Design-Synthesis Co-Optimisation Using Skewed and Tapered Gates

Ayan Datta[1], James D. Warnock[2], Ankur Shukla[1], Saurabh Gupta[1], Yiu. H. Chan [2], Karthik Mohan[1], and Charudhattan Nagarajan[1]

[1]India Systems Development Lab, IBM India, Bangalore
[2]STG HW Development Lab, IBM USA, NY USA

***Abstract*—This paper presents a novel technique to optimize the design of non-conventional tapered and skewed standard cell gates, and the synthesis algorithms for efficient usage of such gates in IBMs high-performance 22nm CMOS SOI technology. The focus is on design considerations to ensure that synthesis can use these gates efficiently, leveraging the resulting timing improvements for faster timing closure of high-performance microprocessor designs. A detailed analysis is presented, where by exposing these gates to synthesis at different points in the process, the optimal point of insertion is identified. Also an efficient algorithm is proposed to handle decisions regarding the conversion of conventional gates to non-conventional gates, after taking into account multiple factors including delay and slew. Results show 25 - 30% improvement in total negative slack of designs and 20 -25% reduction in the total number of negative paths, without any major impact on total power of the designs.**

Keyword - High Performance design, skewed gates, tapered gates, efficient timing closure, timing improvement, standard cell design, optimising synthesis.

I. Introduction

Timing closure of high-performance (HP) designs is becoming more challenging as technology scales into the deep submicron regime.Node-to-node transistor speed improvements were typically in the 40% range [1],much larger than recent generational performance improvements, due to difficulties and non-idealities associated with scaling. Another dimension was added to the problem as scaling enabled logic designers to add complex instructions and more functionality. In order to avoid increasing the time to market, larger design blocks were created, so that designs could be closed more quickly and with less design effort [2]. New synthesis techniques have been invented to handle the increasing complexity of HP designs, but total turnaround time constraints may often limit the ability to locally optimize all the critical timing paths in the design. One of the approaches developed to allow further tuning of critical paths, is the use of non-conventional logic gates, which we will refer to as tapered and skewed gates, as explained later in this paper.

Tapered and skewed gates (TSGs) refer to library cells built with specific device sizing paradigms, aimed at boosting circuit speed on critical timing paths, in HP microprocessor designs. Skewed gates are designed by strengthening the pull-up or pull-down network in order to speed up rising or falling transitions respectively. On the other hand, tapered gates are designed by varying the width of the devices in a stacked configuration, reducing the delay for a specific critical input [3]. This paper describes a novel approach to timing closure, using tapered and skewed gates, and a concurrent exploration of standard cell design and synthesis algorithms. To the best of our knowledge this is the first work to report on such a design/synthesis co-optimization for efficient timing closure of HP microprocessor designs.

II. Optimal Design of TSGs

For efficient closure of timing critical paths, the transistor level design of the tapered and skewed gates is extremely important. These should be designed such that the impact on power and area is as small as possible, and with consideration for ease of use by the synthesis tools. The following sub-section describes the design of these gates and the synthesis considerations.

A. Tapered Gates Design Considerations

Tapered gates are designed by varying the sizes of the devices in a stack (for example, the nFETs in a NAND2) in order to improve the timing for a critical input by increasing overall drive strength, but at a constant input capacitance for the critical input as shown in figure 1(b). In this way, the pull-down delay from A-Y, is sped up, with approximately the same input capacitance on the A pin, but with a penalty of increased capacitance on the B input, as shown in figure 1(c) and 1(d).

While designing these TSGs several design considerations needs to be taken into account. The first consideration is that we would like for the input capacitance on the critical input pin to stay about constant. This makes it easy for the synthesis engine to figure out whether a tapered gate will improve overall timing slack, or not. If tapered and non-tapered devices did not come in pairs with matching input pin capacitance, for every attempted tapered gate swap, the other gates on the critical path would likely have to re-optimized for delay minimization. The set of tapered gates was thus designed with total n+p gate width matching that of the non-tapered design. However, the larger gates, with more than two fingers, tended to see an increase in input capacitance, due to the fact that the wire needed to strap all gates together increased in length compared to the non-tapered version, due to the extra device fingers needed on the non-critical inputs.

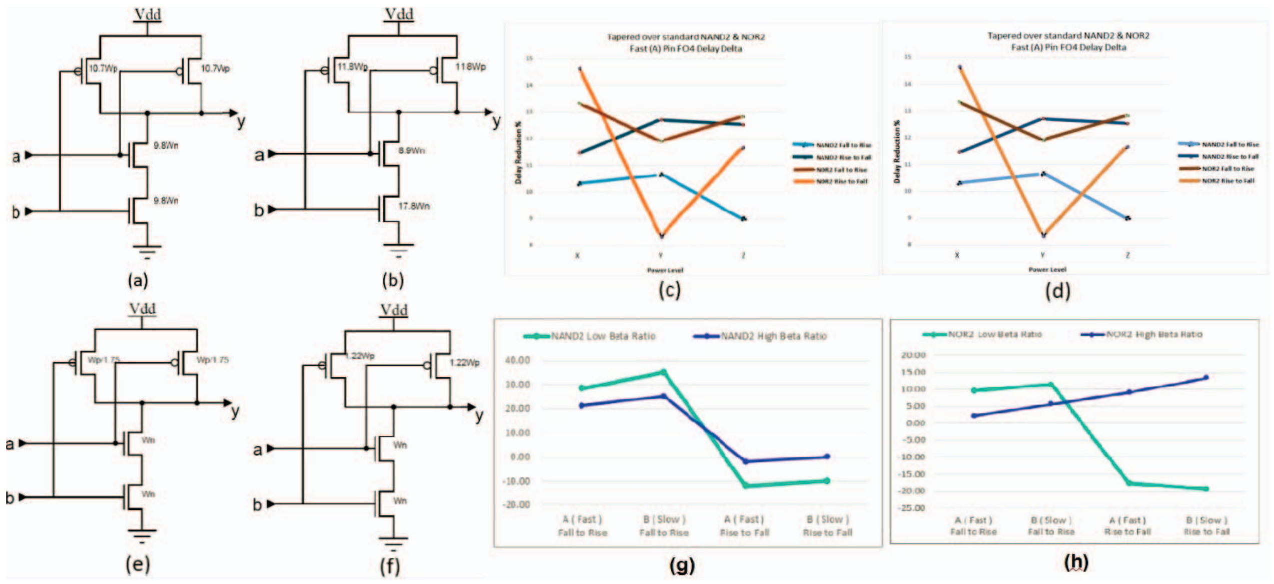

Fig. 1: (a) Schematic of conventional Nand2 (b) Schematic of Tapered nand2 (c) %delay improvement in A pin of tapered Nand2 and Nor2 over conventional gate (d) %delay increase in B pin of tapered Nand2 and Nor2 over conventional gate (d) Schematic of conventional Nand2 (e) Schematic of skewed Nand2 (g) %delay delta of Nand2 over conventional gate (h) %delay delta of Nor2 over conventional gate

A second consideration was that it was considered desirable to divide the performance benefit for rising (R) versus falling (F) transitions as equally as possible. For example, when swapping in a tapered NAND gate for its non-tapered twin, tapering of the n-stack would tend to speed up the falling output transition considerably, while slowing down the rising transition. So, if both R and F transitions had about the same timing criticality, and especially if R transition was a little more critical, a tapered gate designed strictly in the fashion of figure 1 would be much more difficult to use. The synthesis engine would have to do multiple swaps, to remove the resulting R/F timing skew. With this goal in mind, for the tapered NAND gate example, the n-stack was shrunk somewhat (by about 10%), and the p device width was increased (keeping the sum n+p width the same to meet the first criterion, above). Careful choice of design point then resulted in roughly equivalent speedups for both R and F transitions, as seen in figure 1(b).

B. *Skewed Gates Design Considerations*

Skewed gates are designed by varying the size of all the devices in the pull-up or pull-down network as shown in figure 1(e,f). The original (non-skewed) gates are designed such that the ratio of R to F delays is relatively consistent across all gate types, matching that of the inverters which are designed to use the maximum available area in the image for both pFET and nFET.The skewed-beta cells were designed with a beta ratio determined by comparison to a NAND2 and NOR2 gate with both pFET and nFET sizes matching that of the inverter. So, for example, the high beta ratio cells (strong pFET) used a NAND2 as a reference,as shown in figure 1(f), where the pFET width had been increased from that of the regular (non-skewed) NAND2 until it matched that of an inverter with the same nFET width. This design point makes maximum use of the available cell area for both nFET and pFET (ie similar to the non-skewed inverter), but of course is skewed heavily towards pFET strength. This set of skewed gates was found to be useful for the three types of scenarios described in the following paragraphs.

Firstly skewed logic gates could be useful, was in AND-OR kinds of situations, where multiple levels of back-to-back NAND gates might typically be used. The high-beta NAND gates tended to be faster in chains of NAND logic, due to their more efficient use of the available cell area.So, even though such skewed gates were not very effective in general logic , they could be used effectively in certain types of circumstances.Finally, due to variable loading, the granular nature of the library, and various other factors in a typical design, R vs F timing slacks were usually a little mismatched in the design, even after optimization with the regular non-skewed gates. The strategic introduction of a skewed gate in just the right spot could then neutralize the mismatch, slowing down the faster edge, but simultaneously speeding up the slower (limiting) edge.

III. METHODOLOGY FOR OPTIMAL SYNTHESIS WITH SKEWED AND TAPERED GATES

An efficient methodology was needed in order to fully leverage the timing credits coming from these special gates. This section will describe the design considerations to identify the right insertion point and a good algorithm for efficient usage of these gates by synthesis.

```
Begin Proc SWAP_Tapered
  For each gate {
    if (Funtionality of gate=Nand2) {
      If (Y_Slack < 0) {
      Get the Vt and size of the gate
      Create a tapered gate of same Vt ,size
       Place the tapered gate  and legalize
       Delete the original gate
       Compute $\delta_{Y\_improvement}$ , $\delta_{B\_degradation}$ based on Eqn 1,2
         if ($\delta_{Y\_improvement} > 0$ ){
           if ((Tapered$_{B\_Slack} > 0$ || ($\delta_{Y\_improvement} > \delta_{B\_degradation}$ ) {
               continue}
               else {Substitute the original gate}
               }
           }
       }
   Check for Nor2, AOI21, OAI21, Nand3, Nor3.
   }
End Proc Swap_Tapered
```

Fig. 2: Algorithm to swap tapered gate in place of conv gate

```
Begin Proc SWAP_Skewed
  For each gate {
    if(gate=Inverter) {
      Compute $\delta_{Y\_RF}$ based on Eqn 3.
        if ($\delta_{Y\_RF}$ > Threshold) {
            Replace with the weak pull-up skewed gate
            if (New Y_slack > Old Y_slack) {
            continue
            } else { Revert back to the original gate}
      } elseif ($\delta_{Y\_RF}$< -Threshold) {
            Replace with the weak pull-down skewed gate
            if (New Y_slack > Old Y_slack) {
            continue
            } else { Revert back to the original gate}
        }
    }
  Check for Nand2, Nor2, NAND3, NOR3
}
End Proc Swap_Tapered
```

Fig. 3: Algorithm to swap skewed gate in place of conv gate

A. Algorithm

Typical designs will have timing paths which can be improved by swapping in skewed gates, i.e. wherever uneven R and F timing slacks are observed. Also timing paths may potentially be improved by swapping in tapered gates, wherever there exists a slack difference between the different pins of a gate containing stacked devices. So an optimal algorithm should classify these paths efficiently, swapping in skewed or tapered gates based on the path.

Such an algorithm should swap in the new gate if it improves the timing of the path, but should revert back to the conventional (conv) gate, if the timing deteriorates. These timing requirements can be obtained from the following equations,

$$\delta_{Y_improvement} = Y\_Slack_{tapered} - Y\_Slack_{conv} \quad (1)$$

where $\delta_{Y\_improvement}$ is the difference in the output slack *(Y_slack)* with the tapered verses the conventional gates.

$$\delta_{B_Degradation} = B\_Slack_{conv} - B\_Slack_{tapered} \quad (2)$$

Whereas $\delta_{B\_degradation}$ is the difference in the input slack at non critical input pin (*B_slack*) of a tapered gate and slack at the same pin with the conventional gate.

The procedure SWAP_Taper in Figure 2, is a sketch of the algorithms called iteratively on all the gates in a design. The algorithm initiates by selecting a gate having negative output slack and replacing it with a tapered gate, retaining the connectivity, size and Vt type. This swap will be retained if $\delta_{Y\_improvement}$ is greater than zero. If the output slack with the tapered gate degrades, or if the amount of slack degradation at the non-critical input exceeds over the slack improvement in the critical input, i.e. ($\delta_{Y\_improvement}$ - $\delta_{B\_degradation}$) is negative, or the slack at non-critical input after swapping is negative, then the original gate is swapped back in. This algorithm will repeat for all tapered logic gates types like Nand2, Nor2, AOI21, OAI21, Nand3, Nor3.

For Skewed gates, the difference between rise and fall slack at the output pin, plays the major role in deciding whether a stronger pull-up or pull-down is needed. This is represented by the following equation.

$$\delta_{Y_RF} = Y\_Slack_R - Y\_Slack_F \quad (3)$$

The procedure SWAP_Skew in figure 3, initiates by selecting the gates in the design and obtaining the δ_{Y_RF}. If δ_{Y_RF} exceeds a threshold value then the gate will be replaced with the skewed gate having weak pull-up, and when the δ_{Y_RF} falls below another (negative) threshold, then the gate will be replaced with the skewed gate having weak pull-down. The original gate is swapped back in if the worst slack among rise and fall at output degrades. The threshold value is based on the technology rules of these gates.

B. Identifying the Optimal insertion point in Synthesis

To get maximum timing benefit from these non-conventional gates, they should be introduced or swapped in at the right point in synthesis process. Since tapered gates speed up critical paths, it may be a good option to introduce them early in the synthesis cycle, so that these gates get precedence over other optimization techniques. But the larger area foot print of these gates, as compared to their conventional counterparts, may lead to an unacceptable increase in area and power. Early introduction does help synthesis to react to the undesirable impacts of using tapered gates, for example to fix the timing degradation on the other pins of the gates. But in order to reduce the impact on area and power, it may be more ideal to introduce tapered gates during synthesis at a later stage. However if these gates are introduced too late, then synthesis may not be able to fix the perturbations introduced in the design.

Skewed gates on the other hand are much easier to add in synthesis as they are either footprint-compatible, or close to footprint-compatible with conventional gates, making it easier for existing algorithms to handle these swaps. However, it should be noted that caution is still required when these gates are added very late in synthesis, because of the impact on the timing slack of the other transition edge.

Design	# Latches	With Conventional gates			With Tapered gates only			With skewed gates only			With both tapered and skewed gates		
		TNS	Area	Power (mW)	TNS	Area	Power (mW)	TNS	Area	Power (mW)	TNS	Area	Power (mW)
IFU_1	4401	-2930	448315	77.87	-2284	452240	78.11	-2702	448317	77.54	-1913	451577	77.72
IFU_2	1359	-6577	142080	69.77	-5597	147329	70.3	-6018	142086	69.18	-5216	147336	69.18
IFU_3	5810	-1995	457251	109.26	-1823	460834	109.44	-1758	457255	108.64	-1526	460840	108.8
IDU_1	2486	-2677	222891	15.81	-1703	227908	16.03	-2332	222895	15.72	-1382	227911	15.91
IDU_2	3218	-8906	319294	71.53	-7818	324051	71.85	-8074	319297	70.71	-7046	324052	70.98

TABLE I: Comparison of design characteristics with Tapered and Skewed gates with conventional gates

IV. Results

In order to evaluate the proposed method, experiments were carried out on 64 bit Linux 2.6 based grid of four Intel Xeon 2.2 Ghz CPUs . The designs used in this work are from the Instruction Fetch and Instruction Decode units in the core of the IBM $Z13^{TM}$ microprocessor [4] , which uses IBM High performance 22nm SOI technology [5]. These designs runs at a frequency of 5.0 Ghz and timing closure is extremely challenging. The designs were synthesized with timing driven synthesis [6] tool from IBM EDA.

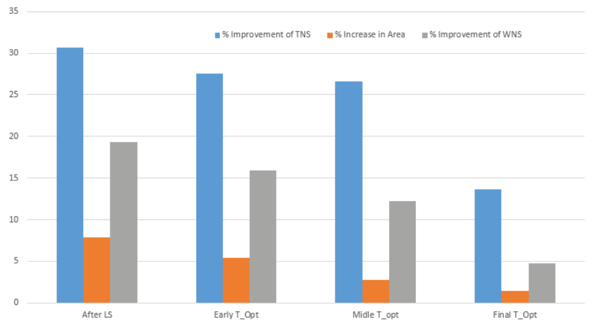

Fig. 4: Design comparison different steps of Synthesis

The first experiment was carried out to identify the optimal insertion point of tapered and skewed gates in synthesis. The TSGs were hidden from synthesis, during initialization, and then later unhidden and inserted after logic synthesis(LS) and different timing optimization (T Opt) steps. It is observed that if TSGs are unhidden very early in synthesis, then the timing improvement, total negative slack (TNS) and worst negative slack (WNS), show the most improvement, but at the cost of higher area. It can be inferred from figure 4 that if TSGs are inserted during the middle timing optimization step, then the timing improvement is still comparable to that from insertion during earlier synthesis steps, but with a lower impact on area.

The second experiment was focused on swapping TSGs into previously synthesised designs,implemented with only conventional gates.Table I presents the comparison of timing, area and power metrics with respect to the original design synthesized without TSGs. The algorithms described above were used individually, and then all together, in order to understand the impact of TSGs on the design.

For all designs, it was observed that TSGs improved the timing characteristics of designs. When only tapered gates were used, the TNS of designs improved by an average of 15%. Total cell area of the design increased by an average of 1.75%, as the footprint of tapered gates is larger. The total power increase was minimal, at an average of 0.4%. On the other hand when only skewed gates were used, the average improvement of TNS is 9-10% more . The total power actually improved slightly over the designs synthesized with only conventional gates. The best results were obtained when both tapered and skewed gates were used, with an average TNS improvement of 26% and maximum improvement of 34% without any impact on the total power of the design. The power reduction obtained from skewed gates nullified the minor power increase from tapered gates, resulting comparable total power with conventional gates.

V. Conclusion

As technology scaling continues, it is clear that power delivery and dissipation is limiting the performance of HP microprocessors design, and that ongoing reductions in supply voltage will be required to manage those issues. However, it will become increasingly difficult to maintain reasonable circuit performance as voltage is lowered. Thus methods for tuning designs for additional performance, while avoiding significant impact on total power dissipation, are needed in order to continue providing higher levels of performance. We have described a novel technique based on co-optimization of standard cell designs and synthesis algorithms in order to efficiently tune synthesized circuits, to squeeze extra performance from any given design. Results show improvements of total negative slacks by 25-30% without any impact on the total power of the design when TSGs are used together.

References

[1] Williams, Thomas W. ”EDA to the Rescue of the Silicon Roadmap.” Multiple Valued Logic, 2008. ISMVL 2008. 38th International Symposium on. IEEE, 2008.

[2] R. Puri and D. Kung, ”The dawn of 22nm era: Design and cad challenges,” in VLSI Design, 2010. VLSID ’10. 23rd International Conference on, jan. 2010, pp. 429-433.

[3] Hwang, W. et al., ”Performance Analysis of Tapered Gate in PD/SOI CMOS Technology”, Proceedings of Technical Papers, 2001 International Symposium on VLSI Technology, Systema and Applications, Apr. 18-20, 2001, pp. 287-290.

[4] J. Warnock et al. 22nm Next-Generation IBM System z Microprocessor. 2015 IEEE International Solid-State Circuits Conference.Jan 2015.

[5] S. Narasimha, et al., 22nm High-Performance SOI Technology Featuring Dual-Embedded Stressors, Epi-Plate High-K Deep-Trench Embedded DRAM and Self-Aligned Via 15LM BEOL, IEDM Dig. Tech. Papers, p. 3.3.1-3.3.4, 2012.

[6] L. Stok, D. S. Kung, D. Brand, A. D. Drumm, A. J. Sullivan, L. N. Reddy, N. Hieter, D. J. Geiger, H. H. Chao, and P. J. Osler, ”BooleDozer: Logic synthesis for ASIC’,s”, IBM J. Res. Develop., vol. 40, no. 4, pp.407 -430 1996