Layout Techniques Supporting the Use of Dual Supply Voltages for Cell-Based Designs

Chingwei Yeh, Yin-Shuin Kang
Shan-Jih Shieh, Jinn-Shyan Wang
EE Dept., Nat'l Chung-Cheng Univ., Chiayi 621, Taiwan, ROC
principal author's email: ieecwy@ccunix.ccu.edu.tw

Abstract

Gate-level voltage scaling is an approach that allows different supply voltages for different gates in order to achieve power reduction. Previous researches focused on determining the voltage level for each gate and ascertaining the power saving capability of the approach via logic-level power estimation. In this paper, we present the layout techniques that feasiblize the approach in cell-based design environment. A new block layout style is proposed to support the voltage scaling with conventional standard cell libraries. The block layout can be automatically generated via a simulated annealing based placement algorithm. In addition, we propose a new cell layout style with built-in multiple supply rails. Using the cell layout, gate-level voltage scaling can be immediately embedded in a typical cell-based design flow. Experimental results show that proposed techniques produce very promising results.

1 Introduction

Power consumption has become a critical design concern due to the increasing gap between the energy required by portable computation/communication devices and the energy supplied by the battery. In addition, as the number of devices packed into a single chip approaching millions, heat dissipation becomes a problem that can adversely affect the reliability and packaging cost of a design. These factors have driven numerous research efforts to address various kinds of power-saving techniques [1].

The primary source of power consumption for a CMOS design comes from the switching of logic states. The switching power is expressed as [1]

$$P_{switch} = \alpha_{0\rightarrow 1} * f_{clk} * (C_{load} * V_{dd}^2) \qquad (1)$$

where $\alpha_{0\rightarrow 1}$ is the average number of times in a clock cycle that a switch from 0 to 1 occurs, f_{clk} is the clock frequency, and C_{load} is the loading capacitance. The equation clearly shows the supply voltage (V_{dd}) affects power dissipation in a quadratic order. Thus, voltage scaling has been deemed a major way to reduce power consumption [1]. Among the various voltage scaling approaches, *gate-level* voltage scaling is the one that allows different supply voltages for different gates in the same circuit. The approach is more aggressive than the block-level voltage scaling, where different supply voltages are allowed *only* in different blocks of a design [4, 2, 6, 7].

The advantage of gate-level voltage scaling has been studied in previous researches [9, 12, 10]. Algorithms have also been developed [9, 12, 10] to apply dual supply voltages on the same circuit. Despite these work, however, there are layout issues that must be carefully considered in order to realize the desired voltage scaling for cell-based designs. This is because in conventional standard cell designs, the positions of power and ground lines are fixed at the top and bottom of the cell layout [11]. Hence, cells in the same row can be abutted. When there are multiple supply voltages, cells of different voltage levels can not be abutted, as this would lead to electrical short.

In this regard, we present the layout techniques that feasiblize gate-level voltage scaling for cell-based designs. We first propose a new block layout style that makes voltage scaling applicable for those designs that employ conventional standard cell libraries. Then, we propose a simulated annealing based placement algorithm to generate the proposed layout. In addition, we propose a new cell layout style with built-in multiple supply rails. Using the cell layout, gate-level voltage scaling can be immediately embedded in a typical cell-based design flow. The proposed techniques are all evaluated using benchmark circuits and have shown very promising results.

2 Realizing Voltage Scaling with Placement Control

To avoid electrical short, cells of different voltages can not be abutted as in the conventional standard cell layout. This section presents the block layout styles that achieve the separation of cells of different voltages. Then, a simulated annealing based method is proposed to automatically generate the desired layout. Finally, empirical results are given to demonstrate the effectiveness of the proposed method.

2.1 Block Layouts Achieving Voltage Separation

There are several ways to achieve voltage separation. The simplest one is shown in Fig. 1 [10]. In the layout, cells of the same voltage are grouped together to form a macro block. Each macro block has it own supply voltage. Within the macro block, cells are placed as in conventional standard cell design.

The most appealing reason for using the macro block layout style is that existing P&R tools can be used directly for intra-block layout. However, the simplicity comes at the expense of inter-block layout. Many cells that were adjacent in the logic schematic may be well separated in different blocks. In this case, except for cells lying on the boundary of blocks, long wires are necessary to realize the interconnection between high-voltage cells and low-voltage ones.

Permission to make digital/hardcopy of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 99, New Orleans, Louisiana
(c) 1999 ACM 1-58113-109-7/99/06..$5.00

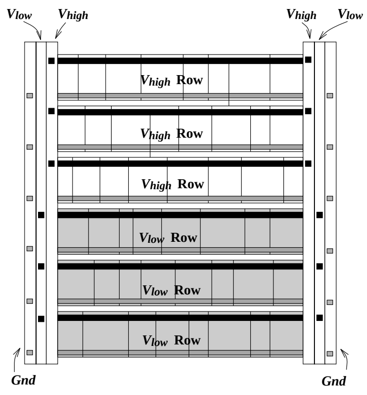

Figure 1: Macro Block Layout Schemes for Voltage Separation

The macro block layout style can be improved by interleaving the rows (Fig. 2) [10]. There, voltage separation is achieved by rows rather than blocks. Accordingly, each cell row can only have one supply voltage. By interleaving the rows, the length of the ”boundary” between the high-voltage block and the low-voltage block is increased, which contributes to less demand for long wires and greater chance for layout optimization.

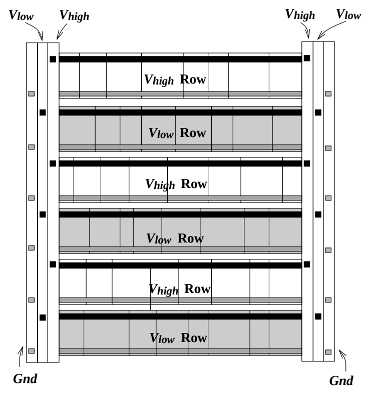

Figure 2: Interleaved Layout Scheme for Voltage Separation

Nevertheless, there are difficulties associated with practicing the layout. To do the interleaving, we can start with placed macro blocks and proceed with the interleaving as doing a 1-D placement. However, the row-to-row interconnection is much more complicated than the node-to-node interconnection model in the conventional 1-D placement formulation. Consequently, it is not trivial to set a proper objective to guide the placement towards global optimization. On the other hand, if we start with unplaced logic schematic, we have to provide a placement tool so that the one-voltage-per-row rule is strictly conformed. However, as will be elaborated shortly, there is another layout style which *subsumes* the interleaved layout with a greater optimization dimension, and yet requires the similar tooling effort. Owing to this reason, the interleaved layout will not be used in our work.

To achieve voltage separation while at the same time relieving the layout from unnecessary constraints, we propose the layout shown in Fig. 3. The layout does not require a uniform voltage across the entire row. Instead, up to two voltages can be accommodated in the same row. If two voltages are present in the same row, then each voltage occupies either the left or right part of the row. In the figure, we have shown four possible voltage distributions for a row, namely, all V_{high}, all V_{low}, $V_{high} \rightarrow V_{low}$, and $V_{low} \rightarrow V_{high}$. The necessary voltage separation is achieved via the insertion of feedthroughs.

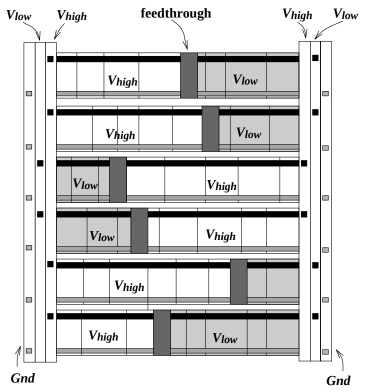

Figure 3: The Proposed Block Layout Scheme

It can be seen that, comparing to the previous layouts, the proposed layout imposes the least constraint and hence leaves the greatest freedom for the placement tool. Whether the freedom can be fully turned into layout's advantage now depends on the capability of the placement tool. The next section describes how to modify a given placement method to make it sensible to the proposed layout style.

2.2 A Simulated Annealing Based Placement Control

We use TimberWolf6.0 [8], which is a simulated annealing [3] based method, to do the placement. Unlike a plain, generic simulated annealing method, the TimberWolf package has gone through several revisions and tunings to yield good placement results. Hence, we choose not to make dramatic changes to the package. Instead, we modify the cost function of each move so that moves contributing to voltage clustering are *gradually encouraged* as the annealing temperature decreases.

The modification is depicted in the following pseudo code. For ease of explanation, we assume the positions on a row are numbered ascendingly from left to right, beginning with the number 0. Also, since the inequality of cell widths has been well treated by the TimberWolf package, we will ignore the cell width in the computation of cost associated with voltage clustering.

```
NewCost($cell, row, post, moveDir, orgCost, iterG$) {
    $rowPol = desired\_polarity(row)$;
    $midPt = midpoint(row, rowPol)$;
    $postCost = compute\_postCost(row, rowPol, post, midPt)$;
    $nbrCost = compute\_nbrCost(row, post, moveDir)$;
    $baseCost = postCost + nbrCost$;

    /* scaling $|baseCost|$ towards $|orgCost|$, illustrated in Fig. 4 */
    if ($iterG \leq MIDSTART$)
        $voltCost = baseCost * (0.5 * |orgCost|)$;
    else if ($MIDSTART < iterG < MIDEND$)
        $voltCost = baseCost*$
            $((0.5 + 1.5 * \frac{iterG-MIDSTART}{MIDEND-MIDSTART}) * |orgCost|)$
    else if ($MIDSTART \geq MIDEND$)
        $voltCost = baseCost * (2.0 * |orgCost|)$;
    return $voltCost + orgCost$ ;
}
```

The procedure $NewCost$ takes as inputs a cell, the target row, the target location of the cell on the target row ($post$), the move di-

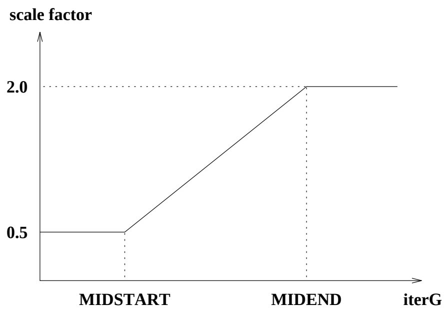

Figure 4: Scaling baseCost to get voltCost

rection ($moveDir$: 1 for in and -1 for out), the original cost computed by TimberWolf ($orgCost$), and the current iteration number from TimberWolf ($iterG$). Also, the two constants ($MIDSTART$ and $MIDEND$) are pre-defined in the TimberWolf package to mark the start and end of the middle temperature region. The philosophy of $NewCost$ is to separately compute the cost associated with a position on the row ($postCost$), and the cost associated with the neighboring cell types ($nbrCost$). For ease of adjustment, all costs are first composed via simple numerical values like 1.0 and 0.5. Then, a scale function is used to make the initial costs comparable to the inherent TimberWolf cost value.

To derive $postCost$, procedure $NewCost$ starts with evoking $desired\_polarity$ to determine the desired polarity of the row. There are only four possible polarities for a row: all V_{high}, all V_{low}, $V_{high} \rightarrow V_{low}$, and $V_{low} \rightarrow V_{high}$. The desired polarity is the one that is *closest* to the current voltage distribution. The polarity is obtained via comparing the mean positions of the high-voltage and low-voltage cells on the row. For example, if the mean position of the high-voltage cells is less than that of the low-voltage cells, then $rowPol = \langle V_{high} \rightarrow V_{low} \rangle$. Fig. 5 illustrates the polarity computations.

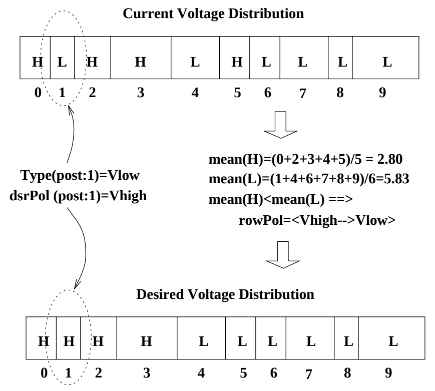

Figure 5: Polarity Computation

Having fixed the row polarity, the next thing is to determine the desired boundary point ($midPt$) in the row *if* the cells were grouped in clusters according to the row polarity. Let us assume that $rowPol = \langle V_{high} \rightarrow V_{low} \rangle$, as shown in Fig. 5. Then, for each position $post$ to the left of $midPt$, we assign V_{high} as the desired polarity, i.e., $dsrPol(post) = V_{high}$ if $post \leq midPt$. The positions where $post > midPt$ are assigned with V_{low} as their desired polarity. Thus, in contrast to the four possible polarities for a row, there are only two possible polarities: V_{high} and V_{low}, for each position on the row.

Since different types of cells (V_{high} or V_{low}) moving into the same position produces the opposite effects, we define

$$matchCost(post, cell) =$$

$$\begin{cases} -moveDir & \text{if } dsrPol(post) = Type(cell) \\ moveDir & \text{if } dsrPol(post) \neq Type(cell) \end{cases}$$

The formula states two matching situations. One is moving a cell *into* a position whose desired polarity equals the type of the cell. The other is moving a cell *out of* a position where the type of the cell is not desired by the position. In both cases, $matchCost(post, cell)$ returns a value of -1, where a minus sign represents a favorable situation. Otherwise, there is no match and $matchCost(post, cell)$ returns a value of 1.

With $matchCost(post, cell)$ and $midPt$, we are now ready to compute $postCost$, as shown below:

$$postCost =$$

$$\begin{cases} \frac{midPt - post}{midPt} * matchCost(post, cell) & \text{if } post \leq midPt \\ \frac{post - midPt}{rowlength - midPt} * matchCost(post, cell) & \text{if } post > midPt \end{cases}$$

The above formula intends to make moves associated with $midPt$ a $postCost$ value of 0, and to linearly scale the cost towards 1 or -1.

Next, we determine the cost associated with the neighbors of the intended position. Let $lcell$ and $rcell$ denote the left and right neighbors. If $Type(lcell) \neq Type(rcell)$, then from the viewpoint of doing voltage clustering among $lcell$, $rcell$, and $cell$, the type of $cell$ does not matter. Thus, $nbrCost$ is assigned a value of 0. If $Type(lcell) = Type(rcell) = Type(cell)$, then moving *in* $cell$ reinforces the clustering, while moving *out* $cell$ weakens the clustering. Thus, the former should be assigned a negative cost, while the latter a positive cost. To make $nbrCost$ comparable to $postCost$, we use $\frac{1}{2}(max.|postCost|) = 0.5$ as the base value. That is, the cost of moving in $cell$ is -0.5, while that of moving out $cell$ is 0.5. The reverse situation is found when $Type(lcell) = Type(rcell) \neq Type(cell)$. These rules are summarized in the following table.

Case	$moveDir$	
	1(in)	-1(out)
$Type(lcell) = Type(rcell) = Type(cell)$	-0.5	0.5
$Type(lcell) = Type(rcell) \neq Type(cell)$	0.5	-0.5
otherwise	0	0

The $baseCost$ can now be determined as the sum of $nbrCost$ and $postCost$. We then scale $baseCost$ to make it comparable to the original cost computed by TimberWolf ($orgCost$), producing the final $voltCost$. A linear formula shown in Fig 4 is used to scale $baseCost$. Various bounds other than the used ones (0.5 and 2.0) have been tried, but returned no conclusive advantage. The sum of $voltCost$ and $orgCost$ is fed back the to the TimberWolf package for placement control.

The scaling of the cost function displays a balance between enforcing the voltage clustering and generating a compact layout. Consequently, there is a chance that some cells are not conformed to the clustering at the end of the annealing process. For each such cell, we first identify the positions whose polarities match with the type of the cell. Then, among these positions, we select the one with the least wiring cost and forcibly move the cell into the position.

2.3 Empirical Evaluation

We take the gate-level voltage scaling results from [12] and use $TWvolt$ to place them. In [12], the test circuits were dissected into two voltage clusters: one operates with $V_{dd} = 5.0V$, and the other operates with $V_{dd} = 3.8V$. To capture the true layout overhead, we use $TWvolt$ only for placement, and let the Cadence Cell Ensemble to finish the rest of the design. Fig. 6 shows the layout flow.

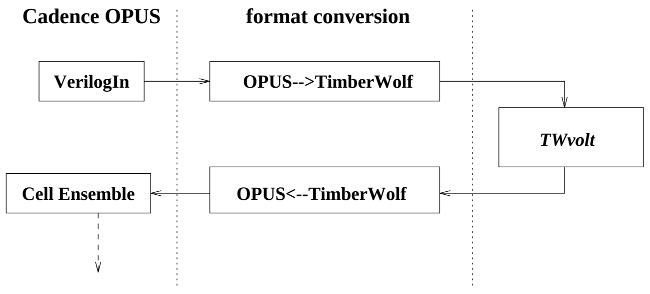

Figure 6: The Layout Design Flow

Table 1 and 2 list the experimental results. The data recorded under "TW" in column 2 are the core sizes and wire lengths of the circuits placed using the original TimberWolf, i.e., without regard to voltage clustering. For $TWvolt$, we record the core sizes and the wire lengths in column 3. The overhead in terms of core sizes and wire length are listed in column 4.

Table 1: Evaluation of Physical Clustering–Core Size(μm^2)

circuit	TW	$TWvolt$	Over(%)
b9	71088.50	80178.92	12.79
frg1	81442.33	90383.54	10.98
c8	77983.83	87122.34	11.72
apex7	142071.34	147432.85	3.77
c880	203587.20	232710.00	14.30
i6	242924.05	259727.32	6.92
apex6	484800.32	486836.48	0.42
rot	442953.15	493438.62	11.40
frg2	664830.71	669750.45	0.74
i7	380407.75	456945.83	17.34
i9	362605.21	393115.42	8.41
i8	626453.10	742636.22	18.55
c5315	1017600.44	1124439.92	10.50
i10	1839997.50	1930644.02	4.93
average			9.48

In core size comparison, 6 out of 14 cases (apex7, i6, apex6, frg2, i9, i10) experience less than 10% overhead. Particularly, on circuits apex6 and frg2, the overhead is almost negligible (both less than 1%). On the other hand, 8 out of 14 cases experience more than 10% overhead. Among the 8 cases, 6 of them (b9, frg1, c8, c880, rot, c5315) do not deviate much from 10%. The worst situation occurs on circuits i7 and i8, where 18.55% and 17.34% overheads are observed. Averagely, the incorporation of voltage clustering into the placement imposes 9.48% core size overhead.

In wire length comparison, 8 out of 14 cases (frg1, c8, c880, i6, apex6, frg2, i9, i10) experience less than 10% overhead. The minimum overhead occurs on circuit apex6, where 3.96% increase is obtained. On the other hand, there are 6 cases (b9, apex7, rot, c5315, i7, i8) where more than 10% overhead is experienced. However, as in the core size comparison, there are only 2 cases (i7, i8) with large wire length overhead (19.94% and 23.18%). The rest of the cases (b9, apex7, rot, c5315) all have wire length increase not far beyond 10%. Averagely, the incorporation of voltage clustering into the placement imposes a 10.52% wire length overhead.

Table 2: Evaluation of Physical Clustering–Wire Length (μm)

circuit	TW	$TWvolt$	Over(%)
b9	13677.68	15218.19	11.26
frg1	17447.55	18879.80	8.20
c8	16228.06	17817.18	9.79
apex7	30690.45	34732.41	13.17
c880	53525.40	58306.50	8.93
i6	64383.83	68627.54	6.59
apex6	153208.10	159278.42	3.96
rot	146658.90	162005.25	10.46
frg2	218812.13	231129.85	5.63
i7	107246.75	132109.57	23.18
i9	118091.58	125387.65	6.18
i8	229496.42	275253.81	19.94
c5315	364646.34	412195.45	13.04
i10	739900.42	791189.62	6.93
average			10.52

The above data demonstrate that except on circuits i7 and i8, $TWvolt$ has rather consistent performance. In other words, with 85.71% (12/14) confidence, we can say that 10% layout quality degradation can be expected when using $TWvolt$.

Lastly, by incorporating voltage clustering computation in the TimberWolf, run time penalty is inevitable. Based on the 14 benchmark circuits, we measured the average running time of $TWvolt$ to be 5 times longer than the original TimberWolf. We expect the figure to be reduced upon further fusion of the voltage clustering computation with the TimberWolf internal data structure and routines.

3 Realizing Voltage Scaling with New Cell Layout

The block layout scheme proposed in the previous section requires detail placement control. For the cases where such control can not be achieved, this section propose a cell layout style that achieves the separation of V_{high} and V_{low} without any modification to the existing placement tool.

We use the layout style proposed in [11] as the base. In the cell layout, transistors are placed in two parallel rows. All P-type transistors are in the upper row while all N-type transistors are in the lower row. The power(ground) lines of all cells are aligned at the position above(below) the P-row(N-row). To utilize the existing P&R tools for layout design, arbitrary cell abuttment must be supported while at the same time achieving the voltage separation. The requirement leads naturally to the layout where separate power rails (in the same cell) are used to carry separate supply voltages. Thus, each cell in the original library is transformed into two cells in the new library: the high-voltage(H) type and the low-voltage (L) type. In the transformation, the internal layout of the cell is copied to the H-cell and L-cell. The only difference between the generated H-cell and L-cell is the connection to the power rails. Fig. 7 shows the layouts of the H-type and L-type 2-input NAND gates.

Using these library cells, it is not necessary to have any physical clustering related to different voltages. In other words, the physical layout has nothing to do with the clustering prescribed by gate-level voltage scaling. This produces two advantages. One is that the routing between adjacent rows will not be compromised. The other is that the P&R tools are able to maintain their full optimization strength. Also, being free from layout tool intervention, the approach is very attractive for situations where layout tool modification presents a major design concern.

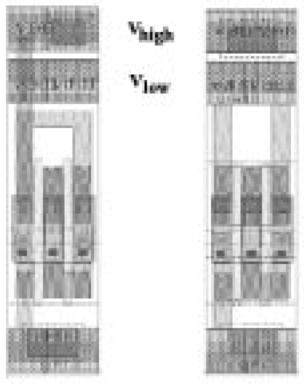

Figure 7: H-type and L-type 2-input NAND Layouts

Note that by adding a second power rail, the height of the cell has been increased. In an original one-power-rail standard cell design using TSMC 0.6μm SPDM technology, the width of the power line is 4μm and the cell height is 30μm. Hence, adding another power line *directly* should result in $\frac{(0.8+4)}{30} \times 100\% = 16.0\%$ area overhead, where the 0.8μm is the required spacing between two power rails. The above calculation is based on the assumption that both the high and low power lines carry the same amount of current. In fact, the current of the latter is usually lower than the former, meaning that the width of the low power line can be reduced. To capture the actual reduction, we run PowerMill [5] on circuits realized with one-power-rail standard cells. The empirical data show that when the supply voltage is reduced from 5.0V to 3.8V, the current also reduces to approximately 60% of the original value. Thus, for $V_{high} = 5.0V$ and $V_{low} = 3.8V$, we only need to add a $4 \times 60\% = 2.4\mu$m wide power rail. The area overhead then becomes $\frac{(0.8+2.4)}{30} \times 100\% = 10.6\%$.

Determining the width of the second power rail via the above procedure is the right way to push down the overhead of the cell layout approach. The strategy produces a dual-power library that is optimized for a specific voltage value. Once completed, however, the library is not applicable for other V_{low} values that are greater than the target one. In other words, if the V_{low} value is raised, the whole library must be re-designed, which involves power width determination, cell layout modification, cell characterization, etc. Redesign is also necessary for a different V_{high}–V_{low} setting. These overhead must be incorporated in assessing the value of the cell layout approach.

4 Placement Control or Cell Layout — Which Way to Go

At this point, we have presented two approaches for realizing gate-level voltage scaling. The first one comprises a new block layout style in conjunction with a simulated annealing based placement strategy for proper voltage separation. The approach allows the use of conventional standard cells, yet requires layout tool intervention. The second approach consists of a new cell layout style which achieves voltage separation via adding a power line. The approach allows the use of the existing P&R tools at the expense of cell layout modification. Both approaches have their strength and weakness and so deserve further comparison.

The voltage scaling results from [12] are used again as the test bench. A new cell library composed of dual power rails were developed. The experimental evaluation is shown in Table 3. The results of "$TWvolt$" were obtained via $TWvolt$ using the single power rail standard cells, and then via Cadence Cell Ensemble for final layout. Those of "CellLayout" were obtained via TimberWolf using the dual power rail standard cells, and again via Cadence Cell Ensemble for final layout. For each case, we highlight the one that produces the better layout quality.

Table 3: Comparions Between Two Physical Approaches

circuit	Core Size (μm^2)		Wire Length (μm)	
	$TWvolt$	CellLayout	$TWvolt$	CellLayout
b9	**80178.92**	94369.88	**15218.19**	16452.10
frg1	**90383.54**	110740.19	**18879.80**	20278.30
c8	**87122.34**	109680.80	**17817.18**	18951.00
apex7	**147432.85**	175579.12	34732.41	**34117.68**
c880	**232710.00**	249040.53	58306.50	**57771.58**
i6	**259727.32**	281573.54	**68627.54**	69268.92
apex6	486836.48	**437661.08**	159278.42	**128647.95**
rot	**493438.62**	551228.73	162005.25	**160532.48**
frg2	**669750.45**	744926.52	**231129.85**	233310.32
i7	**456945.83**	503812.07	132109.57	**121368.95**
i9	**393115.42**	451354.74	125387.65	**124204.75**
i8	**742636.22**	848727.11	275253.81	**272960.03**
c5315	**1124439.92**	1226661.70	412195.45	**397604.46**
i10	1930644.02	**1765160.02**	791189.62	**680275.19**

In core size comparision, there are 12 out of 14 cases where $TWvolt$ outperforms the cell layout approach. The only two circuits that favor the cell layout approach are apex6 and i10. Apparently, for cell layout approach, the height increase of individual cells is the major pain. This height increase is *permanent* in the sense that it can not be reduced through any layout manipulation. The placement approach ($TWvolt$), on the other hand, has the freedom of packing the layout. This leads to its superior core size in most of the cases.

In wire length comparison, there are 9 out of 14 cases where the cell layout approach delivers better results. In particular, the wire length advantage of the cell layout approach seems to be in accordance with the size of the circuit. This is probably due to that there is no voltage clustering required in the cell layout approach. Thus, the placement process is capable of exerting its full optimization strength. Moreover, as the size of the circuit increases, the optimization strength becomes more critical for a good layout. Hence, it is conceivable that the cell layout approach should be better for larger test cases.

Finally, since the intention of voltage scaling is to save power, it is necessary to see whether the power reduction claimed by the logic-level voltage scaling tools [9, 12, 10] would diminish after layout. Thus, we obtain the post-layout power consumptions. The results are shown in Table 4. All power values are obtained via the PowerMill [5] package. The column under "Org" records the post-layout power consumption of the original circuits. These circuits have only one supply voltage and are placed using the TimberWolf. The same circuits are processed via the gate-level voltage scaling method proposed in [12] to produce dual-voltage circuits. For the placement approach, we realize these circuits via single-power-rail standard cells and then place them via $TWvolt$. For the cell layout approach, we realize these circuits via double-power-rail standard cells and then place them via TimberWolf. All placements go through Candence Cell Ensemble to generate the routed layout.

In the table, it can be clearly seen that in 10 out of 14 cases, the cell layout approach produces better power reduction. There are four cases (b9, frg1, c8, frg2) in which $TWvolt$ delivers better result. The data also correlates approximately with the wire length comparison shown in Table 3, where the cell layout approach is shown to be superior in large test cases.

Table 4: Post-Layout Power Comparison

circuit	Power(mW)		
	Org	$TWvolt$	CellLayout
b9	4.27	**3.54**	3.63
frg1	5.68	**4.63**	4.97
c8	7.18	**5.60**	5.73
apex7	12.19	10.21	**9.97**
c880	18.84	15.70	**15.06**
i6	19.75	16.49	**16.25**
apex6	36.41	24.68	**23.23**
rot	35.59	28.20	**27.14**
frg2	38.85	**28.58**	28.60
i7	34.93	29.66	**28.26**
i9	44.39	31.07	**30.59**
i8	61.74	54.26	**53.41**
c5315	134.82	109.56	**103.64**
i10	142.43	110.19	**94.45**

5 Conclusion

We have presented the layout techniques that realize gate-level voltage scaling in cell-based design environment. A new block layout style and the associated placement method are proposed to support voltage scaling for designs using conventional standard cell libraries. A new cell layout style is also proposed that makes gate-level voltage scaling immediately applicable in a typical cell-based design flow.

Our future work is to improve the voltage clustering computation so that the running time of $TWvolt$ can be reduced. On the other hand, we are seeking ways to automate the process of creating the dual-power library, including power width determination, cell layout modification, and cell characterization.

References

[1] Chandrakasan, A. P. and Brodersen, R. W. *Low-power CMOS digital design*. Kluwer Academic Publishers, 1995.

[2] Chang, J. M. and Pedram, M. Energy minimization using multiple supply voltages. *Proc. 1996 Int. Symp. on Low Power Electronics and Design*, pp. 157-162, 1996.

[3] Kirkpatrick, S. et. al., Optimization by Simulated Annealing. *Science*, Vol. 220, May 1983, pp. 671-680.

[4] Raje, S. and Sarrafzadeh, M. Variable voltage scheduling. *Int. Symp. on Low Power Design*, 1995, pp. 9-14.

[5] Deng, C. Power analysis for CMOS/BiCMOS circuits. *Proc. Int. Workshop on Low Power Design*, Apr. 1994, pp. 3-8.

[6] Johnson, M. C. and Roy, K. Optimal selection of supply voltages and level conversions during data path scheduling under resource constraints. *Proc. Int. Conf. on Computer Design*, pp. 72-77, 1996.

[7] Johnson, M. C. and Roy, K. Scheduling and optimal voltage selection for low power multi-voltage DSP datapaths. *Proc. Int. Symp. on Circuits and Systems*, vol. 3, pp. 2152-2155, 1997.

[8] Sechen, C. et al., TimberWolf: mixed macro/standard cell floorplanning, placement, and routing package. Yale University, May 1, 1992.

[9] Usami, K. and Horowitz, M. Clustered voltage scaling technique for low-power design. *Int. Symp. on Low Power Design*, 1995, pp. 3-8.

[10] Usami, K. et al. Automated Low-Power Technique Exploiting Multiple Supply Voltages Applied to a Media Processor. *IEEE J. Solid-State Circuits*, vol. 33, No. 3, Mar. 1998, pp. 463-472.

[11] Uehara, T. and van Cleemput, W. M. Optimal layout of CMOS functional arrays. *IEEE Trans. Comput.*, vol. C-30, pp. 305-312, May 1981.

[12] Chang, M. C., *Master Thesis*, EE Dept., Nat'l Chung-Cheng Univ., Jul. 1997.