meeting the time and area constraints. The obtained results show that our method is a potential and a promising way to deal in a reasonable CPU time with wires of circuits designed for modern technologies.

REFERENCES


