An Efficient Low-Power Buffer Insertion with Time and Area Constraints

Ali Mahdoum

Amel Chenouf
Advanced Technologies Development Center, CDTA

Mohamed Lamine Berrandjia
Centre Algérien de Développement des Technologies Avancées

An Efficient Low-Power Buffer Insertion with Time and Area Constraints

A. MAHDOUM, R. BENMADACHE, A. CHENOUF, M. L. BERRANDJIA
Division of Microelectronics and Nanotechnologies
Centre de Développement des Technologies Avancées
BP 17 Baba Hassan 16303 Algiers
ALGERIA
amahdoum@cdta.dz http://www.cdta.dz

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. Thanks to a genetic-based algorithm, our work addresses the interconnect delay problem while meeting power and area constraints.

Key-Words: - Submicron interconnections, buffer insertion, low-power design, area and time constraints

1 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] - [23].

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 ([24]-[28]). 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.

2 Model Definitions

2.1 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. [29]), 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 ($2 \le i \le 5$), $C_{Load(i)}$ 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 (similar model as that shown in Equation 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 of the i^{th} inverter. Equation(2) is a delay model of the wire portion between nodes i and j ([27]). C_{Wij} and r_{Wij} are the

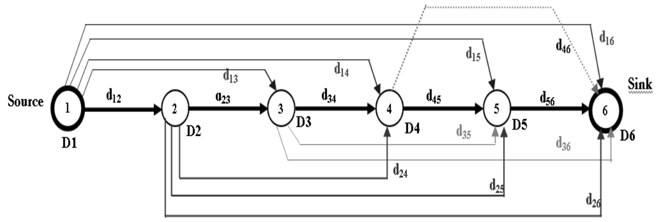

Figure 1. Delays involved by candidate positions for buffer insertion.

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

capacitance and the resistance of the wire portion between nodes i and j, respectively. l_{Wij} is the length of this wire portion.

$$d_{ij} = \frac{1}{2}\left(r_{Wij} C_{Wij}\ l_{Wij}^2\right) + r_{Wij}\ l_{Wij} C_{bj} \quad (2)$$

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

$$D_{ij} = D_i + d_{ij} \quad (3)$$

2.2 Area Model

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

2.3 Power Model

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

$$P_{sw} = 0.5 \times V_{dd}^2 \times f \sum_{i=1}^{Nb\_gates} C_{Gi} \times N_{Gi} \quad (4)$$

$$P_{sw} = 0.5 \times f \left[V_{dd,L}^2 \sum_{i=1}^{|E_L|} C_{Gi} N_{Gi} + V_{dd,H}^2 \sum_{i=1}^{|E_H|} C_{Gi} N_{Gi} \right] \quad (5)$$

V_{dd} and f are respectively the supply voltage and the frequency. C_{Gi} is the load capacitance of the ith 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_L (E_H) 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 Equation(6).

$$P_{leak} = 0.2813 \times V_{dd} \times K \times N_{trans} \times W_{avg} \times L \times {}^{-V_t}\!/_{\alpha_V} \quad (6)$$

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

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

$$P_{leak} = 0.2813 \times K \times W_{avg} \times L \times \sum_{i=1}^{Nb\_gates} \left[Nb_{N,i} \times 10^{-V_{tN,i}/\alpha_V} + Nb_{P,i} \times 10^{V_{tP,i}/\alpha_V} \right] V_{dd,i} \quad (7)$$

$Nb_{N,i}$ ($Nb_{P,i}$) is the number of NMOS (PMOS) transistors of the buffers, $V_{tN,i}$ ($V_{tP,i}$) is the threshold voltage of the NMOS (PMOS) transistors in the ith buffer and $V_{dd,i}$ is the supply voltage of the ith buffer.

3 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 ≠ 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 [30], 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 [31].

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.

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 *Generate_Individual()*. In case the previous individual met the constraints but it is not the ideal one, *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).

Select_configuration() returns, if the combinational problem is solvable, three possible solutions:

- E_{cand_1} (the set that includes the positions of the buffers whose electrical parameters are stored in E_{buffer_1}) and E_{buffer_1} (the set that includes the solutions that meet the time constraint while consuming both the less power and the less area)
- E_{cand_S} (the set that includes the positions of the buffers whose electrical parameters are stored in E_{buffer_S}) and E_{buffer_S} (the set that includes the solutions that meet the time constraint while consuming the less area)
- E_{cand_P} (the set that includes the positions of the buffers whose electrical parameters are stored in E_{buffer_P}) and E_{buffer_P} (the set that includes the solutions that meet the time constraint while consuming the less power)

We give hereafter the details of our algorithms with necessary comments:

BEGIN /* main algorithm */
<u>for each</u> equipotential
<u>do</u> {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 */
<u>for each</u> interconnection
<u>do</u> {Determine G = (V, E); /* V={nodes in the wire, including the source and sink ones}, $E=\{(v_i,v_j);\ v_i \in V\ \forall\ i \neq j\}$ –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 - */ }
<u>end</u> }
<u>end</u>
END

<u>Determine_buffer_positions()</u>
S_{min}=+∞; P_{min}=+∞; E_{cand_1}=∅; E_{cand_S}=∅; E_{cand_P}=∅;
E_{buffer_1}= ∅; E_{buffer_S}= ∅; E_{buffer_P}= ∅;
/* S_{min} (P_{min}) is the minimal area (power) of the buffer configuration that meets the time and area constraints
E_{buffer_1} is the set that includes the solutions that meet the time constraint while consuming both the less power and the less area
E_{cand_1} is the set that includes the positions of the buffers whose electrical parameters are stored in E_{buffer_1}
E_{buffer_S} (E_{buffer_P}) is the set that includes the solutions that meet the time constraint while consuming the less area (power) but not the less power (area)
E_{cand_S} (E_{cand_P}) is the set that includes the positions of the buffers whose electrical parameters are stored in E_{buffer_S} (E_{buffer_P}) */
<u>for</u> i=1 <u>to</u> M /*M is the number of explored combinations; $M \leq \sum_{k=1}^{N} C_N^k$ */
<u>do</u> {Generate_Ideal_Individual(); /* Assign W_L, V_{thNH}, V_{thPL}, V_{ddL} 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_{thNH}, V_{thPL} and V_{ddL}, 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
<u>if</u> $|D - T_f| \leq \varepsilon$ <u>and</u> $|S - S_f| \leq \varepsilon$ /* T_f and S_f are the time and area constraints, respectively */
<u>then</u> {P=Power(); /* calculate the power due to the current buffer insertion */
<u>if</u> $S < S_{min}$ <u>and</u> $P < P_{min}$
<u>then</u> {E_{buffer_1}= {combination i};

```
                /* combination i stores the electrical
parameters W, Vdd, Vth of the m buffers (1 ≤ m ≤ N) */
                E_cand_1={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 if P = P_min
                        then {E_buffer_S =
                                 {combination i};
                              E_cand_S={less costly path
                                 that is found }; }
                        else {E_buffer_S=E_buffer_S ∪
                                 {combination i};
                              E_cand_S=E_cand_S ∪
                   { less costly path that is found}; }
                        end if
                   end if
                   if P < P_min
                   then   if S=S_min
                          then {E_buffer_P=
                                 {combination i};
                                E_cand_P={less costly
                                 path that is found}; }
                          else {E_buffer_P=E_buffer_P ∪
                                 {combination i};
                                E_cand_P=E_cand_P ∪ {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   }
      end if
      k++;
      if k ≤ nb_individuals
      then {Generate_Individual(D, S, P);
            goto LABEL;}
      endif     }
end

Generate_Individual(D, S, P)
{ if k > 2
  then {i=1;
        while |D - Tf| ≤ ε and i ≤ nb_buffers
        do {if Wi=W_H   /* minimize power and area
                 while  meeting the time constraint */
            then {Wi=W_L;    calculate D;  }
            end if
            i++; }
        end
        if |D - Tf| > ε and i > 1
        then {i--; Wi=W_H; }
        end if
       // Begin process with VthN
       i=1;
        while |D – Tf| ≤ ε and i ≤ nb_buffers in the
                                  current combination
        do {if VthN,i = Vth_NL      /* minimize the
               leakage current in the NMOS transistors */
             then {VthN,i=Vth_NH;   calculate D; }
             end if
             i++; }
        end
        if |D – Tf| > ε and i > 1
        then {i--; VthN,i=Vth_NL; }
        end if
        // End process with VthN
        //  Begin process with VthP
           Use process with VthN, replacing: VthN with
              VthP, Vth_NL with Vth_PH, Vth_NH with Vth_PL
        // End process with VthP
        //  Begin process with Vdd
           Use process with VthN, replacing: VthN with
              Vdd, Vth_NL with Vdd_H, Vth_NH with Vdd_L
        // 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-Tf| > ε and i ≤ nb_buffers in the
                                 current combination
        do {if Wi=W_L
             then {Wi=W_H;  /* Attempting to meet the
                         time constraint with enlarging
                         the sizes of the transistors */
                   S1=S;      calculate S;
                   if |S – Sf| ≤ ε
                   then calculate D;
                   else { Wi=W_L;  S=S1; }
                   endif
                  }
             endif
             i++; }
        end
        // Begin process with VthN
        i=1;
       while |D – Tf| > ε and i ≤ nb_buffers in the
                                 current combination
       do {if VthN,i=Vth_NH
            then {VthN,i=Vth_NL;    /* Attempting to
meet the time constraint with reducing the threshold
voltage of NMOS transistors  */
                   calculate D;  }
```

```
            endif
             i++;   }
        end
      // End process with VthN
      // Begin process with VthP
        Use the last process with VthN, replacing:
VthN with VthP, Vth_NL with Vth_PH, Vth_NH with Vth_PL
      // End process with VthP
      while |D – Tf| > ε and i ≤ nb_buffers in the
                                current combination
      do {if Vdd,i=Vdd_L
          then {Vdd,i=Vdd_H;
                for j=i+1 to  nb_buffers in the
                                current combination
                do {Vdd,j=Vdd_H;  /* Attempting
to meet the time constraint with increasing the supply
voltages of the buffers */
                          j++;    }
                   end
                   i=j; }
             else i++;
             endif }
        end  }
endif

Select_configuration()
{
 if Ebuffer_1 = ∅ and Ebuffer_P = ∅ and
                                         Ebuffer_S = ∅
 then {Write "No solution for this problem: Too hard
                constraints";    exit();}
 endif
 if Ebuffer_1 ≠ ∅ /* this set includes solutions that
minimize both the power and the area while meeting the
time constraint */
 then use Ebuffer_1 and Ecand_1 for buffer insertion;
 else if Ebuffer_S ≠ ∅ and Ebuffer_P ≠ ∅
      then {select 1 combination ∈ Ebuffer_S (resp.
              ∈ Ebuffer_P)
               /* in case the area constraint (resp. the
     power constraint) has the highest priority for the
     interested application */
            use (Ebuffer_S and Ecand_S) or
                  (Ebuffer_P et Ecand_P) for buffer
                                         insertion;   }
      else {select 1 combination among those
                         included in the non-empty set ;
            use (Ebuffer_S and Ecand_S)
               (resp. (Ebuffer_P and Ecand_P)) for
                                        buffer insertion;
/* according to Ebuffer_S ≠ ∅ (resp. Ebuffer_P ≠ ∅) */
            }
      endif
  endif  }
```

4 Results

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

		Toal Power (µWatts)	Wire delay (ps)	Area (µm²)	CPU Time (s)
Without Buffer Insertion		12.05	2.60	NA	NA
Heuristic-Based Method	path : 0 2 4 7	0.85	2.49	0.237600	4
	path : 0 2 5 7	0.85	2.41	0.237600	
	path : 0 3 5 7	0.59	2.41	0.237600	
Exact Method	path : 0 2 4 7	0.85	2.49	0.237600	1015
	path : 0 2 5 7	0.83	2.52	1.069200	
	path : 0 3 5 7	0.59	2.41	0.237600	

NA: Not Applicable

Table1. Obtained Results with Wire Length=750 µm , Tf=2.60 ps and Sf=5.7 µm²

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 1 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 5.7024µm^2, 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=0.44 µm
- 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 - 59/12.05) 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 2 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 7.6 µm^2, 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 1 - 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=0.44 µm
- Buffer8: V_{dd}=3.30V,Vth_N=0.55V, Vth_P=-0.55V, W_N=0.22µm, W_P=0.44µm

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.

		Toal Power (µWatts)	Wire delay (ps)	Area (µm²)	CPU Time (s)
Without Buffer Insertion		14.46	3.73	NA	NA
Heuristic-Based Method	path : 0 1 3 5 7 9	1.19	3.59	0.475200	23
	path : 0 1 3 9	1.23	3.67	0.237600	
	path : 0 1 4 9	1.23	3.33	0.237600	
	path : 0 1 5 9	1.13	3.17	0.237600	
	path : 0 1 6 9	1.23	3.18	0.237600	
	path : 0 1 7 9	1.23	3.38	0.237600	
	path : 0 2 3 5 7 9	1.04	3.59	0.475200	
	path : 0 2 3 9	1.09	3.67	0.237600	
	path : 0 2 4 5 7 9	1.04	3.59	0.475200	
	path : 0 2 4 6 7 9	1.04	3.59	0.475200	
	path : 0 2 4 6 8 9	1.04	3.52	0.475200	
	path : 0 2 4 9	1.14	3.24	0.237600	
	path : 0 2 5 9	0.98	2.99	0.237600	
	path : 0 2 6 9	0.98	2.91	0.237600	
	path : 0 2 7 9	1.14	3.02	0.237600	
	path : 0 2 8 9	1.09	3.30	0.237600	
	path : 0 3 4 9	1.00	3.33	0.237600	
	path : 0 3 5 9	0.89	2.99	0.237600	
	path : 0 3 6 9	0.84	2.82	0.237600	
	path : 0 3 7 9	0.89	2.84	0.237600	
	path : 0 3 8 9	1.00	3.03	0.237600	
	path : 0 4 5 9	0.75	3.17	0.237600	
	path : 0 4 6 9	0.75	2.91	0.237600	
	path : 0 4 7 9	0.75	2.84	0.237600	
	path : 0 4 8 9	0.75	2.94	0.237600	
	path : 0 5 6 9	0.60	3.18	0.237600	
	path : 0 5 7 9	0.60	3.02	0.237600	
	path : 0 5 8 9	0.60	3.03	0.237600	
	path : 0 6 7 9	0.51	3.38	0.237600	
	path : 0 6 8 9	0.51	3.30	0.237600	
Exact Method	path : 0 1 3 5 7 9	1.19	3.59	0.475200	21529
	…	…	…	…	
	path : 0 6 8 9	0.51	3.30	0.237600	

NA: Not Applicable

Table 2. Obtained Results with Wire Length =900 µm, Tf=3.73 ps and Sf=7.6 µm²

5 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:

[1] A. P. Chandrakasan, M.Potkonjak, R.Mehra, J. Rabaey, and R.W. Brodersen, Optimizing power using transformations, *IEEE Trans. CAD of ICS*, Vol. 14, No. 1, pp. 12-31.

[2] B.S. Haroun, and M.I. Elmasry, Architectural synthesis for DSP silicon compilers, *IEEE Trans. CAD for ICS*, Vol. 8, No. 4, pp. 431-447.

[3] A. Mahdoum, Synthèse de systèmes monopuce à faible consommation d'énergie, Faible Tension Faible Consommation FTFC'09, 3-5 Juin 2009, Centre Suisse d'Electronique et de Microtechnique, Neuchâtel, Suisse.

[4] Y-H. Lu, L. Benini, and G. De Micheli, Low- power task scheduling for multiple devices, in *Proc. Intl. Workshop on Hardware/Software Codesign*, 2000, pp. 39-43.

[5] P. Rong, and M. Pedram, Power-aware scheduling and DVS for tasks running on a hard real-time system, in *Proc. ASPDAC*, 2006, pp.463-478.

[6] A. Kumar, L. Shang, L.-S. Peh, and N. K. Jha, HybDTM: A coordinated HW–SW approach for dynamic thermal management, in *Proc. DAC*, 2006, pp. 548–553.

[7] G. Paci, P. Marchal, F. Poletti, and L. Benini, Exploring temperature-aware design in low-power MPSoC, in *Proc. Design And Test in Europe*, 2006, pp. 838–843.

[8] B. Tabbara, A. Tabbara, and A. Sangiovanni-Vincentelli, *Function/architecture optimization and Co-Design of Embedded Systems*, Kluwer Academic Publishers, 2000.

[9] M. Lajolo, A. Raghunathan, S. Dey, L. Lavagno, and A. Sangiovanni-Vincentelli, Efficient power estimation techniques for HW/SW systems, in *Proc. IEEE VOLTA'99 International Workshop on Low Power Design*, Como, 1999, pp. 191-199.

[10] L. Yuan, G. Qu, T. Villa, and A. Sangiovanni-Vincentelli, FSM Re-engineering and its application in low power state encoding, in *Proc. ASP-DAC*, Shanghai, 2005.

[11] E. Hwang, F. Vahid, and Y-C. Hsu, FSMD Functional partitioning for low power, in *Proc. Design And test in Europe*, 1999, pp. 22-28.

[12] E. Hwang, F. Vahid, and Y-C. Hsu, Procedural functional partitioning for low power, in *Proc. Intl. Symp. on Low-Power design*, 2000, pp. 65-69.

[13] M. Takahashi, N. Ishuara, A. Yamada, and T. Kambe, Thread composition method for hardware compiler Bach maximizing resource sharing among processes, *IEICE Trans. Fundamentals*, Vol.E83-A, No.12, pp. 2456-2463.

[14] Y. Fei, and N. K. Jha, Functional partitioning for low power distributed systems of systems-on-a-chip, in *Proc. 15th Intl Conf. on VLSI Design*, 2002, pp. 274-281.

[15] L. A. Cortes, P. Eles, and Z. Peng, Quasi-static assignment of voltages and optional cycles for maximizing rewards in real-time systems with energy constraints, in Proc. Design Automation Conference, 2005, pp. 889-894.

[16] A. Mahdoum, N. Badache, and H. Bessalah, An efficient assignment of voltages and optional cycles for maximizing rewards in real-time systems with energy constraints, *Journal Of Low Power Electronics*, Vol.2, No.2, pp. 189-200.

[17] A. Mahdoum, SPOT: A tool for estimating the maximal switching power dissipation of CMOS circuits, accepted in *SASIMI'97*, 1-2 Dec. 97, Osaka, Japan.

[18] A. Mahdoum, SPOT: A tool for estimating the average and the maximal switching power dissipation of CMOS circuits, in *Designer's Forum Proc. Design And Test in Europe*, 2002, p 260.

[19] T. Mahnke et all, Power optimization through dual supply voltage scaling using power compiler, *European Synopsys Users Group Meeting*, Paris, 2002, pp. 87-92.

[20] Q. Wang, and S.B.K. Vrudhula, Static power optimization of deep submicron CMOS circuits for dual V_t technology, in *Proc. IEEE ICCAD*, 1998, pp. 490-496.

[21] K. Usami et all, Automated Low-power technique exploiting multiple supply voltages applied to a media processor, *IEEE Trans. J. Solid State Circuits*, Vol. 33, No. 3, pp. 463-472.

[22] A. Mahdoum, M. L. Berrandjia, FREEZER2: Un outil à base d'un algorithme génétique pour une aide à la conception de circuits digitaux à faible consommation de puissance, *IEEE/FTFC'07*, 21-23 Mai 2007, Paris, France, pp. 143-148.

[23] D. Markovic, C. Wang, L. Alarcon, T.T. Liu, and J. Rabaey, Ultra low-power design in near-threshold_region, in *Proc. IEEE*, 98, pp. 237-252, 2010.

[24] S. Turgis, N. Azemard, and D. Auvergne, Design and selection of buffers for minimum power-delay product, *ED&TC'96*, pp. 224–227.

[25] Y. Gao, and D. F. Wong, A graph based algorithm for optimal buffer insertion under accurate delay models, *DATE'01*, pp. 535–539.

[26] C. Alpert, C. Chu, G. Gandham, M. Hrkic, J. Hu, C. Kashyap, and S. Quay, Simultaneous driver sizing and buffer insertion using a delay penalty estimation technique, *IEEE Transactions on computer-aided design of integrated circuits and systems*, Vol.23, No.1, pp. 136–141.

[27] R. R. Rao, D. Blaauw, D. Sylvester, C. J. Alpert, and S. Nassif, An efficient surface-based low-power buffer insertion algorithm, *ISPD*, pp. 86–93.

[28] H. Ghasemzadeh, N. Jain, M. Sgroi, and R. Jafari, Communication minimization for in-network processing in body sensor networks: a buffer assignment technique, *DATE'09*, pp. 358-363.

[29] N. H. E. Weste, and K. Eshraghian, *Principles of CMOS design: a systems perspective*, Addison Wesley, 1993.

[30] M.R. Garey, and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, San Francisco, CA, Freeman, 1979.

[31] A. Mahdoum, Review of the book *Representations for genetic and evolutionary algorithms* written by F. Rothlauf, edited by Springer-Verlag Editions, The Computer Journal, Oxford journals, Vol.49, No.5, p 629, 2006.