Sensitisable-path-oriented clustered voltage scaling technique for low power

J.-Y. Jou
D.-S. Chou

Indexing terms: Clustered voltage scaling technique, Benchmark circuits

Abstract: Because the average power consumption of CMOS digital circuits is proportional to the square of the supplied voltage, a clustered voltage scaling (CVS) technique has previously been proposed to reduce power without sacrificing the circuit performance. In this paper the authors propose a path-oriented CVS algorithm, which can take the false paths into account. Extensive experiments are conducted on ISCAS85 benchmark circuits. These experiments show that many more gates can be voltage scaled down in comparison with the original CVS technique. An additional 22% power reduction ratio over that of the original CVS technique is achieved.

1 Introduction

In recent years power consumption has become an important issue in VLSI design in addition to speed and area considerations. There are many driving forces behind this. The most apparent one is that a lot of battery-operated applications such as cellular phones, laptop computers, notebook computers and other portable electronic equipment are more and more popular. Making the circuits in portable applications consume less power is just a way of achieving portability with longer life.

Circuit reliability is also an important driving force for low power VLSI design. While working in a high operating temperature, the probabilities of many failure mechanisms such as thermal runaway, junction fatigue, electromigration, will grow rapidly. Every 10°C increase in operating temperature is likely to double a device's failure rate. Therefore, low power VLSI design methodologies are necessary for VLSI designs with higher reliability.

In this paper, a power optimisation technique known as the voltage scaling technique is considered. Power consumption of a CMOS circuit is proportional to the square of the supplied voltage, therefore, the voltage scaling could attain a greater power reduction ratio with respect to other power optimisation techniques. However, the voltage scaling technique has some problems that need to be overcome. First, mixed voltage sources will cause difficulties in circuit layout, and make routing overheads grow rapidly. Second, level converters need to be inserted between low voltage cells and high voltage cells. Because level converters also consume power, reducing the number of level converters becomes an important issue. Finally, it is a critical problem to determine which cells in a circuit are selected to be replaced with low voltage cells.

A clustered voltage scaling (CVS) technique has been proposed by Usami and Horowitz [1]. This technique divides a circuit into two blocks with different voltage supplies, and appends a layer of level converters to primary outputs. The circuit structure created by the CVS technique is called the CVS structure. This structure reduces the overheads of layout and the number of level converters needed. It calculates the slack of each cell and uses it to determine if this cell could be replaced with a corresponding low voltage cell. If the slacks of all other cells are positive after replacing the processed gate with its corresponding low voltage cell, the processed gate is then replaced with a low voltage cell. if one or more cells have negative slacks, the replacement is not allowed, and the high voltage form of the processed gate must be used.

In CVS, the circuit delay is estimated to the delay of the longest path. However the delay of a circuit is determined by its longest sensitisable path, which may not be the longest path. The proposed path-oriented clustered voltage scaling (PCVS) technique adopts the idea of false paths and sensitisable paths. This means that the paths being optimised are long sensitisable paths, instead of long paths. With this improvement, the result of optimisation will be much better than the original clustered voltage scaling technique.

2 Clustered voltage scaling technique

2.1 Definitions and terminology

The *arrival time* of the signal at node j, denoted by $A(j)$, is the latest time when the signal at node j may be stable. Let $D(i, j)$ be the delay between node i and node j. If the arrival time at each fan-in node of node j, and the delay between each fan-in node i and the node j, are known, the arrival time at node j can be computed by the following equation:

$$A(j) = \max_{i \in FI(j)} [A(i) + D(i, j)] \quad (1)$$

The *required time* of the signal at node j, denoted by

IEE Proceedings online no. 19982018

Paper received 5th November 1997

The authors are with the Department of Electronics Engineering, National Chiao Tung University, 1001 Ta Hsueh Road, Hsinchu 300, Taiwan, Republic of China

$R(j)$, is the earliest time when the signal at node j is required to be stable. If the required time at each fan-out node of node j, and the delay between each fan-out node k and the node j, are known, the required time at node j can be computed by the following equation:

$$R(j) = \min_{k \in FO(j)} [R(k) - D(j,k)] \quad (2)$$

The *slack* of the signal at node j, denoted by $S(j)$, is just the difference between the required time and the arrival time. Hence, the slack at node j can be computed by the equation below:

$$S(j) = R(j) - A(j) \quad (3)$$

2.2 Clustered voltage scaling structure

The basic idea of the clustered voltage scaling technique is to apply two different supply voltages, V_{DDL} (the reduced voltage) and V_{DDH} (the original voltage), to a circuit. Cells in the circuit with excessive slack are made to operate at V_{DDL}, while those along the long paths are made to operate at V_{DDH}. However, there are some problems to be solved when applying two different voltages to a circuit.

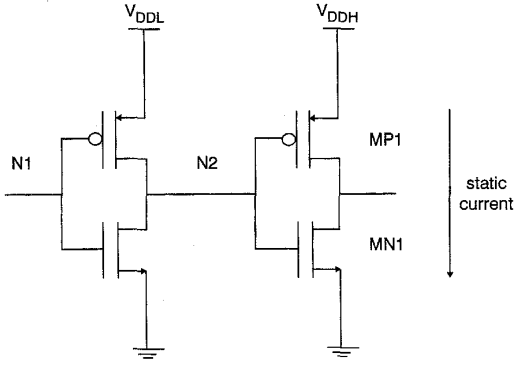

Fig. 1 *Direct connection of V_{DDL} circuit and the V_{DDH} circuit*

As shown in Fig. 1, the output of the V_{DDL} circuit is directly connected to the input of the V_{DDH} circuit. If the input node $N1$ of the V_{DDL} circuit goes to low, the output node $N2$ then goes to high, that is V_{DDL}. The logical high at node $N2$ should subsequently turn off the pull-up network $MP1$ and turn on the pull-down network $MN1$. Although the voltage at node $N2$ is high enough to activate the NMOS $MN1$, it cannot cut off the PMOS $MP1$ due to the fact that $V_{DDL} < V_{DDH} - |V_{th,p}|$. Therefore, there exists a static current flowing directly from the applied voltage source to ground through the path of $MP1$ and $MN1$. This static current also consumes power so is not desirable for low power design. To avoid this unnecessary power dissipation, level converters should be placed between V_{DDL} and V_{DDH} circuits. A level converter transforms a logical high produced by a V_{DDL} circuit to the logical high for a V_{DDH} circuit. Thus the condition that both networks $MP1$ and $MN1$ are activated at the same time as described above will not occur, and the power dissipated by this static current is eliminated. It should be noted that no level converter is needed in the reversed case (i.e. when the output of a V_{DDH} circuit is connected to the input of a V_{DDL} circuit).

Unfortunately, level converters consume power too. Hence a structure such as 'V_{DDL} circuit – V_{DDH} circuit – V_{DDL} circuit – V_{DDH} circuit ...' will need a lot of level converters to be inserted at each 'V_{DDL} circuit – V_{DDH} circuit' interface. This means that the power consumption reduced by voltage scaling will be compensated by the extra power consumed on level converters. To avoid this problem, a CVS structure is proposed in [1]. This structure arranges all paths of a circuit in the manner 'primary inputs – V_{DDH} cells – V_{DDL} cells – level converters – primary outputs'. This leads to the formation of the cluster of V_{DDH} cells and that of V_{DDL} cells, as shown in Fig. 2. No level converters are needed elsewhere except at primary outputs because of V_{DDH} cell – V_{DDL} cell connections. The number of level converters is then minimised. The CVS structure has an additional advantage that the layout overhead is also minimised. When performing placement and routing using standard cells, one can have separate groups for the V_{DDH} cells and the V_{DDL} cells, respectively, and then place them in distinct rows to reduce routing overheads.

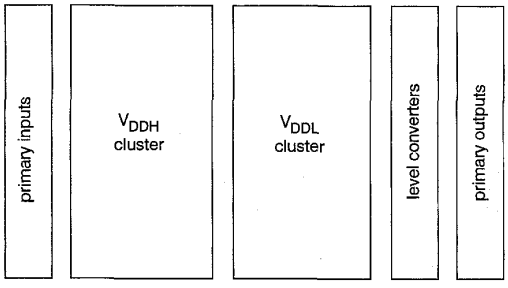

Fig. 2 *Clustered voltage scaling structure*

2.3 Basic algorithm

Assume that all the cells are initially in the V_{DDH} version and that the timing constraints are met. In other words, the slacks of all cells are non-negative. For the gate-level netlist, backward graph-traversal is performed by using the depth-first-search (DFS) algorithm from the primary outputs toward the primary inputs. Each time a cell is visited, the CVS algorithm tries to replace it with its corresponding V_{DDL} version. If the timing constraints are still met after replacement, the cell is replaced. This process is repeated until all the cells in the circuit are visited. However, if any of the slacks is detected to be negative, the replacement process at this cell is not allowed.

A couple of heuristics to determine the replacing order of candidate gates are provided. They are 'larger C' and 'larger Slack', respectively. if the heuristic 'larger Slack' is chosen, the candidate nodes are sorted in descending order with their slacks. The candidate nodes are sorted in descending order with their capacitance when the 'larger C' heuristic is used. The heuristics are applied to the whole circuit instead of only to primary outputs.

In performing the traversal, care should be taken when the visited cell has multiple fan-outs. When trying to replace the cells with multiple fan-outs, the forward traversal using the DFS toward the primary outputs is performed. It checks if the child cells can be replaced with V_{DDL} cells, and is repeated until every child cell is checked.

Finally, both the V_{DDH} cells and the V_{DDL} cells are clustered and the CVS structure is formed.

3 Path-oriented clustered voltage scaling technique

The CVS technique assumes that a circuit is originally operating under the supplied voltage V_{DDH}, and that it is faster than the performance requirement. Therefore, some of the gates in the circuit can be replaced with their corresponding V_{DDL} cells so that the speed of the circuit is just above the speed needed to meet the required performance.

We can consider this problem from another point of view. Assume that the circuit is operating in the lower supplied voltage V_{DDL}, instead of the higher voltage V_{DDH}. If the circuit delay still meets the performance requirement, each gate of the circuit can be in its V_{DDL} version. However, if the circuit delay does not meet the timing constraint, the circuit should be optimised for performance. We can replace some of the gates in the circuit with their corresponding V_{DDH} cells to enhance the performance of the circuit (i.e. trade power for speed). However, the power being traded must be minimised.

We propose the PCVS technique, which adopts the point of view described above. In Section 2 we saw that the CVS technique uses the delay of the longest path to represent the delay of a circuit, and optimises the circuit according to the slack information. We know that the delay of a circuit, which is equal to the delay of the longest sensitisable path, is bounded by the delay of its longest path. Therefore, the delay estimation used by the clustered voltage scaling technique is correct but pessimistic. However, if the circuit is considered to be under V_{DDL} at the beginning, our PCVS technique can take advantage of the path sensitisation criterion and path selection criterion to identify the true critical long paths to be optimised. As the critical path is found, the delay of the circuit, which is equal to the delay of the critical path, can be more accurately calculated. Moreover, as long as a set of long paths is identified, it is possible to apply the optimisation process to each selected path respectively instead of taking the whole circuit into account.

The CVS structure proposed in [1] is preferred for two different applied voltages because it minimises the number of level converters needed. Therefore our PCVS technique will adopt this circuit structure to minimise the power consumed by level converters, and we also propose a clustering algorithm to achieve this. The power reduction of a circuit optimised by the PCVS technique is better than that optimised by the CVS technique because the PCVS technique only considers the long sensitisable paths. We will discuss each step of the PCVS technique in detail in the following Sections.

3.1 Definitions and terminology

The following definitions can be found in [2].

A combinational circuit that is bounded by primary inputs and primary outputs composes of simple gates, which are AND, NAND, OR, NOR, and NOT gates, and leads between gates. The delays of gate G and lead f are denoted by $d(G)$ and $d(f)$.

A *path* $P = (G_0, f_0, G_1, f_1, ..., f_{m-1}, G_m)$ in a combinational circuit is an alternating sequence of leads and gates. Gate G_0 is a primary input and G_m is a primary output of the circuit. Lead f_i, $0 \leq i \leq m-1$, is an *on-input* of P that connects gate G_i and gate G_{i+1}. A lead is called a *side-input* of f_i if it is connected to G_i but not originated from G_{i-1}. The delay of P is the sum of the delays of all the gates and leads along P, and is denoted by $d_{path}(P)$.

A *controlling value* to a gate G is the logic input value that independently determines the output value of G, and is denoted by $c(G)$. For example, if one of the inputs of AND has logic value 0, the output value is determined to be 0. Therefore, the controlling value to AND is 0. A *noncontrolling value* to gate G is denoted by $nc(G)$, which is the complementary value of $c(G)$. Hence the noncontrolling value of AND is the logic value 1.

Lead f, which is connected to gate G, is said to be a *controlling input* to G under the primary input vector v if the stable value at f under v is $c(G)$. On the other hand, f is said to be a noncontrolling input to G under v if the stable value at f is $nc(G)$.

Let the stable value and the stable time at each input to gate G under v be unknown. Lead f, which is connected to G, is considered to *dominate* G if the stable value and the stable time at G are determined by those at f. A path is considered to be *activated by* the primary input vector v if each on-input of the path dominates the gate to which it connects. A path is defined as a *sensitisable path* if there is at least one primary input vector that activates the path. On the contrary, paths that cannot be activated by any primary input vector are called *false paths*. The *critical paths* are the longest sensitisable paths in the circuit.

3.2 Path selection algorithm

There are various path sensitisation criteria with different dominance definitions for the true path selection algorithms. Hence the delay estimated by using different path sensitisation criteria can have different results. Among these criteria, the exact path sensitisation criterion is the tightest criterion. Therefore, any other path sensitisation criterion is considered to be correct if the estimated path delay based on it is equal to or greater than the delay estimated by the exact path sensitisation criterion. As a consequence, an approximate path sensitisation criterion is considered to be correct if the criterion is looser than the exact path sensitisation criterion.

Let v be an applied primary input vector. A lead is considered to dominate the gate to which it connects if it follows the conditions described in each path sensitisation criterion under the primary input vector v.

Definition 1. Exact path sensitisation criterion

A path is considered to be an exact sensitisable path if there is at least one primary input vector, such that each on-input of the path is either the earliest controlling input or the latest noncontrolling input with all its side inputs being noncontrolling inputs too.

Definition 2. The Brand–Iyengar path sensitisation criterion

A path is considered to be a Brand–Iyengar sensitisable path, it there is at least one primary input vector, such that either each on-input of the path is the lowest controlling input or all its side inputs are noncontrolling inputs too.

We modify the path selection algorithm proposed in [2] by adopting the Brand–Iyengar sensitisation criterion for the sake of easier implementation. According to [2], a path sensitisation criterion may not be used directly as the path selection criterion. However, the

Brand–Iyengar criterion is an exception. Note that a path is considered to be a Brand–Iyengar sensitisable path, if there is at least one primary input vector such that either each on-input of the path is the lowest controlling input or all its side-inputs are noncontrolling inputs too. The Brand–Iyengar criterion does not sensitise a path by checking both the stable times and stable values of the fanins of each gate along the path. It identifies a path as sensitisable only by the geometry order of the fanins and their values. Hence during the optimisation process, the true paths selected by the Brand–Iyengar criterion will not become false paths and vice versa. Therefore, the Brand–Iyengar criterion can be used for the path sensitisation as well as the path selection.

Typically, the rising propagation delay and the falling propagation delay of a gate are not equal. Other approaches using the larger of the rising and falling delays to represent the delay of a gate will surely overestimate the delay of the circuit. In fact, the algorithm presented here can correctly handle the situation of different rising delay and falling delay of a given gate. Therefore, the performance of a circuit estimated by this algorithm is more accurate.

The algorithm of path selection is shown in Fig. 3. Without loss of generality the circuit being processed is assumed to have only one primary output, which is denoted by O. For a circuit with m primary outputs, we can simply introduce a dummy primary output O into the circuit and then connect each original primary output to O. It also creates a best-first-search queue (BFSQ) to perform the best-first-search algorithm from the primary outputs to the primary inputs. The algorithm first initialises at the dummy sink with esperance larger than the required delay τ. Note that the *esperance* of a partial path is the path delay of the longest path that contains the partial path. If the esperance of a partial path is longer than the required delay τ, it means that it is possible for the partial path to expand to a complete path with the path delay longer than τ. The rising esperance is the esperance of the partial path such that its stable value is 1; while the falling esperance is the esperance of the partial path such that its stable value is 0. The partial path with the longest esperance can be found at the top of the BFSQ. If it is a complete path it is included in the feasible set FS, and another partial path is selected from the BFSQ. Otherwise, the algorithm tries to expand the partial path with the best lead/gate such that the expanded partial path is the longest untraced path. The stable value of input of the last gate is assigned to the parity of the expanded partial path. If the value is the controlling value of the last gate along the expanded partial path, all the lower side-inputs should be set to noncontrolling values. On the other hand, if the value is the noncontrolling value of the last gate, then all the side-inputs should be set to noncontrolling values. This is based on the Brand–Iyengar criterion. The side-inputs, with their set values are included in the condition set of the expanded partial path. The leads/values of the condition set are propagated forward and backward by the D-algorithm to verify if any inconsistency occurs. If no inconsistency is detected, the expanded partial path is inserted into the BFSQ.

3.3 Path optimisation algorithm

For each selected long path, we apply the path optimisation algorithm to shorten its delay. Since the circuit is optimised to have two different applied voltages, V_{DDH} and V_{DDL}, its performance is certainly upper bounded by the performance when the circuit is only operating under the voltage V_{DDH}, and is lower bounded by the performance when the circuit is operating under V_{DDL} only. Therefore, if the performance requirement of a circuit is met when only V_{DDL} is applied, all the gates in the circuit can surely be in their V_{DDL} versions. Similarly, if the performance requirement of the circuit cannot be met when only V_{DDH} is applied, the timing optimisation using CVS cannot achieve the intended goal.

Initially, all gates of the circuit are in their V_{DDL} versions. Let the selected long path be $P = (G_0, f_0, G_1, f_1, G_2, f_2, \ldots, f_{m-1}, G_m)$, where gate G_0 is a primary input and G_m is a primary output of the circuit. We try to optimise the path and make its delay equal to, or shorter than, the circuit delay requirement. According to the CVS structure, every single path in the circuit should be optimised to the form of 'primary input – V_{DDH} circuit – V_{DDL} circuit – primary output'. In our algorithm, we cluster each path first. The whole circuit

```
FS = φ
If the dummy sink O has falling esperance > τ
  initialise partial path (O) with parity = 0 into BFSQ;
If the dummy sink O has rising esperance > τ
  initialise partial path (O) with parity = 1 into BFSQ;
While (BFSQ is not empty)
  Let P = (f_{i-1}, G_i, f_i, ..., O) be on top of BFSQ;
  While (P = (f_{i-1}, G_i, f_i, ..., O))
    If P is a complete path FS = FS ∪ {P};
    Else
      Let f_{i-2} be an output to G_{i-1}, the best remaining fan-in of G_i;
      ExtP = (f_{i-2}, g_{i-1}) * P;
      ExtP.parity = P.parity ⊕ parity (G_{i-1});
      ExtP.cond = P.cond ∪ {[f_{i-1}, P.parity]};
      mark f_{i-2} as ExtP.parity;
      If (ExtP.parity == c(G_{i-1}))
        For each lower side input f of f_{i-2}
          ExtP.cond = ExtP.cond ∪ {[f, nc(G_{i-1})]};
      Else (ExtP.parity == nc(G_{i-1}))
        For each side input f of f_{i-2}
          ExtP.cond = ExtP.cond ∪ {[f, nc(G_{i-1})]};
      If (ExtP. cond is consistent) insert ExtP to BFSQ;
      If (no remaining fan-in of G_i has esperance > τ)
        remove P from BFSQ;
```

Fig. 3 *Path-oriented selection algorithm with Brand–Iyengar criterion*

```
While (F is not empty)
  Let P = (G0, f0, ..., Gi, Fi, ..., O) be a path in FS;
  For gates from G0 to O along P
    Replace Gi with its VDDH version;
    If the delay of P > τ
      Replace the next gate Gi+1;
    Else
      Remove P from FS;
```

Fig. 4 *Path optimisation algorithm*

is then clustered accordingly. As shown in Fig. 4, we optimise the path by trying to replace the gates along it with their V_{DDH} cells in order from the primary input G_0 toward the primary output G_m. We first replace the gate G_1 with a V_{DDH} cell. if the path delay of P meets the timing constraint, the optimisation process of the path is stopped. However, if the path delay of P does not meet the timing constraint, the algorithm will try to replace the next gate G_2 with its V_{DDH} cell, and so on. The replacement process on the path P is not stopped until the timing constraint is met or the replacement reaches the primary output G_m. If the performance requirement cannot be satisfied even when the whole path is operating in V_{DDH}, the optimisation process fails.

3.4 Clustering algorithm

After the delays of all the selected paths are shortened, the circuit is guaranteed to meet the performance requirement. However after each path being clustered respectively, the whole circuit is not guaranteed to be in the CVS structure. To ensure that the circuit is clustered, each gate in the circuit must obey the following rule: *If a gate is selected to be replaced with a V_{DDH} cell, all its fanin gates must also be replaced with V_{DDH} cells.* As shown in Fig. 5, we use a breadth-first-search algorithm, traversing from the primary outputs toward the primary inputs, and apply the clustering rule to each gate to ensure the circuit will eventually be in the form of the CVS structure.

```
For all gates from Gm to G0
  If Gi is a VDDH cell
    For each fanin Gi-1 of Gi
      If (Gi-1 is a VDDL cell)
        Replace Gi-1 with it VDDH version;
```

Fig. 5 *Clustering algorithm*

The clustering algorithm is different from that of the CVS technique. This is because the PCVS technique does not dynamically update the gate information, as does the CVS. Therefore, it just analyses the circuit to determine whether additional gates should be replaced with their corresponding V_{DDH} cells. On the contrary, the CVS technique determines whether or not a gate should be replaced by using depth-first-search algorithm traversing from the primary inputs toward the primary outputs. Our clustering algorithm is thus more efficient.

4 Experimental results

We have implemented the path-oriented clustered voltage scaling technique in the C language and performed the experiments on the ISCAS85 benchmark circuits on a SUN SPARC–20 workstation.

The delay model of each gate we used here is shown in Table 1. The linear model is used, and the delay of a gate is related to the number of fan-ins and fan-outs in a linear relationship. These equations are extracted from the low power LP900C CMOS standard-cell library of Lucent Technologies. Note that the rising and falling delay of each gate is different, which is the usual case in the real world.

We also assume the delay of a circuit is inversely proportional to the current flowing through it. The relationship between the delay of a V_{DDL} cell and a V_{DDH} cell is shown in eqn. 4.

$$\text{delay}_{V_{DDL}} = \frac{V_{DDL}}{(V_{DDL} - V_{th})^2} \times \frac{(V_{DDH} - V_{th})^2}{V_{DDH}} \times \text{delay}_{V_{DDH}} \quad (4)$$

Because the delay of a circuit is bounded by its delay in V_{DDH} and V_{DDL}, respectively, we divide the delay range identified by the Brand–Iyengar criterion into ten sections and use the delay of each section as the required delay to optimise the circuit: e.g. if the required delay increment is 60%, it means that the delay requirement of the circuit is $\text{delay}_{VDDH} + (\text{delay}_{VDDL} - \text{delay}_{VDDH}) \times 60\%$. Therefore the circuit is optimised by both the

Table 1: Delay model of each gate type

Gate type	Gate delay equation	
AND	falling	1.68 + (#fan-in – 3) × 0.175 + 0.07623 × #fan-out
	rising	2.20 + (#fan-in – 3) × 0.56 + 0.16803 × #fan-out
NAND	falling	1.72 + (#fan-in – 3) × 0.205 + 0.22698 × #fan-out
	rising	0.95 + (#fan-in – 3) × 0.08 + 0.17145 × #fan-out
OR	falling	3.19 + (#fan-in – 3) × 1.01 + 0.09126 × #fan-out
	rising	1.47 + (#fan-in – 3) × 0.175 + 0.07623 × #fan-out
NOR	falling	1.65 + (#fan-in – 3) × 0.07 + 0.09936 × #fan-out
	rising	1.57 + (#fan-in – 3) × 0.3 + 0.48978 × #fan-out
NOT	falling	1.51 + 0.10044 × #fan-out
	rising	0.81 + 0.17145 × #fan-out
BUFFER	falling	2.00 + 0.002484 × #fan-out
	rising	1.94 + 0.003645 × #fan-out

CVS and the PCVS techniques over the whole possible delay range.

The power of a given gate can be calculated by eqn. 5,

$$P_{dyn} = \left(\frac{1}{2}CV_{dd}^2\right)\alpha f \tag{5}$$

Therefore, the relationship between the power consumption when operating in V_{DDH} and the power consumption when operating in V_{DDL} is:

$$\text{power}_{V_{DDL}} = \left(\frac{V_{DDL}}{V_{DDH}}\right)^2 \times \text{power}_{V_{DDH}} \tag{6}$$

Assume the number of V_{DDH} cells is H and the number of V_{DDL} cells is L, the power reduction ratio can be computed by eqn. 7:

$$\text{power reduction ratio} = \frac{\left(1 - \left(\frac{V_{DDL}}{V_{DDH}}\right)^2\right)L}{L + H} \tag{7}$$

We conduct the experiment with V_{DDH} = 5V, V_{DDL} = 3V and V_{th} = 0.7V. The delay of each gate is calculated according to Table 1 and eqn. 4, and the power reduction ratio can be estimated by eqn. 7.

Table 2: Experimental results of c1908

Circuit name		c1908		
Total number of gates		880		
Circuit delay in 5V (ns)		59.99		
Circuit delay in 3V(ns)		125.80		
Clustered voltage scaling technique				
(%)	Number of replaced gates	Power reduction ratio (%)	CPU time	
0	—	—	< 1	
10	—	—	< 1	
20	—	—	< 1	
30	—	—	< 1	
40	371	26.98	< 1	
50	391	28.44	1	
60	395	28.73	1	
70	402	29.24	1	
80	484	35.20	2	
90	538	39.13	3	
100	586	42.62	6	
Path-oriented CVS technique				
(%)	Number of paths found	Number of replaced gates	Power reduction ratio (%)	CPU time
0	347 732	228	16.58	3274
10	337 321	372	27.05	3202
20	319 345	384	27.93	3032
30	292 383	486	35.35	2849
40	252 015	551	40.07	2489
50	188 352	641	46.62	1910
60	128 513	710	51.64	1310
70	68 951	774	56.29	692
80	17 807	815	59.27	186
90	1 042	863	62.76	38
100	0	880	64.00	25

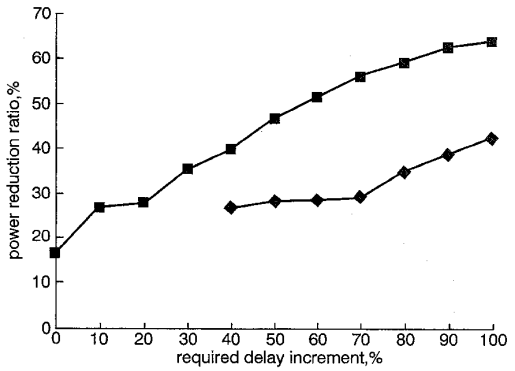

Fig. 6 *Experimental result of c1908*
—◆— CVS; —■— PCVS

The PCVS technique can identify the false paths, as well as consider the situation with different rising delays and falling delays. Therefore, an improvement in PCVS techniques over the CVS technique is expected. Let's use c1908 as an example for comparison first. In Table 2 and Fig. 6, we can see that the power reduction ratio of PCVS technique is much larger than that of the CVS technique over all possible delay ranges. Table 3 shows that the performance of the circuit estimated by the PCVS technique is not as pessimistic as the CVS technique. The average results of the nine benchmark circuits are shown in Table 4 and Fig. 7 over the ten delay ranges. Table 5 shows the normalised power consumption with respect to the original circuit, while Table 6 shows the normalised power consumption with respect to the circuit optimised by the CVS technique. We can see that the PCVS technique adds another 22% power reduction ratio on average over that of the clustered voltage scaling technique.

Table 3: Ratio of the highest speed estimated by the CVS and the PCVS

	$\frac{\text{HighestSpeed}_{CVS}}{\text{HighestSpeed}_{PCVS}}$		$\frac{\text{HighestSpeed}_{CVS}}{\text{HighestSpeed}_{PCVS}}$
c432	0.83	c2670	0.87
c499	0.83	c3540	0.84
c880	0.87	c5315	0.82
c1355	0.84	c7552	0.81
c1908	0.75		

Table 4: Average experimental results

Required delay increment (%)	Average power reduction ratio	
	CVS (%)	PCVS (%)
0	—	15.013
10	—	22.959
20	6.353	30.020
30	21.570	34.428
40	29.137	40.506
50	33.499	44.962
60	36.438	51.310
70	40.910	55.990
80	44.424	59.053
90	49.733	61.154
100	53.384	64.00
Average	28.677	43.581

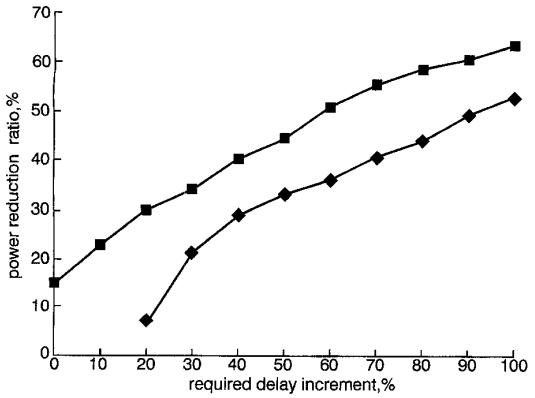

Fig. 7 *Average experimental results*
—◆— CVS; —■— PCVS

Table 5: Normalised power consumption with respect to the original circuit

Required delay increment (%)	Original circuit	Optimised by the CVS technique	Optimised by the PCVS technique
0	1	—	0.850
10	1	—	0.770
20	1	0.936	0.700
30	1	0.784	0.656
40	1	0.709	0.595
50	1	0.665	0.550
60	1	0.636	0.487
70	1	0.591	0.440
80	1	0.556	0.409
90	1	0.503	0.388
100	1	0.466	0.360
Average	1	0.713	0.564

Table 6: Normalised power consumption with respect to the CVS technique

Required delay increment (%)	Original circuit	Optimised by the CVS technique	Optimised by the PCVS technique
0	—	—	—
10	—	—	—
20	1.068	1	0.748
30	1.276	1	0.837
40	1.410	1	0.839
50	1.504	1	0.827
60	1.572	1	0.766
70	1.692	1	0.745
80	1.799	1	0.736
90	1.988	1	0.771
100	2.146	1	0.773
Average	1.606	1	0.782

5 Conclusions

In this paper, we discussed the idea of the CVS technique for low power. We then proposed the PCVS technique to improve the power reduction of a circuit. It combines the voltage scaling technique, and the long sensitisable paths identification, to achieve the CVS structure and get more power reduction. The PCVS algorithm has been verified with the ISCAS85 benchmark circuits. The experimental results show that more gates can be voltage scaled down as compared with the CVS technique.

6 Acknowledgments

Thanks to Dr Hsi-Chuan Chen for his fruitful discussion and help in software development.

7 References

1 USAMI, K., and HOROWITZ, M.: 'Clustered voltage scaling technique for low-power design'. ISLPD'95, 1995
2 CHEN, H.–C., DU, D.H.–C., and LIU, L.–R.: 'Critical path selection for performance optimisation', *IEEE Trans.*, 1993, **CAD-12**, (2), pp. 185–195
3 IN, H.–R., and HWANG, T.T.: 'Dynamical identification of critical paths for iterative gate sizing'. Proceedings of IEEE/ACM International Conf. on Computer-Aided Design, 1994,
4 CHEN, H.–C., and DU, D.H.–C.: 'Path sensitization in critical path problem', *IEEE Trans.*, 1993, **CAD–12**, (2), pp. 196–207
5 DU, D.H.–C., YEN, S.H.C., and GHANTA, S.: 'On the general false path problem in timing analysis'. Proceedings of 26th *Design automation* conference, 1989, pp. 555–560
6 JONE, W.–B., and FANG, C.–L.: 'Timing optimization by gate resizing and critical path identification'. Proceedings of 30th *Design automation* conference, 1993, pp. 135–140
7 CHANDRAKASAN, A.P., SHENG, S., and BRODERSEN, R.W.: 'Low-power CMOS digital design', *IEEE J. Solid-State Circ.*, 1992, **27**, (4), pp. 473–483
8 CIRIT, M.: 'Transistor sizing in CMOS circuits',Proceedings of 24th *Design automation* conference, 1987, pp. 121–124
9 DEVADAS, S., KEUTZER, K., and MALIK, S.: 'Delay computation in combinational logic circuits: theory and algorithms'. Proceedings of the IEEE ACM International Conf. on *Computer-aided design*, 1991, pp. 176–179
10 GASSER, L., and HOYTE, L.: 'Delay and power optimization in VLSI circuit'. Proceedings of the 21st *Design automation* conference, 1984, pp. 529–535
11 PERREMANS, S., CLAESEN, L., and DEMAN, H.: 'Static timing analysis of dynamically sensitizable paths'. Proceedings of the 26th *Design automation* conference, 1989, pp. 568–573
12 BRAND, D., and IYENGAR, V.: 'Timing analysis using functional analysis'. IBM Thomas J. Watson Research Centre, technical report, 1986