Layout Level Timing Optimization by Leveraging Active Area Dependent Mobility of Strained-Silicon Devices

Ashutosh Chakraborty Sean X. Shi David Z. Pan
Department of Electrical and Computer Engineering
The University of Texas at Austin
ashutosh@cerc.utexas.edu sean.shi@mail.utexas.edu dpan@ece.utexas.edu

ABSTRACT

Advanced MOSFETs such as Strained Silicon (SS) devices have emerged as critical enablers to keep Moore's law on track for sub-100nm technologies. Use of Strained Silicon devices provides performance improvement equivalent to use of next generation devices, without actually requiring scaling. Traditionally, the research in the field of SS has been focussed on device modeling and process characterization. Recently (in [1] [2]), the dependence of mobility of a SS MOSFET device on its poly-to-poly distance has been reported. In this work, we propose a new methodology to exploit this dependence to achieve cycle time reduction of a design at the layout level. To the best of our knowledge, this is the first research work to tackle timing closure by layout modifications using active area dependent mobility of SS devices. Our methodology shows consistent improvement for benchmark designs mapped onto various 90*nm commercial standard cell libraries. This work enables reduction of cycle time by as much as 6.31% (and on an average 5.25%) very late in the design closure cycle without requiring any optimization iterations.*

1. Introduction

Last four decades have witnessed tremendous IC performance and cost improvement every couple of years. The most important enabler of such success of semiconductor industry is the continuous device scaling which delivers faster and smaller transistors each generation. State-of-the-art technologies have gate lengths in tens of nano-meter long. Since the realm of sub-100nm technology node, scaling has become extremely costly and technologically challenging. Examples of such challenges include: leakage power, thermal packaging, yield levels, interconnect delays, and printability issues. The conflicting need of performance improvement and challenges of physical scaling requires exploration of techniques which improve performance without scaling. Strained silicon is one such technology.

Advanced MOSFETs rely heavily on the use of Strained Silicon (SS) process to impart performance boost to existing devices (nMOS and pMOS) [3]. SS works by imparting mechanical stress to the channel of a device which significantly boosts the mobility of carriers. In general, more is the stress in the channel (tensile or compressive depending on type of the MOS), better is the improvement in performance. Recently, [2] reported fabrication of pMOS with 200% improved mobility using SS. This phenomenal performance improvement is equivalent to that provided by two generations of technology scaling.

One of the most formidable challenges of modern IC design is to achieve timing closure. Due to rising interconnect delay and timing model inaccuracies at pre-routing stages, there are situations when the design does not meeting timing requirements post-routing. Such a design can be fixed by either re-synthesis, or improving placement or routing of the design, but all these steps require long turn-around-time. A reasonable assumption is that the design team has put the required optimization efforts during all phases so that at the post routing level, the timing *surprise* (observed vs predicted) is sufficiently small[1]. For such a practical case, there is critical need of late-mode optimization methodology which can handle ECO optimizations without requiring resynthesis, placement and routing.

In this work, we propose a completely new methodology to exploit a peculiar property of SS devices reported recently [1]: increasing the distance between adjacent poly of a SS device (also known as poly-to-poly distance or L_{pp}) improves its transconductance thus making it faster. This phenomenon occurs due to improved propagation of mechanical stress due to larger stressors between adjacent poly gates. We propose a new L_{pp} stretching aware cell delay model and demonstrate its linear nature for current and future generation device's L_{pp} range. Based on this model, our technique transforms the cycle time reduction problem into a Linear Program (LP) formulation. This LP when solved, determines the amount by which L_{pp} of a particular cell should be increased. A fast minimally modifying legalization algorithm is also presented to remove any overlap created due to cell stretching. Our methodology entails no extra fabrication/manufacturing cost since current processes already use Strained Silicon technology: IBM's PowerPC5, Intel Pentium-IV and AMD's Opteron and Athlon all have used SS technology [4]. Further, our methodology is particularly suited for late-mode ECO optimization since it directly works on the layout of the design.

The rest of the paper is organized as follows: Necessary background and existing works appear in Section 2. Section 3 describes the derivation of our cell geometry aware L_{pp} dependent timing model. We explain our cell L_{pp} stretching based timing closure methodology in Section 4 . Experimental setup and results are detailed in Section 5. We conclude our paper in Section 6 after suggesting some future research directions.

2. Background & Previous Work

2.1 Device related description

Though there are many ways of producing a SS device, we will focus on S/D transistor variant [1]. Such a device can be fabricated by etching out the the source and drain regions of a conventional transistor. The etched out regions are then epitaxially filled with $Si_{1-x}Ge_x$ (SiGe). SiGe, which now exists under drain and source region, stretches the lattice of Silicon crystal under the channel region. This leaves lots of empty area for electrons and holes to rush through the channel region making the device faster.

It has been known for a long time that the mobility (and thus the delay) of a device in SS technology is dependent on the amount of

[1]Large timing *surprises* means the tools used in synthesis, placement, routing had bad models

978-3-9810801-3-1/DATE08 © 2008 EDAA

stress in the channel. Recent works [1] [3] have reported that the stress in the channel of a SS device can be modulated by changing the size of active area around the channel. In particular, [1] plotted the transconductance improvement for a nested[2] device for very long (1000nm) and short channel (45nm) transistors and showed significant transconductance improvement as a function of L_{pp}. Since the modern devices have gate length around 50nm or so, we plot delay improvement (based on transconductance improvement) of short channel transistor from [1] in Figure 1. We refer users to the cited work for details of experiments carried out.

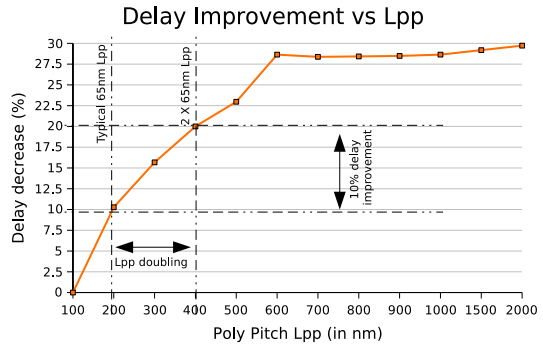

Figure 1: Delay improvement of device as a function of L_{pp}. Delay improves by 10% when L_{pp} is doubled from its nominal (for 65nm DRC) value of 190nm to 400nm.

The DRC for 65nm requires L_{pp} to be around 190nm. It can be seen from Figure 1 that increasing L_{pp} from 190nm to 400nm (around 2X), gives 10% performance increase. The effect saturates when L_{pp} reaches around 600nm. The top figure in Figure 2 shows the view of a nested transistor. Each of the vertical bars are gates, and the horizontal bar is the active area. On increasing the distance L_{pp} the delay through the gates can be reduced.

2.2 Timing analysis and optimization

The speed of operation of a chip is determined by the speed of the slowest path in it. A path can be considered as a sequence of gates and interconnects which starts either from the primary inputs of the circuit, or the output of a flip-flop cell (or latch), and ends either on primary output or the input of a flip-flop cell. The information whether the design meets the timing or not is typically provided by timing analyzer. A design whose performance does not meet the timing requirement needs to be fixed. There are a lot of techniques available to the designer to improve design timing. At the logic synthesis level some of these techniques are cell sizing, logic decomposition and cell duplication. At the placement stage, techniques such as timing driven P&R, buffer insertion and threshold voltage assignment can be used. At the routing stage, techniques such as net ordering, wire-sizing and rip-up can be used to prioritize critical paths.

There has been plethora of work for timing optimization of a design at various levels of abstraction such as architectural, logic level and physical design level. Also, there has been a lot of research in the area of SS devices, mostly about their characteristics and related process technology. However, to the best of our knowledge, there has been no previous work to use the active area dependent mobility of a SS standard cell to achieve timing closure on a given layout which has already been optimized for timing using the techniques already available to designers. Interested readers may refer to [1] [5] [3] [2] for the analysis and properties of SS devices and specifically [1] for the impact of layout of cell on mobility increase. The comprehensive description of making of a SS device can be found in [6].

We would like to point out that [1] shows delay improvement of PMOS gates, and in this work we assume that only the PMOS becomes faster. Since the delay of a cell is average of fall and rise time, we report the reduction in cycle time throughout this paper after scaling it by a factor of 0.5 to compensate for only PMOS improvement. In case similar effects exist for NMOS devices, our cycle time reduction results should be doubled for fair comparison.

[2]According to the terminology in [1], nested device is a device having more than 1 poly gates - for example NAND, XOR

3. Stretching aware Timing Model

3.1 L_{pp} Increase at Standard Cell Level

As discussed in Section 2.1, doubling the L_{pp} (poly-to-poly distance) of a transistor can decrease its delay by 10%. Since in our methodology we stretch standard cells, in this section we will derive the effective L_{pp} increase when a typical standard cell is stretched. In particular, our derivation shows that performance benefit is much more than 10% for stretching standard cells due to their geometry.

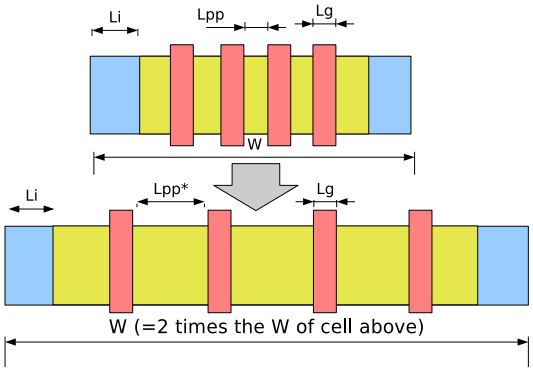

Figure 2: Illustration showing how increasing (doubling in this case) the width of standard cell affects the gate lengths (Lg), poly-to-poly distance (L_{pp}) and insulation oxide thickness (Li).

Modern standard cells typically surround active area with insulating oxide. Thus, the layout to the cell may look like the top layout in Figure 2. The lower half depicts the layout of the stretched cell when the *total* standard cell size has been doubled. It is visually evident that the increase in L_{pp} is not 2X, but more than 2X when the total width of the standard cell increases by 2X. Let us assume that the gate length is L_g (which is *not* expanded), and the oxide isolation thickness on each side of the active area is L_i. The total width of the standard cell is W in the top figure and $2 \times$ W in after stretching. We consider the case where there are n poly in the cell.

The original poly-to-poly distance L_{pp} is given as

$$L_{pp} = (W - n * L_g - 2 * L_i)/n \quad (1)$$

After increasing the width of the cell by N times L^*_{pp} is given as

$$L^*_{pp} = (N * W - n * L_g - 2 * L_i)/n \quad (2)$$

Thus, the ratio of L^*_{pp} to L_{pp} is given as

$$L^*_{pp}/L_{pp} = N + (n * L_g + 2 * L_i)/(W - n * L_g - 2 * L_i) \quad (3)$$

Thus, increasing the width of a cell by N times increases its poly-to-poly distance by more than N times. For each standard cell

in our library, we stored the ratio L^*_{pp}/L_{pp} and each time a cell was expanded by a amount d, the effective poly-to-poly stretching was computed by multiplying by the ratio L^*_{pp} / L_{pp}. For a typical case where L_g and $n * L_g$ can each be 30% of W, we can achieve 3.5X increase in poly-to-poly length for just 2X increase in standard cell width.

3.2 Timing Model

Using Figure 1, we observe that, to the first order, delay improvement of cells is linear function of L_{pp} in the range of interest (200-500nm). Thus, our timing model is: increasing the L_{pp} of a cell by A times decreases its delay by B% as long as A is not large ($\geq$3X). The delay of the device when its width if W (and original width = W_o) can then be represented as

$$D(W) = D(W_o)(1 + K(W_o - W)) \quad (4)$$

The parameter K can lump the value of L^*_{pp} / L_{pp} inside it (see Equation 3), translating the change in width of the cell to the change in width of poly-to-poly distance. Solving for the boundary condition ($W = A \times W_o$ implies $D(W) = D(W_o) * (1 - B)$)we get

$$K = -B/(A \times W_o) \quad (5)$$

Thus, the parameter K is a function of only the nominal width (without any stretching being applied) of each cell. This representation is very useful because during timing optimization, we always know the original width of each cell and thus the value of K can be easily computed.

4. Timing Closure by Cell Stretching

Using our cell stretching based timing closure flow at post-layout stage of design has a few major benefits:

- L_{pp} stretching of a cell has minuscule impact on the timing of its fanin cell. This is because by keeping the L_g of the cell same, the capacitive load seen by the fanin cell remains unchanged. This is in stark contrast to techniques such as gate-sizing which produces a ripple effect on the timing of the fanin gates invalidating any previously done timing analysis.
- Increasing the L_{pp} introduces negligible power consumption and does not use extra routing resources as opposed to techniques such as buffer insertion which add significant power consumption and use extra wires.

In the next subsections, we describe our optimization flow and the formulation for cycle time reduction along with a set of constraints we force to avoid generation of new critical paths.

4.1 L_{pp} Expansion based Optimization Flow

Figure 3 depicts our overall flow. Given a benchmark, we map the RTL into two commercial 90nm standard cell libraries. Since the expansion based timing optimization is critically dependent on the amount and distribution of whitespace in the initial layout, we generated floorplan with many different row utilization (ranging from 20% to 85% with steps of 5%). All the resultant floorplans were placed using Cadence *QPlace*.

An important factor to take into account is the impact of routing congestion on timing and how the improvement using expansion based timing optimization performs under it. To measure this effect, we simulated different routing congestion on the layout by forcing the router to route the design in increasingly lower number of metal layers (maximum=7 metal layers, down to minimum=4 metal layers in step of 1). Although this is an indirect way to model routing congestion, however since the reduction in allowed routing

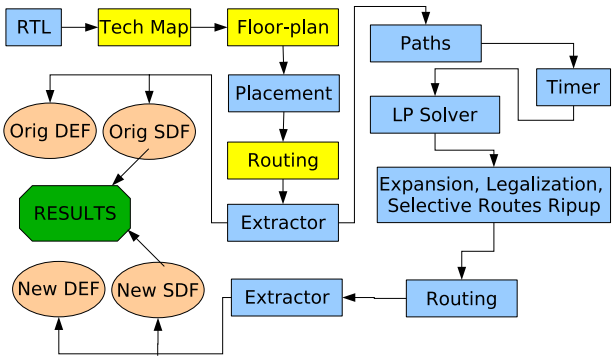

Figure 3: Flow used in our experiments.

layers can be thought of as reducing track capacity per grid cell (summed over all routable layers), we conjecture that this is a good way to model the impact of routing congestion on timing.

The routed design was extracted for the SDF (containing timing information) and DEF (containing physical information) files which are fed into our C++ code to generate the failing paths. Any path whose delay was more than 90% of the observed worst path delay was considered failing. An external LP solver *glpsol* [7] was used to calculate the expansion each cell should undergo. The output was read back into our code, which legalizes the design and rips up all the nets which were incident on to the cells which were moved during legalization. The resulting netlist is fed into the Cadence *Wroute* router in ECO mode to route the ripped-up nets and the timing information is re-generated from the resultant layout. The improvement is then accessed by comparing the timing from original SDF and the SDF generated after our flow.

4.2 LP-based Cell Stretching Algorithm

Any timing analyzer can be run on the routed design to extract failing paths. Cells lying on these paths are critical [3]and possible candidates for expansion. However, how much each of these cells should individually expand to, needs to be calculated. We formulate this problem as a cycle time minimization linear program (LP). All the critical cells compete concurrently for the white-space in the design to stretch themselves and the LP guides these cells to the optimal combination of stretching to get best (smallest) cycle time.

Increasing the width of a critical cell introduces overlap with non-critical cell adjacent to it and needs to be legalized. If this overlap is not controlled properly, the legalization step would cause substantial change in the location of non-critical cells. This increases the chances of turning a non-critical path into a critical path and can lead to need for several iterations of timing improvement over and over again. To control this effect, we propose the following constraints which our optimization flow must adhere to, while deciding about the extent of stretching of the critical cells. Due to stretching of critical cells,

1. No critical cell can move (thus, can only stretch),
2. No non-critical cell should require to jump over critical cell,
3. No cell (critical or not) should leave its row, and
4. A critical cell can stretch only until a particular limit.

[3]A cell lying on a failing path or having negative slack (in path and block based timing analysis respectively), is called as a *critical* cell hereon. A cell which is not critical is referred to as non-critical cell

The above rules are graphically represented in terms of valid and invalid movement in Figure 4. The original layout for the first three cases is given in Figure 4-A.

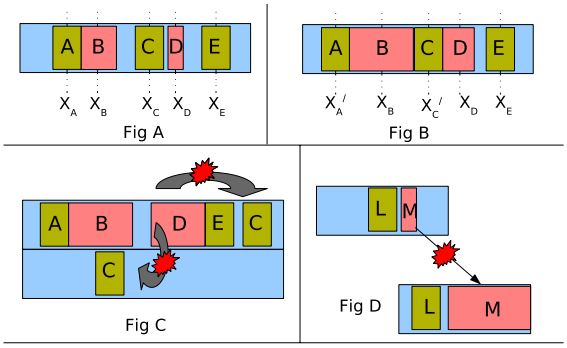

Figure 4: Graphical representation of rules which determine the stretching of critical cells. Color code: Red=Critical cells, Yellow=Non-critical cells, Blue=Standard Cell Row

Point 1) enforces that the interconnect delay between any pair of critical cells remains unchanged after expansion. In Figure 4-B, the coordinates of cell A and C have changed, but that of critical cells B and D have not (even though these cells stretched).

Point 2) prevents "wide" movement of a non-critical cell which circumvents the potential problem of large increase in interconnect delay of nets incident on non-critical cell. The intra-row movement in Figure 4-C represents this invalid move.

Point 3) is like Point 2) with an added advantage that the modifications done in a row cannot induce detrimental changes to neighboring rows. The inter-row movement in Figure 4-C shows an example of invalid move.

Point 4) is a trade-off between the extent of layout modifications and cycle time improvement in light of the saturation of improvement for large L_{pp} values (see Section 2.1). Figure 4-D is invalid if the cell M is allowed to stretch up to (say) 2X original size.

In our experiments we enforce that no cell can be stretched to more than twice its original size. This decision was based on two factors: a) To avoid stretching beyond the limit after which improvements saturates (see Figure 1) and b) Constraining the extent of modification to existing layout so that non-critical cells do not move very far from their original locations. This constraint can be relaxed (to say 3X), resulting in better timing improvement, thus *our results are on the conservative side*.

Consider a timing failing path p in the circuit. This path (as any other path) consists of a sequence of cells and interconnects. Since the path is failing, all the cells will be critical (and thus, under purview of being stretched). Let the set $CELLS_i = C_i^0, C_i^1 .. C_i^l$ be the cells in the path i. Since the critical cells never move from their original location (as per Rule 1 above), the interconnect delays between these cells can be summed up and taken as fixed before and after stretching. Let d_i^I represents the total interconnect delay for path i. Further let the delay of cell C be D_C (pin-to-pin delay). Thus, the total delay of the path i (= $DELAY_i$) can be written as:

$$DELAY_i = d_i^I + \sum_{j \in CELLS_i} D_j \quad (6)$$

Recall that the delay of cell C can be written according to Equation 4. Let the quantity W_o - W in Equation 4 be denoted as ΔW_C for cell C. Thus, Equation 6 can be re-written as Equation 7 where $D_j(W_o)$ is the nominal delay of cell j and K_j is the constant K of Equation 4 for cell j.

$$DELAY_i = d_i^I + \sum_{j \in CELLS_i} D_j(W_o) * (1 + K_j \Delta W_j) \quad (7)$$

Let the worst (longest) delay in the design be D_{max}. Further, let P_{crit} denote all failing paths. Let the whitespace available between two consecutive critical cells a and b (excluding the space used up by non-critical cells between them) be WS_{ab} and each such consecutive pairs be part of the set $PAIRS$. The set of all critical cells is denoted as $CRITCELLS$. Our linear programming formulation can thus be written as follows:

$$\begin{array}{lrcl} \text{Maximise} & M & & \\ \text{subject to} & DELAY_i & \leq & D_{max} - M \ \ (i \in P_{crit}) \\ & \Delta W_a + \Delta W_b & \leq & WS_{ab} \ \ (ab \in PAIRS) \\ & \Delta W_x & \leq & W_x \ \ (x \in CRITCELLS) \end{array}$$

The first set (w.r.t. subscript i) of equations are to minimize the cycle time of the design through the proxy (dummy) variable M. The second set of equations force a solution in which there is always enough space between two critical cells so that cells lying between them can be locally moved without overlap, thus never violating Rule 2 and Rule 3. Third set of equation is basically to prevent any cell to become more than twice its original size: in adherence to Rule 4. The objective function, when maximized, is equivalent to choosing the right values of amount of expansion for each cell (basically ΔW_x for each cell x which is critical) so that the circuit is fastest possible.

The original layout can be seen in Figure 4-A which, when expanded using the Linear Programming formulation leads to the upper layout in Figure 5. Note that the cell B did not grow all the way to consume the remaining white space on its right because of the second set of constraints in the LP formulation which dictate that after stretching, there should be at least the space to put the cell C which is non-critical between cell B and D which are critical.

4.3 Minimum Perturbation Legalization

In general, after the LP has been solved, there will be overlap between critical cells which have expanded and neighboring non-critical cells. This overlap needs to be removed through legalization. In view of the general philosophy of perturbing the minimum amount of interconnects and cells, we propose the following algorithm for two pass, row-by-row legalization which shifts the least amount of cells to get a legalized placement.

Algorithm 1 LegalizeDesign (Argument $DESIGN$)

```
LegalizeInDirection(LEFT)
LegalizeInDirection(RIGHT)
```

The above algorithm makes two calls to LegalizeIndirection (described in Algorithm 2), once to legalize from from left direction and the second time to legalize from the right direction.

Here we describe Algorithm 2 for the case where legalization is performed from left direction towards right direction. Legalization from right to left direction works similarly. For each critical cell I (initialized to start of the row in Line 9), we find the overlap of this cell with the cell immediately on its right side C (Line 12). This overlap is reduced by the amount of whitespace between the cell I and cell C (Line 14). The cell C is then shifted by the value of overlap if it is positive (Line 15). The cell I is now set as cell C for and the while loop of Line 13 is executed again with reduced value of overlap. At some stage, the variable overlap will be non-positive and that terminates the inner while loop. After each termination of inner while loop, cells from the position of I to that of J are

Algorithm 2 LegalizeInDirection (Argument $DIRECTION$)

1: // D_{XY} gives Distance between cell X and Y
2: // WS_{XY} gives original whitespace between cell X and Y
3: // W_X gives stretched width of cell X
4: // NXTCRT(X) gives *critical* cell after (to $DIRECTION$) X
5: // NEXT(X) gives the cell after (to $DIRECTION$) X
6: **for all** Row R in the design **do**
7: Insert dummy critical cell START at the start of row R
8: Insert dummy critical cell END at the end of row R
9: Cell $I \Leftarrow$ START , Cell $J \Leftarrow NXTCRT(I)$
10: **while** Cell $I \neq$ END **do**
11: Cell $C \Leftarrow NEXT(I)$
12: $overlap \Leftarrow D_{IC} - 0.5 * (W_I + W_C)$
13: **while** $overlap \geq 0$ **do**
14: $overlap \Leftarrow overlap - WS_{IC}$
15: Shift cell C to right by $overlap$
16: $I \Leftarrow C$, $C \Leftarrow$ NEXT(C)
Ensure: $C \neq J$ OR $overlap = 0$ {Checks rule 2 and 3}
17: **end while**
18: $I \Leftarrow J$, $J \Leftarrow$ NXTCRT(J)
19: **end while**
20: **end for**

legalized. The outer loop, then moves I to the next critical cell and at the end of the outer loop the algorithm completes (Line 13).

An example of how the legalization works is shown in Figure 5 where layout with overlaps (the rectangles are cells), are legalized. The markers $X_A..X_E$ denote the center of corresponding cell.

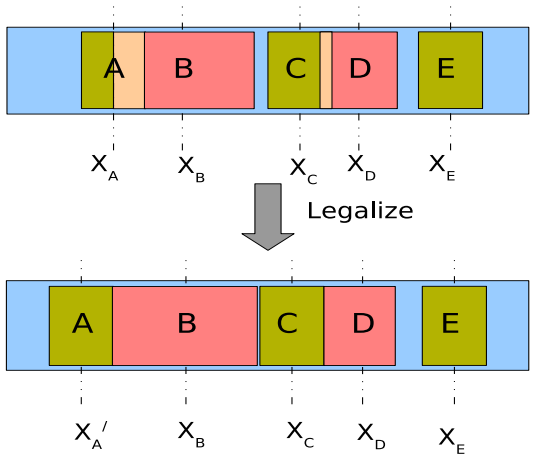

Figure 5: Overlap among cells generated by the LP formulation and its removal using Algorithm 1

5. Experimental Results

For the purpose of demonstrating the benefit of our proposed technique, we took the technology independent RTL of design *WIMS* from opencores.org [8]. This RTL was synthesized using high V_T and low V_T variants of a commercial 90nm cell library resulting in two mapped RTLs. The V_T value in the high threshold variant of library is 350mV and 200mV for low threshold variant. Our code was written in C++, which ran on a 64-bit 4-CPU 8-GB Linux machine. For place and route, we used Cadence Silicon Ensemble^{TM} v5.4 running on solaris 5.7, 4GB ram, 2-CPU machine. The vital statistics of our benchmark *WIMS*, which is a wireless integrated microsystem, are in Table 1.

Table 2 tabulates some of the representative observation points. All delay values are presented in adjacent pair of columns for the

Name	WIMS Controller
# Cells High Vt Mapping	11454 GATES
# Interconnects for High Vt Mapping	21209
# Cells Low Vt Mapping	12238 GATES
# Interconnects for Low Vt Mapping	22492
Row Utilization	0.2 - 0.8
# Routing Layers	4 - 7

Table 1: The benchmark used in our design and its details

benchmark design mapped onto low and high Vt libraries. Columns 2 and 3 show the row utilization of the layout and number of layers used for routing of the design. The cycle time of original layout after parasitic extraction appears under heading *T-Original*. Columns under *T-Expanded* show the cycle time after the critical cells are expanded but before any legalization and re-routing is done. The significance of this set of data will be explained in next paragraph. Columns under *T-Rerouted* show the final cycle time after legalization and re-routing. Heading *Improvement* shows the cycle time improvement (=column *T-Rerouted* - column *T-Original*). The set of columns under *Cells Moved* have the number of (non-critical) cells moved due to legalization after expansion of critical cells.

Comparison of entries under Column *T-Expanded* and *T-Rerouted*, shows that the cycle time before and after the legalization and re-routing stages are usually exactly the same which means that legalization and re-routing did not degrade a non-critical path into critical path. When these entries are not the same, they are very close to each other with a maximum difference of 0.01% in row utilization range of 0.4-0.7. This proves that adhering to rules in Section 4.2 successfully maintain very high correlation between cycle time estimation before and after re-routing. Also note that *very* few cells ($\leq$0.7%) were moved during legalization. The total number of nets which are re-routed were always found to be under 0.4% of the total nets.

We observe in Table 2 that our technique is able to reduce the cycle time of the design by nearly 5.25% on average. Achieving this improvement so late in the design cycle without going back to placement or routing stages is remarkable. Further, as discussed in previous paragraph, the timing improvement can be predicted right after the LP formulation is solved with very high accuracy. This feature allows the designer to tweak the maximum limit to which any cell can expand to get even better timing improvement without going through legalization and re-routing phase. We next discuss the impact of various design choices such as V_t values, number of routing layers used, row utilization of a design on the achieved cycle time reduction using our optimization methodology.

Figure 6 shows the cycle time reduction achieved using our technique for our benchmark routed in 4 vs 7 metal layers. Our hypothesis was that a congested (w.r.t. routing) design will offer lesser timing improvement due to possible detours during re-routing of nets ripped due to movement of cell. However, we observed that the timing optimization is pretty much independent of number of routing layers. We believe it is due to the minuscule number of nets that need to be re-routed due to minimal layout modification owing to constraints in Section 4.2. Overall, an average of 5.1% cycle time reduction was achieved over row utilization ratio between 0.4 and 0.7.

Figure 7 shows the variation of timing improvement for the low threshold design and high threshold design. In general, high threshold voltage designs are slightly more amenable to stretched silicon expansion based timing optimization method. The reason for this is rather simple: The timing improvement we achieve comes from the decrease in cell delay and the higher relative proportion of cell

Design	Row	Rout	T-Original(ns)		T-Expanded(ns)		T-Rerouted(ns)		Improvement (%)		Cells Moved	
	Util	Layr	LVT	HVT	LVT	HVT	LVT	HVT	LVT	HVT	LVT	HVT
wims	0.20	4	9.879	40.171	8.727	35.553	8.727	35.553	5.83	5.75	78	88
wims	0.20	7	9.883	40.180	8.731	35.271	8.731	35.271	5.83	6.11	80	89
wims	0.30	4	9.389	32.889	8.352	29.260	8.352	29.260	5.50	5.53	83	79
wims	0.30	7	9.379	32.811	8.349	29.182	8.349	29.182	5.52	5.53	79	80
wims	0.40	4	9.257	31.923	8.089	28.261	8.088	28.264	6.31	5.73	79	74
wims	0.40	7	9.241	31.918	8.074	28.257	8.074	28.260	6.32	5.73	80	78
wims	0.50	4	9.060	32.110	8.068	28.322	8.068	28.322	5.47	5.90	76	69
wims	0.50	7	9.062	31.423	8.071	28.423	8.071	28.423	5.47	5.68	84	69
wims	0.60	4	8.865	31.242	7.969	26.452	7.969	26.455	5.05	4.95	67	78
wims	0.60	7	8.862	29.066	7.969	25.954	7.969	25.958	5.05	5.35	77	73
wims	0.70	4	8.354	31.734	7.584	29.019	7.580	29.019	4.65	4.28	80	69
wims	0.70	7	8.343	31.703	7.567	28.981	7.567	28.981	4.63	4.29	78	78
wims	0.80	4	8.161	30.823	7.558	27.846	7.558	27.842	3.67	4.12	72	84
wims	0.80	7	8.106	29.672	7.511	26.582	7.573	26.582	3.70	5.20	81	76

Table 2: Absolute value and improvement of cycle time achieved for some representative observation points.

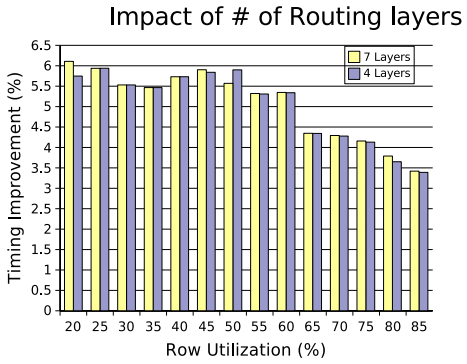

Figure 6: Timing improvement vs Row utilization for routing using 4 and 7 layers. Benchmark - WIMS High Vt Design routed in 7 layers

delays to interconnect delays in a high Vt design allows higher possibility of timing optimization. Overall, an average of 5.2% cycle time reduction was achieved over row utilization ratio between 0.4 and 0.7.

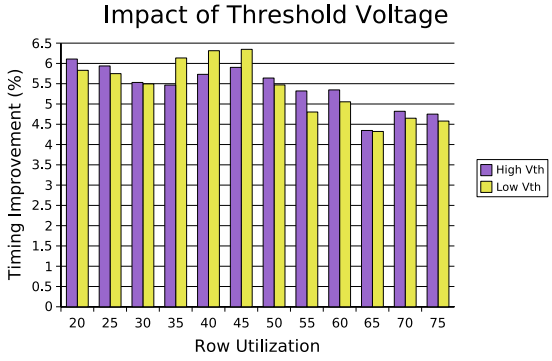

Figure 7: Timing improvement vs row utilization for low and high Vt benchmark WIMS variants

The amount of whitespace directly impacts our technique, because in essence, our technique consumes whitespace to improve cycle time. Looking at Figure 7 and Figure 6, we can observe how the timing improvement varies as row utilization is varied for different Vth values and routing layers used. As expected, lower whitespace in the design (=higher row utilization), leaves lesser room for our flow to improve cycle time. Overall, in the practical working range of row utilization of 0.4-0.7, our technique achieved average cycle time reduction of 5.3%.

6. Conclusions and Future Work

In this work we have analyzed the impact of active area dependent mobility improvement for Strained Silicon (SS) devices by considering the isolating oxide around the standard cell. A conservative L_{pp} aware cell delay model was extracted from this observation. For the first time, a methodology to exploit this property inside the conventional design flow is proposed incorporating the active area sizing based optimization during timing closure. A set of constraints were proposed whose adherence results in impeccable fidelity of timing improvement. A legalization algorithm with minimum perturbation to existing cells was also proposed. On a wide range of design parameters (such as threshold voltages, row utilization, routing congestion), our technique achieves 5.25% reduction in cycle time very late in design closure flow without requiring any iterations. Future works include considering L_{pp} dependent cell delay during detailed placement phase and discretization of L_{pp} expansion so that pre-characterized library cells can be used. Another line of research is to explore concurrent usage of gate sizing and L_{pp} sizing to achieve timing closure under a given power envelop and floorplan constraints. Since strained silicon is going to be pervasively used in future designs, we expect our new design methodology which takes advantage of its layout-dependent properties to prove very useful for timing closure.

7. REFERENCES

[1] Eneman, G. et al., "Scalability of the $Si_{1-x}Ge_x$ Source/Drain Technology for the 45-nm Technology Node and Beyond", *IEEE Transactions on Electron Devices*, Vol 53, No.7, July 2006. pp. 11647-1656

[2] Washington, L. et al., "pMOSFET With 200% Mobility Enhancement Induced by Multiple Stressors" *IEEE Electron Device Letters*, Vol 27, No. 6, June 2006. pp. 511-513.

[3] Bianchi, R. A. et al., "Accurate Modeling of Trench Isolation Induced Mechanical Stress Effects on MOSFET Electrical Performance", *International Electron Devices Meeting* 2002, pp. 117- 120

[4] Corporate Websites for AMD®, Intel® and IBM®

[5] Gallon, C. et al., "Electrical Analysis of External Mechanical Stress Effects in Short Channel MOSFETs on (0 0 1) Silicon", *Solid State Electronics*, Volume 48, Issue 4. http://dx.doi.org/10.1016/j.sse.2003.09.024 pp. 561-566

[6] Thompson, E. S. et al., "A 90-nm Logic Technology Featuring Strained Silicon", *IEEE Transaction on Electron Devices*, Vol 51, No 11, Nov 2004, pp. 1790-1797

[7] GNU Linear Programming Kit, Available at http://www.gnu.org/software/glpk/

[8] http://www.opencores.org