Short Path Padding with Multiple-V_t cells for Wide-Pulsed-Latch Based Circuits at Ultra-Low Voltage

Yongming Ding, Wei Jin, Guanghui He, Weifeng He
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
(dingyongming, hewf) @sjtu.edu.cn

***Abstract*—This paper presents a short path padding technique for wide-pulsed-latch based circuit design in near/sub-threshold (V_t) regime. To reduce the additional hardware cost, multiple-V_t buffer cells are used to pad the short paths to avoid hold time violations. To reduce the runtime of the padding algorithm further, step-by-step based and path group based short path padding schemes are proposed. Employing the integer linear programming (ILP) solver, an automatic short path padding software is developed. Experimental results show that our proposed short path padding technique can reduce 52.3% hardware padding cost on average. Furthermore, the runtime of padding software is reduced 79.6%, 74.95% and 80.88%, by using the step-by-step based, path group based and the hybrid scheme, respectively. In consequence, this technique supports up to a wide pulse of 1/3 cycle time in the pulsed-latch pipelines to enable a large time-borrowing capability and tolerance of variations.**

Keywords—buffer insertion; pulsed-latch; hold time violation; ultra-low voltage; short path padding

I. INTRODUCTION

Pulsed-latch is an attractive option in designing pipelines for near/sub-threshold (V_t) digital circuits because of its lower sequential overhead and higher variation tolerance than previous edge-triggered flip-flop. In recent years, some works [1-5] have focused on the pulsed-latch-based circuit design. However, in these designs, only a small amount of timing can be borrowed since they use very narrow pulses. To enable larger amount of timing to borrow, wide-pulsed-latches based circuit techniques are presented [6, 14].

In current wide-pulsed-latch based circuit design, hold time violation is a severe problem. The hold time violation can be resolved by inserting delay buffers into all short paths of the design [7-12]. However, the additional hardware overhead of the padding is unbearable because there are many of short paths with large timing slack need to be padded in a wide-pulsed latch based circuit.

To remove the hold time violations, N. V. Shenoy et al. [11] presented a greedy heuristic algorithm based on linear programming for buffer insertion in short paths in 1993. This algorithm was proposed to resolve hold time violations in traditional edge-triggered circuits at a normal supply voltage. In 2013, W. P. Tu et al. [12] proposed a method to fix hold time violations under different power modes for ultra-low voltage designs. In 2016, a buffer insertion technique to remove hold violations at multiple process corners is presented by Inhak Han et al. [7] In all these designs, the required hold time is very small.

Different from previous works, our work are focused on short path padding for wide-pulsed-latch based near/sub-threshold circuits. In these circuits, the pulse width is expected to 1/3 cycle time to enable a large time-borrowing capability and tolerance of variations at ultra-low voltage. However, the additional hardware cost for short path padding is very large because the required hold time is equal to the pulse width [6, 14]. Thus it is important to develop effective techniques to pad short paths with lower hardware cost.

In this paper, we proposed an automatic short path padding technique based on the ILP solver while minimizing the area and power overhead of short path padding with multiple-V_t cells. Furthermore, two improved padding scheme based on step-by-step and path group are proposed to reduce the complexity and runtime of short path padding algoarithm. Experimental results show that our proposed short path padding technique can reduce 52.3% hardware padding cost on average. Furthermore, the runtime of padding algorithm is reduced 79.58%, 74.95% and 80.88%, by using the step-by-step based, path group based and the hybrid scheme, respectively.

II. PROPOSED SHORT PATH PADDING FOR WIDE-PULSED-LATCH BASED CIRCUIT

A. Short path padding with multiple-Vt cells

A circuit C can be represented as a directed acyclic graph $G = (V, E)$, where V is a set of pins including every input and output pins of each gate. E, representing a set of edges, is a combination of internal edges E_i and external edges E_e of the C. The internal edge means an edge directed from input pin of a gate to its output pin while external edge portrays a interconnection among the gates of this circuit. For example, the circuit shown in Fig. 1 has 8 pins names as *a-h*, respectively. The internal edges E_i can be illustrated as the edges a→d, b→d, c→e, f→h and g→h, shown as dashed arrows. Meanwhile ,the circuit has two external edges, d→f and e→g, shown as solid arrows in Fig.1 (b). The delay of an edge e is represented as $d(e)$. Obviously, the delay of the E_i is determined by the propagation delay of all gates in the circuit. The delay of an external edge is determined by the number of buffers inserted into this path after short path padding (wire delay is neglected).

Moreover, a timing path p is defined as a logic path between the start point and end point. The start point can be a input port of the design or clock pin of a register. The end point can be a data input pin of a register or an output port of the design. The delay of the path p is represented as $D(p)$ which is determined by a set $E(p)$ ($E(p)$ represents all edges involved in the p).

This work was supported by the National Natural Science Foundation of China under Grant No.61774104.

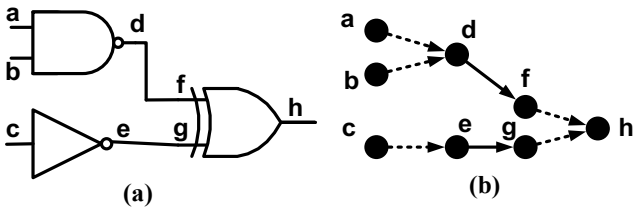

Fig. 1. (a) Schematic of a combinational circuit (b) It's directed acyclic graph

For each path p in the path set P, $D(p)$ has to be no larger than the critical path delay T and no shorter than the required hold time t. In a pulsed-latch based design, the required hold time is the same as the pulse width. As a result, we can conclude a equation and two inequalities as follows.

$$D(p) = \sum_{e \in E(p)} d(e) \tag{1}$$

$$D(p) \le T \tag{2}$$

$$D(p) \ge t \tag{3}$$

If a hold time violation appears in E_e, we can insert timing buffers into the path to resolve it. Assuming B represents the set of buffer types. For each buffer b in B, its delay is $r(b)$ while its area is $a(b)$. $n(e, b)$ refers to the number of buffer b inserted into the edge e. As a result, the delay $d(e)$ of the external edge e is given by:

$$d(e) = \sum_{b \in B} n(e,b) r(b) \tag{4}$$

To fix all hold time violations of a wide-pulsed-latch based circuit while minimizing the padding hardware overhead, the target can be described as

$$\text{Minimize} \quad \sum_{e \in E_e} \sum_{b \in B} n(e,b) a(b) \tag{5}$$

Subject to

$$d(e) = \sum_{b \in B} n(e,b) r(b) \qquad \forall e \in E_e$$

$$d(e) = constant \qquad \forall e \in E_i$$

$$D(p) = \sum_{e \in E(p)} d(e) \qquad \forall p \in P$$

$$D(p) \le T \qquad \forall p \in P$$

$$D(p) \ge t \qquad \forall p \in P$$

According to the above equations and inequalities, we solved the short path padding problem by the ILP solver and a short path padding software is developed.

In previous works, the buffers used in short padding are single standard-V_t cells. As we know, in the near-/sub-threshold voltage regime, high-V_t cells have much longer delay and less power than low-V_t and standard-V_t cells [6]. To reduce the padding hardware cost, low-V_t, standard-V_t and high-V_t cells are employed by our algorithm. Using multiple-V_t buffer cells can remove hold time violations with lower additional area and power cost. Similar to the baseline algorithm (short path padding with standard-V_t cells), we extend B to a set of multiple-V_t buffer types when performing short path padding.

B. Step-by-step based short path padding scheme

One of the key issues in using ILP to solve the short path padding problem in wide-pulsed-latch based circuit is its long runtime, especially for large scale circuits.

The runtime of ILP increases rapidly when the number of variable becomes large. We can observe that the total number of variables $n(e, b)$ in ILP is the product of $|E_e|$ and $|B|$ in baseline algorithm. So we proposed a scheme that we choose a buffer type in B and use this single buffer to perform short path padding in one iteration. This method avoids a huge number of variables in one ILP, and instead it performs several ILPs with much smaller number of variables step by step. The method is described in pseudocode as shown in Fig. 2.

Algorithm *Step_by_Step(G, B, T, t)*

L1 *Sort(B, key=B.delay, order=descend)*

L2 **For** *b* **in** *B*:

L3 **if** *t - b.delay > 0*:

L4 *t' = t – b.delay ∗ factor*

L5 *B' = list(b)*

L6 *Buffer_Insertion(G, B', T, t')*

L7 *Timing_Update(G)*

Fig. 2. Pseudocode of step-by step based short path padding algorithm

Firstly, we sort the buffer type set B according to the delay value of each buffer type in an descending order (L1) to insure that the delay of chosen buffer type varies from large to small. Then we will check whether the delay of b is larger than the required hold time t or not. If so, we skip this buffer because it's delay is too large to be padded. If not, we define a new required hold time t' (descried in L4). Its value is the difference of the original required hold time t and the delay of buffer b multiplied by a fitting *factor*. Now the new buffer set B' has the only element b (L5). At last, we perform the ILP solver with the new parameters and update the timing information in graph G (L6, L7).

C. Path group based short path padding scheme

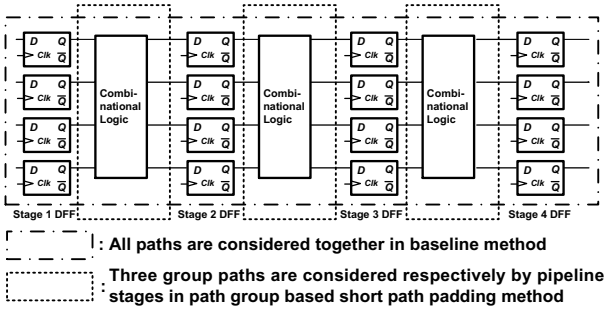

Fig. 3. Path group based circuit partitioning and short path padding scheme

Circuit partitioning can divide a big circuit into several small sub-circuits while keep the timing closure. As a result, the scale of the problem of the short path padding algorithm can be reduced effectively. Fig. 3 shows an example of 3 stage pipeline which can be divided into three path groups according to the pipeline stages. Using the divide and conquer strategy, the path group based short path padding algorithm is descried in Fig.4.

Algorithm *Path_Group(G, B, T, t, stage_list)*

L1 **For** *stage, next_stage* **in** *stage_list*:

L2 *G' = Graph_Partition(G, stage, next_stage)*

L3 *Buffer_Insertion(G', B, T, t)*

L4 *Timing_Update(G)*

Fuction *Graph_Partition(G, stage, next_stage)*

L5 *Paths = list()*

L6 **For** *start_reg* **in** *stage:*

L7 **For** *end_reg* **in** *next_stage:*

L8 *Paths += Find_Paths(G, start_reg, end_reg)*

L9 ***Return*** *G' = Transform(Paths)*

Fig. 4. Pseudocode of path group based short path padding algorithm

Assuming that the variable *stage_list* stores the register list in the order of pipeline stage, we firstly choose two contiguous stages iteratively (L1). Then we partition the original graph *G* using these two stages to get a sub-graph *G'* (L2). The partition is to get a path group (variable *Paths*) by finding all paths from the register in the previous stage to the current stage (L5-L8). According the path groups, we can get a sub-graph *G'* transformed from it (L9). At last, we perform the ILP solver with *G'* and update the timing information in graph *G* (L6, L7).

D. The hybrid scheme based on step-by-step and path group

In this section, we propose a hybrid padding scheme based on the previous two methods to reduce the algorithm complexity and runtime further. For each sub-graph partitioned from the whole graph, we use the step-by-step padding scheme to remove the hold time violation. The pseudocode is shown in Fig. 5. This algorithm is similar to the path group based short path padding algorithm, applying step-by-step based padding algorithm to each *G'* instead.

Algorithm *Hybrid_Method (G, B, T, t, stage_list)*

L1 **For** *stage, next_stage* **in** *stage_list*:

L2 *G' = Graph_Partition(G, stage, next_stage)*

L3 *Step_by_step(G', B, T, t)*

L4 *Timing_Update(G)*

Fig. 5. Psedudocode of the hybrid algorithm

III. EXPERIMENTAL RESULTS AND COMPARISONS

The above three short path padding algorithms with a ILP solver are implemented in Python and PuLP [13]. The experiments are performed on a computer with a 2.2GHz Intel Core i7 processor and 16GB memory running at macOS Sierra operating system.

A. Test cases and test setup

To test runtime and additional hardware cost of padding among different algorithms, two sets of combinational and sequential circuits from ISCAS benchmark shown in Table I and II are used in three experiments: short path padding with standard-V_t cells, short path padding with multiple-V_t cells, and step-by-step based padding scheme.

For the path group based padding algorithm and the hybrid algorithm, we use an 8 bit RCA circuit, an 8×8 multiplier circuit and some pipelines built by these RCAs and multipliers. For example, the circuit rca10 shown in Table III and Table IV means a pipeline circuit with ten stage of RCAs. Rca_mul_2 means a pipeline circuit with two stage of RCAs and two stage of multipliers. All of these test circuits are synthesized with TSMC 65nm CMOS technology at 0.35V which is operated in the near-/sub-threshold regime. Meanwhile, the target pulse with is set to 1/3 of the cycle time, which is the same as the required hold time. This wider pulse can enable a larger time-borrowing ability and tolerance of variations at ultra-low supply voltage. Compared to previous narrow pulsed-latch based designs, we extend the pulse width by 3X and thus have a 3X time-borrowing ability and tolerance of variations. At the same time, the short path padding technique should meet the hold time requirement while keep the critical path delay unchanged.

B. Simulation results and discussions

To compare the padding hardware cost using the standard-V_t buffer cells with that multiple-V_t buffer cells, six benchmark circuits are employed as shown in Table I. The *Area* (column 3, 4) means the hardware cost in μm^2 caused by additional delay buffers after short path padding.

TABLE I. THE PADDING AREA OF STANDARD-VT AND MULTIPLE-VT

Circuit	#Paths	Standard-V_t	Multiple-V_t	ΔArea
		Area	*Area*	
s820	510	102.96	49.68	51.75%
s1488	1059	59.04	27.72	53.05%
s9234	4823	800.64	408.96	48.92%
s13207	9651	3174.12	1620.0	48.96%
s35932	11099	4485.6	1688.04	62.37%
c1908	13960	279.36	143.28	48.71%
Avg.		1483.62	656.28	52.29%

TABLE II. THE PADDING AREA AND RUNTIME OF BASELINE AND STEP-BY-STEP BASED ALGORITHM

Circuit	#Paths	Baseline algorithm		Step-by-step based algorithm			
		Area	*Time*	*Area*	*Time*	Δ*Area*	Δ*Time*
s5378	4721	951	32.8	987	5.9	3.8%	82.2%
s35932	11099	4486	26.5	4510	8.5	0.5%	67.8%
c1908	13960	279	551.5	293	9.9	5.0%	98.2%
c2670	23345	1124	47.3	1130	27.7	0.5%	41.5%
s38584	40955	8497	1293.5	8591	39.8	1.1%	96.9%
s15850	396030	4222	28715	4262	2632	0.9%	90.8%
Avg.		3260	5111.1	3296	454	2.0%	79.6%

TABLE III. THE RUNTIME AND AREA OF PADDING BUFFERS OF BASELINE, PATH GROUP BASED AND HYBRID ALGORITHM

Circuit	#Paths	Baseline algorithm		Path group based algorithm			Hybrid algorithm			
		Time	*Area*	*Time*	*Area*	Δ*Time*	*Time*	*Area*	Δ*Time*	Δ*Area*
rca10	29455	97.96	1085.76	21.90	1085.76	77.64%	18.63	1114.56	80.98%	2.65%
rca15	44680	122.07	1564.56	29.14	1564.56	76.13%	29.09	1611.72	76.17%	3.01%
rca_mul_1	133518	563.59	342.00	269.46	342.00	52.19%	189.15	361.80	66.44%	5.79%
rca_mul_2	211744	3812.72	551.16	396.45	551.16	89.60%	288.72	582.48	92.43%	5.68%
rca_mul_3	289970	9170.50	760.32	719.47	760.32	92.15%	409.15	813.24	95.54%	6.96%
multipiler2	207695	1057.93	313.20	402.30	313.20	61.97%	277.92	320.76	73.73%	2.41%
Avg.		2470.79	769.50	304.20	769.50	74.95%	202.11	800.76	80.88%	4.42%

The ΔArea (column 5) represents the ratio of the reduced buffer area with multiple-V_t cells compared to that with standard-V_t cells in percentage. Compared to the standard-V_t buffer based padding, our proposed multiple-V_t cells based short path padding technique can reduce the hardware cost between 48.71% and 62.37%, and achieves 52.29% hardware cost reduction on average.

Table II shows the padding hardware cost and algorithm runtime of the baseline method and our proposed step-by-step based method. The *Time* column presents the runtime in seconds. The Δ*Time* (last column) represents the ratio of the reduced runtime with the step-by-step based scheme compared to the baseline algorithm runtime in percentage. The Δ*Area* (column 7) represent the ratio of the increased buffer area by the step-by-step based scheme compared to the baseline method buffer area in percentage. Compared to the baseline method, the runtime of our proposed step-by-step based algorithm is reduced by 41.5% at least and 79.6% on average while the additional buffer area is increased from 0.5% to 5.0%, and 2.0% on average.

Table III shows the algorithm runtime and the padding hardware cost of the baseline method, proposed path group based algorithm and hybrid algorithm. Compared to the baseline algorithm, the runtime of the path group based algorithm is reduced by 52.19% at least and 74.95% on average while the hybrid algorithm is 66.44% and 80.88%, respectively. Compared to the baseline method, the path group based algorithm has no hardware cost increment while the hybrid method increases from 2.41% to 6.96%, 4.42% on average.

In summary, the experimental results show that compared to previous works, our proposed short path padding technique by using multiple-V_t buffer cells can reduce the padding cost effectively. Furthermore, our proposed three padding schemes can reduce the runtime of the padding algorithm effectively.

IV. CONCLUSION

In this paper, an effective short path padding technique is proposed to remove hold time violations in wide-pulsed-latch based circuits. To reduce the hardware padding cost and runtime of the padding algorithm effectively, multiple-V_t buffer cells, step-by-step based padding scheme, path group-based padding scheme and hybrid scheme are proposed. Experimental results show that our proposed shot path padding technique can reduce 52.29% of buffer area on average. Furthermore, the step-by-step based and path group based padding scheme can reduce 79.6% and 74.95% of algorithm runtime respectively. Combing with these two methods, the runtime reduction is up to 80.88% on average.

REFERENCES

[1] Hyein Lee, Seungwhun Paik and Youngsoo Shin, "Pulse width allocation with clock skew scheduling for optimizing pulsed latch-based sequential circuits," IEEE/ACM International Conference on Computer-Aided Design, pp. 224-229, 2008.

[2] H. M. Chou, H. Yu and S. C. Chang, "Useful-skew clock optimization for multi-power mode designs," IEEE/ACM International Conference on Computer-Aided Design, pp. 647-650, 2011.

[3] H. T. Lin, Y. L. Chuang and T. Y. Ho, "Pulsed-latch-based clock tree migration for dynamic power reduction," IEEE/ACM International Symposium on Low Power Electronics and Design, pp. 39-44, 2011.

[4] Y. L. Chuang, S. Kim, Y. Shin and Y. W. Chang, "Pulsed-latch aware placement for timing-integrity optimization," ACM/IEEE Design Automation Conference, pp. 280-285, 2010.

[5] S. Lee, S. Paik and Y. Shin, "Retiming and time borrowing: Optimizing high-performance pulsed-latch-based circuits," IEEE/ACM International Conference on Computer-Aided Design, pp. 375-380, 2009.

[6] W. Jin, S. Kim, W. He, Z. Mao, M. Seok, "A 0.35V 1.3pJ/Cycle 20MHz 8-Bit 8-Tap FIR Core Based on Wide-Pulsed-Latch Pipelines," IEEE Asian Solid-State Circuits Conference (A-SSCC), pp. 129-132, 2016.

[7] Inhak Han, Daijoon Hyun and Youngsoo Shin, "Buffer insertion to remove hold violations at multiple process corners," 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macau, 2016, pp. 232-237.

[8] T. Xiao, H. Bagga, G. J. Chen, R. Cheung and R. Pattipati, "Path aware event scheduler in HoldAdvisor for fixing min timing violations," Computer Design (ICCD), 2011 IEEE 29th International Conference on, Amherst, MA, 2011, pp. 71-77.

[9] Y. M. Yang, I. H. R. Jiang and S. T. Ho, "PushPull: Short-Path Padding for Timing Error Resilient Circuits," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 4, pp. 558-570, April 2014.

[10] Pei-Ci Wu, M. D. F. Wong, I. Nedelchev, S. Bhardwaj and V. Parkhe, "On timing closure: Buffer insertion for hold-violation removal," 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, 2014, pp. 1-6.

[11] N. V. Shenoy, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," Computer-Aided Design, 1993. ICCAD-93. Digest of Technical Papers., 1993 IEEE/ACM International Conference on, Santa Clara, CA, USA, 1993, pp. 156-161.

[12] W. P. Tu, C. H. Chou, S. H. Huang, S. C. Chang, Y. T. Nieh and C. Y. Chou, "Low-power timing closure methodology for ultra-low voltage designs," 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, 2013, pp. 697-704.

[13] PuLP 1.6.1, [Online]. Available: https://pypi.python.org/pypi/PuLP/

[14] W. Jin, S. Kim, W. He, Z. Mao, and M. Seok, "In Situ Error Detection Techniques in Ultralow Voltage Pipelines: Analysis and Optimizations," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, pp. 1032-1043, 2017.