Pipelining For Dual Supply Voltages

Teng Xu, Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
{xuteng, miodrag}@cs.ucla.edu

***Abstract*—Power is one of the most important metrics in the modern integrated circuit design. We optimize the circuit power using a dual-supply voltage (dual-V_{dd}) approach. In order to achieve an improved power efficiency, we have applied a new type of pipelining to reduce the number of gates need to be assigned to the high supply voltage given a target delay. Compared to the standard pipelining and the standard dual-V_{dd} assignment, our design achieves the best of the two approaches without compromising any performance. Our overall design is tested on a set of standard ISCAS-85 benchmark circuits using an industrial cell library. Significant power savings (Avg. 10.4%) under a specified target delay are observed.**

I. Introduction

Low power and low delay have always been among the most important metrics in the modern circuit design. Systems such as mobile devices and wireless sensor networks have imposed new requirements to energy efficiency and utilization. For example, battery technology improvements have not kept pace with the rapid device scaling that has continued to follow Moore's Law. The platforms such as mobile phones and tablets are highly restricted by the amount of energy that can be stored in the battery and, therefore, energy efficiency has become a premier design requirement. On the other hand, energy consumption directly impacts the circuit temperature, which leads to the creation of hotspots. Frequent and significant fluctuations in temperature can have ramifications on the circuit reliability. To summarize, energy minimization is of great importance to the modern circuit design.

Many efforts have been made to achieve power/delay optimization, two techniques among them are most recognized, respectively pipelining, and dual-V_{dd}. Pipelining is a circuit optimization technique which is frequently used to speed up the execution of computations in a circuit by dividing the original circuit into n consecutive sub-computation units and overlapping their executions. By adding flip-flops between the sub-computation units to temporarily store the intermediate results, theoretically, an n-stage pipeline circuit should yield a factor of n times delay improvement over the non-pipelined circuit. Dual-V_{dd} is another circuit design technique in which two supply voltages (high V_{dd} and low V_{dd}) are applied to a circuit in order to save power compared to only applying a single supply voltage (medium V_{dd}). The apply of dual-V_{dd} optimization should not sacrifice either the throughput or the delay of the original circuit.

It is a natural idea to combine pipelining with dual-V_{dd} to jointly optimize circuit, expecting to achieve both low delay and low power. However, the optimal way to combine the two techniques is not yet a solved problem, which is due to the following two facts. The first is that there exist multiple pipeline schemes with the same circuit delay. For example, assume that a circuit is to be transformed into a 2-stage pipelining. Since the delay of a pipeline circuit is decided by the maximum delay of each single stage of the circuit, then it is important to guarantee that the delay of the original critical path d_c is evenly divided in such a way that each stage has delay $d_c/2$. However, the non-critical path part of the circuit can be assigned to either stage as long as they do not create a new critical path longer than $d_c/2$. Consequently, multiple pipeline schemes with the same delay exist. The second fact is a technical constraint of dual-V_{dd} assignment. Only the gates on high V_{dd} can directly drive gates on low V_{dd} to prevent DC current due to incomplete PMOS cut-off [1]. The only exception is that a level converting flip-flop allows the logic after low V_{dd} gates to switch back to high V_{dd}. To summarize, only high V_{dd} gates can drive low V_{dd} gates unless converting flip-flops are used.

Combining the above two observed facts, we claim that different pipeline schemes even though with the same delay can have very different power consumption after dual-V_{dd} optimization. Figure 1 shows a motivational example of the effect of dual-V_{dd} assignment on a circuit with different pipelining. The original circuit contains 8 gates, from $G1$ to $G8$. Assume that all the gates have the same delay d_G on the original V_{dd}. Both pipeline designs in (a) and (b) introduce 4 extra flip-flops. We denote that the delay of flip-flops equals to d_{FF} on the original voltage. Then for both pipeline designs in (a) and (b) before dual V_{dd} optimization, the circuit has the same delay $2*d_G+d_{FF}$. Now we apply dual-V_{dd} optimization on both pipeline designs. The gates and flip-flops on high V_{dd} are colored red (solid line) and their delays are denoted as d_G^h and d_{FF}^h. Correspondingly, the gates and flip-flops on low V_{dd} are colored blue (dashed line) and the delays are denoted as d_G^l and d_{FF}^l. Thus, for both pipeline designs in (a) and (b) after dual-V_{dd}, the critical path delay equals to $d_G^l+d_G^h+d_{FF}^h$. Note that in our dual V_{dd} optimization, the critical path delay before and after the optimization should stay the same. However, the key observation is that the number of gates assigned to high V_{dd} in (a) is 4 while the number of high V_{dd} gates in (b) equals to 2. It suggests that the circuit power consumption after dual V_{dd} assignment in (b) is less than the power consumption in (a) although both circuit designs enable the same delay.

Motivated by the above example, we propose a pipeline design with the use of an objective function in which the

978-1-5090-0733-2/16/$31.00 ©2016 IEEE

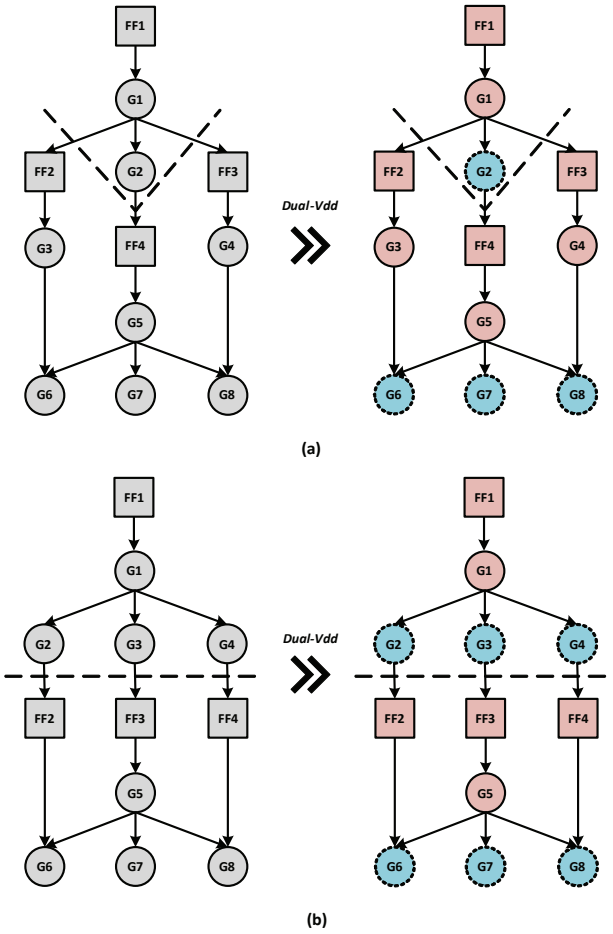

Fig. 1: An example of dual-V_{dd} assignment on (a) non-optimized pipelining, (b) optimized pipelining. The flip-flops in the circuit are represented with rectangle, and the gates are represented with circle. The gates/flip-flops that are on original V_{dd} are colored grey (middle V_{dd}), the ones on low V_{dd} are colored blue (dashed line), and the ones on high V_{dd} are colored red (solid line).

number of gates on high V_{dd} is minimized under the constraint of keeping the delay of the standard optimal pipelining. We only introduce level converting flip-flops between the stages of pipelining, which suggests that within each pipeline stage only high V_{dd} gates can drive low V_{dd} gates. Therefore, it is important that the width of the circuit is small at the positions immediately after the flip-flops so that fewer gates need to be placed at high V_{dd} while satisfying the delay constraints. To distinguish our paper from the previous related work of pipelining and dual voltage assignment, we summarize our contributions as following.

- A novel approach to combine the pipelining and the dual-V_{dd} assignment to jointly optimize energy consumption.
- Modification to the current pipelining algorithm to enable a more energy efficient dual-V_{dd} assignment.

The rest paper is organized in the following way. We first introduce the related work in Section II. Section III offers an overview of our optimization flow. Section IV depicts the power and delay model we have applied. Then we separately demonstrate our design for pipelining and dual-V_{dd} optimization in Section V and Section VI. Lastly, we present our simulation results on ISCAS-85 benchmark circuits in Section VII.

II. Related Work

We review the previous literature on pipelining and dual-V_{dd} optimization in this section.

A. Pipelining

The task of pipelining is described in standard textbooks such as [2] and [3]. Many researches have been focused on the automatic generation of high efficient pipelining designs. Kroening et al. proposed automated pipelining which transforms a sequential machine into a pipelined machine by adding forwarding and interlock logic [4]. Cong et al. presented an architecture-level synthesis solution to support automatic pipelining of on-chip interconnections [5]. The work from Calceran-Oms et al. discussed an automatic pipelining method for micro-architectural systems with loops [6]. Most of the conventional pipelining methods target to maximize the processing performance by reducing the clock period [7][8].

B. Dual V_{dd} Optimization

Usami et al. first proposed to use multiple supply voltages as a way to reduce energy [9][10]. Salil and Sarrafzadeh applied multiple supply voltages at the behavior level for energy minimization [11]. Dynamic programming techniques for solving the dual-V_{dd} scheduling problem in both non-pipelined and functionally pipelined data-paths were proposed by Chang et al. [12]. A low power implementation on the FPGA platform using dual-V_{dd}/dual-V_{th} fabrics was proposed by He [13][14]. Ishihara proposed the level converter required for dual-V_{dd} systems [15]. Srivastava has introduced technology that minimizes both switching and static power using simultaneous V_{dd} assignment [16]. Lee et al. studied the use of dual voltages by considering the requirement for power-network planning [17]. More recently, Xu et al. proposed to combine dual-voltage assignment with retiming to jointly optimize circuit energy [18].

To the best of our knowledge, we report the first approach that combines pipelining and dual voltage assignment for the reduction of circuit power. Our dual-V_{dd} assignment method specifically does not require voltage converters inside each pipeline stage. Under this assumption, our algorithm guarantees significant power minimization using dual-V_{dd} assignment. The most important observation is that pipelining is used both for delay minimization as well as to improve the performance of dual-V_{dd} assignment. The effectiveness of our approach in this generalized technology can be further combined with unfolding and other sequential and combinational transformations.

III. Optimization Flow

Our flow of optimization is shown in Figure 2. In the first stage, we apply gate level simulation to the benchmark circuits based on Markovic's delay and power model [19]. Based on the achieved circuit model, we then apply pipelining to find out where to split the pipeline stages. Our pipelining has the following properties. (1) It does not compromise the performance of standard optimal pipelining. (2) It facilitates the dual-V_{dd} assignment. In the simulation, our pipelining is tested on the 2-stage situation, however, it is not limited to such, as it can be easily extended to an n-stage pipelining ($n > 2$). The last step in the flow is to assign dual-V_{dd} to the circuit which operates at the gate-level. Our cell V_{dd} selection strategy is by efficiently identifying gates to place at high voltage without violating the basic rules that only high V_{dd} gates can drive low V_{dd} gates. In the dual-V_{dd} optimization, we set a target delay and then assign dual voltages to optimize the overall circuit power. Our algorithm returns both the optimal high/low voltages to apply as well their coverage on the circuit.

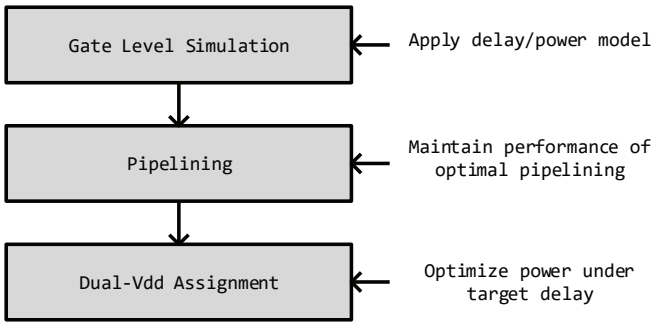

Fig. 2: Flow of optimization.

IV. Power and Delay Model

We build our gate level simulation model by using the delay and power models from Markovic et al. [19]. The delay model is shown in Equation (1), where k_{tp} is the delay-fitting parameter, C_L is load capacitance, V_{dd} is supply voltage, n is substreshold slope, μ is mobility, C_{ox} is oxide capacitance, W is gate width, L is effective channel length, $\phi_t = kT/q$ is thermal voltage, k_{fit} is a model-fitting parameter, σ is the drain induced barrier lowering (DIBL) factor, and V_{th} is threshold voltage. Load capacitance C_L is defined in Equation (2), where γ is the logical effort of the gate and W_{fanout} is the sum of the widths of the load gates.

$$D = \frac{k_{tp} \cdot C_L \cdot V_{dd}}{2 \cdot n \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot (\frac{kT}{q})^2} \cdot \frac{k_{fit}}{(\ln(e^{\frac{(1+\sigma)V_{dd}-V_{th}}{2 \cdot n \ \ cdot(kT/q)}}))^2} \quad (1)$$

$$C_L = C_{ox} \cdot L \cdot (\gamma \cdot W + W_{fanout}) \quad (2)$$

We have also considered two sources of power dissipation in an IC. One is from gate switching, in which the ICs dissipate power by charging the load capacitances of wires and gates. The other source is leakage power. Even if the gates do not switch, they dissipate power due to the leakage current. Equation (3) describes the leakage power model, and Equation (4) describes the gate-level switching power model, where α is the activity factor and f is the frequency.

$$P_{leakage} = 2 \cdot n \cdot \mu \cdot C_{ox} \cdot \frac{W}{L} \cdot (\frac{kT}{q})^2 \cdot V_{dd} \cdot e^{\frac{\sigma \cdot V_{dd} - V_{th}}{n \cdot (kT/q)}} \quad (3)$$

$$P_{switching} = \alpha \cdot C_L \cdot V_{dd}^2 \cdot f \quad (4)$$

V. Pipelining

We discuss our pipelining design in this section. We have focused on the 2-stage pipelining on combinational circuits. Our whole process has two major steps, respectively "critical path identification" and "splitting pipeline stages".

The first step is to identify all the critical paths in the original circuit under the single V_{dd}. This can be easily achieved through the standard critical path identification method, such as dynamic programming proposed by Kirkpatrick [20]. The clock frequency of pipeline circuit is decided by the larger delay of the pipeline stages. Therefore, in order to achieve the optimal delay after pipelining, the gates on the critical paths should be evenly split into 2 groups in terms of delay and then distributed to the two pipeline stages. Assume that the original critical path delay is d_c, then the optimal delay can be achieved after pipelining is $d_c/2 + d_{FF}$, where d_{FF} is the delay of flip-flops.

However, the key remaining question is which pipeline stage should the non-critical path logic be assigned to. Figure 3 depicts an example circuit with critical paths represented by solid black lines. The optimal cuts on the critical paths are represented by the dashed lines in the first figure of $3(a)$ and $3(b)$, with which the original critical path delay is split into approximately half to half. Then two pipeline strategies are presented in the second figure of $3(a)$ and $3(b)$, where the key difference is the assignment of non-critical path logic. Note that we assume in both pipeline cases, the assignment on non-critical path logic does not introduce new critical paths. In other words, the delay in each pipeline stage stays to be approximate $d_c/2 + d_{FF}$.

The dual-V_{dd} optimization is followed by the pipelining. We have observed a large difference in the dual-V_{dd} assignment between the pipelining in the last figure of $3(a)$ and $3(b)$. The number of gates assigned on high V_{dd} of the second stage pipeline in $3(a)$ is much larger than that of $3(b)$. This is because the critical path gates in the second stage pipeline are also driven by the non-critical path logic. Thus in order to put the critical path gates on high V_{dd}, all the non-critical path logic that provides fan-ins to them also need to be assigned to high V_{dd}. Nevertheless, to put the non-critical path logic on high V_{dd} will not further reduce the delay of pipeline stages, with the only consequence to increase the circuit power consumption.

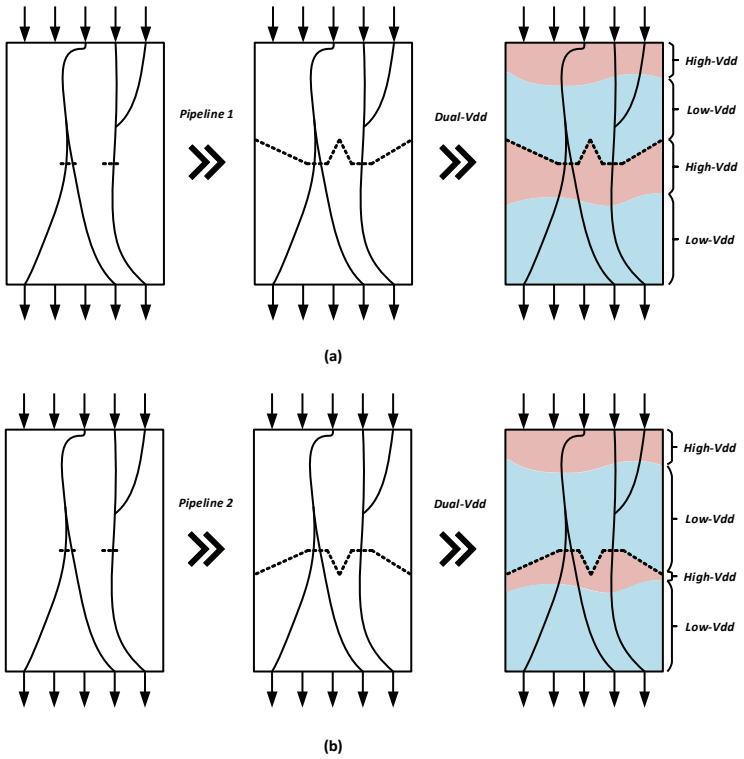

Fig. 3: Comparison of dual-V_{dd} assignment on pipelining 1 and pipelining 2.

Motivated by the above example, we summarize our key idea of pipeline design as following.

- Keep the optimal cut on the critical paths under the single V_{dd}.
- Include as much non-critical path logic as possible to the first pipeline stage without creating a new critical path.

With the above procedures, we guarantee that at the second pipeline stage, the gates on the critical path depend on as little non-critical path logic as possible since most of the logic has been included in the first pipeline stage. Consequently, when putting gates on the critical path to high V_{dd} in the second pipeline stage, the unnecessary power overhead from non-critical fan-ins is minimized.

The algorithmic process to find such pipeline design is presented in Algorithm 1. In the algorithm, we first identify all the critical paths $cp_i, (i \in \{0, 1, .., n\})$ in the original circuit, then we traverse all the gates on the critical paths with breadth-first search (BFS) to find cuts on the paths to minimize the maximum delay between the upper stage (S_1) and the lower stage (S_2) given a single V_{dd}. With the finalized cut on the critical paths, we then traverse through all the gates on the non-critical paths with BFS. As long as a gate does not create new critical paths for S_1, we incorporate it to S_1. Note that at the end of the algorithm, the delay of S_1 and the delay of S_2 should be as close as possible. And the sum of the two delays should equal to d_c plus the delay from the flip-flops.

VI. Dual V_{dd} Assignment

There exist two essential issues when applying dual-V_{dd} to circuits. The first is what voltages should be used, the second is which part of the circuit should be assigned to high V_{dd} (or low V_{dd}). We have assumed that in our design, only the gates with high V_{dd} can drive the gates with low V_{dd}, thus we do not need to use level converters that take high power consumption.

To leverage the above two issues, we propose the procedures in Algorithm 2 to heuristically approximate the best voltage pairs and the corresponding coverage. In each iteration, we choose gate g in the current critical path with the shortest arrival time and place it as well as all the gates that directly or indirectly drive gate g in the group of gates with high V_{dd} (Set'_h). Afterwards, given the current circuit configuration, we use binary search to traverse the pairs of voltages that meet the given target delay and find the pair that achieves the smallest power consumption. We repeat these steps until all gates in the circuit have been placed in the high voltage group. From there, we choose the lowest point of power consumption from all explored iterations. In practice, the minimal power is normally achieved when only a small subset of gates are assigned to high V_{dd}. Therefore, the algorithm can be improved in a way that it terminates early when no more power can be further reduced after certain iterations of the while loop.

Figure 4 depicts the performance of our algorithm on an example circuit. The tested circuit c880, has the following initial configuration: the total number of gates is 383, the initial

Algorithm 1 Pipelining

Input: C - original circuit.
Output: 2-stage pipelining on C. The gates in the first pipeline stage are in S_1 (close to inputs), and the gates in the second pipeline stage are in S_2 (close to outputs).

1. $S_1 = \varnothing$, $S_2 = \varnothing$, $d = 0$.
2. Identify all the critical paths $cp_i, (i \in \{0, 1, .., n\})$ in C with delay d_c.
3. $d = d_c$.
4. Append all gates in cp_i to S_2.
5. **For** all gates g_j in cp_i (inputs $\rightarrow$ outputs, BFS):
6. **If** g_j is directly driven by gates in S_1:
7. **If** $Max(Delay(S_1 + g_j), Delay(S_2 - g_j)) < d$:
8. S_1.append(g_j).
9. S_2.remove(g_j).
10. $d = Max(Delay(S_1), Delay(S_2))$.
11. **Else**:
12. **Continue**.
13. **Else**:
14. **Continue**.
15. $d_1 = Delay(S_1)$. $d_2 = Delay(S_2)$.
16. $S_2 = \{all\ gates\ in\ C - S_1\}$.
17. **For** all gates g_j not in cp_i (inputs $\rightarrow$ outputs, BFS):
18. **If** g_j is directly driven by gates in S_1:
19. **If** $Delay(S_1 + g_j) = d_1$ and $Delay(S_2 - g_j) \geq d_2$:
20. S_1.append(g_j).
21. S_2.remove(g_j).
22. **Else**:
23. **Continue**.
24. **Else**:
25. **Continue**.
26. **Return** S_1, S_2.

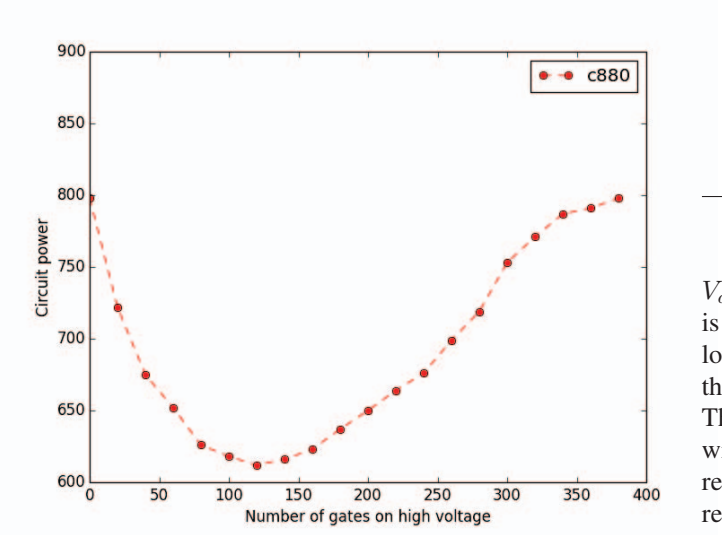

Fig. 4: An example of the performance of the dual-V_{dd} optimization algorithm on the c880 circuit.

Algorithm 2 Dual-vdd Optimization

Input: C - original circuit.
V_{dd} - original supply voltage.
Output: V_{dd}^h - applied high supply voltage.
V_{dd}^h - applied low supply voltage.
Set_h - a set that contains all the gates with V_{dd}^h.
Set_l - a set that contains all the gates with V_{dd}^l.

1. $V_{dd}^h = V_{dd}^l = V_{dd}$.
2. $Set_h = Set'_h = \varnothing$, $Set_l = Set'_l = all\ gates\ in\ C$.
3. $d_c = critical\ delay(C, V_{dd}^h, V_{dd}^l, Set_h, Set_l)$.
4. $p_c = calculate\ power(C, V_{dd}^h, V_{dd}^l, Set_h, Set_l)$.
5. **While** $Set'_l \neq \varnothing$:
6. $cp_i = identify\ critical\ path(C, V_{dd}^h, V_{dd}^l, Set'_h, Set'_l)$.
7. $Set_g = \varnothing$.
8. Find gate g on cp_i with the smallest delay slack.
9. $Set_g.append(g)$.
10. **For** all gates g_j that drives g (direct and indirect):
11. $Set_g.append(g_j)$.
12. **For** all gates g_k in Set_g:
13. **If** g_k not in Set'_h:
14. $Set'_h.append(g_k)$.
15. $Set'_l.remove(g_k)$.
16. **Else**:
17. **Continue**.
18. **For** all possible $V_{dd}^{h\prime} \geq V_{dd}$:
19. $Binary\ search V_{dd}^{l\prime} \leq V_{dd}$ such that:
20. $d_c = critical\ delay(C, V_{dd}^{h\prime}, V_{dd}^{l\prime}, Set'_h, Set'_l)$.
21. $p'_c = calculate\ power(C, V_{dd}^{h\prime}, V_{dd}^{l\prime}, Set'_h, Set'_l)$.
22. Find the minimal p_c^{min} among all p'_c. The corresponding voltages are Vdd_h^{min} and Vdd_l^{min}.
23. **If** $p_c^{min} < p_c$:
24. $Set_h = Set'_h$.
25. $Set_l = Set'_l$.
26. $Vdd_h = Vdd_h^{min}$.
27. $Vdd_l = Vdd_l^{min}$.
28. $p_c = p_c^{min}$.
29. **Else**:
30. **Continue**.
31. **Return** $Vdd_h, Vdd_l, Set_h, Set_l$.

V_{dd} is $0.7V$, the critical path delay is $3.67ps$, and the power is $798.2\mu w$. In each iteration, one gate is switched from the low V_{dd} set to the high V_{dd} set. We observe from Figure 4 that the minimal power is achieved approximately at iteration 120. The circuit achieves its optimal power consumption $612\mu W$ with high V_{dd} set to $0.74V$ and low V_{dd} set to $0.62V$. The reason that the initial part of the iteration causes more energy reduction is that as more gates are assigned to high V_{dd}, the circuit becomes balanced in terms of critical paths, thus the marginal effect on energy reduction using dual-V_{dd} is decreased. Note that throughout the whole process, the target delay of the circuit does not change, only the voltages are scaled to meet the target delay.

To apply dual V_{dd} optimization to pipeline circuit, we still stick to the algorithmic flow presented in Algorithm 2 with certain modifications. First of all, all the flip-flops in the circuit need to assigned to high V_{dd} at the start of the optimization. Secondly, instead of finding a single gate on the critical path to put into Set_h in each iteration, two gates should be put into Set_h, where one is from the critical path in S_1 and the other is from the critical path in S_2. Thirdly, in our model, we assume that the applied high V_{dd} and low V_{dd} in both stages of the circuit are the same. However, following the procedures in Algorithm 2, with the use of binary search to find the optimal pair of V_{dd} to meet the target delay, the result V_{dd} pairs for stage 1 and stage 2 are not likely to be the same. Therefore, we modify the target delay equation in line 20 of Algorithm 2 to a new delay objective function as shown in Equation 5 to enable the use of a same V_{dd} pair. In the equation, the superscript s_1 and s_2 represent pipeline stage S_1 and S_2. With the above modification, a single optimal pair of $< V_{dd}^h, V_{dd}^l >$ on both pipeline stages is discovered together with the voltage coverage.

$$\begin{aligned} &To\ minimize:\ sum\ of \\ &< |d_c^{s1} - critical\ delay(C, V_{dd}^{h\prime}, V_{dd}^{l\prime}, Set_h^{s1\prime}, Set_l^{s1\prime})| \\ &, |d_c^{s2} - critical\ delay(C, V_{dd}^{h\prime}, V_{dd}^{l\prime}, Set_h^{s2\prime}, Set_l^{s2\prime})| > \end{aligned} \quad (5)$$

VII. Experiments

We demonstrate our experimental results on our proposed approach in this section. We first explain our experiment setup, including the cell library, power model, as well as the benchmarks we are using. Then we present the power/delay results on benchmark circuits to illustrate the effectiveness of our optimization algorithm.

A. Experiment Setup

We have used the ISPD2012 standard cell library [21] and fit it accordingly to Markovic's delay and power model. We set the initial V_{dd} to 0.7V and the threshold voltage V_{th} to 0.3V. For our dual-V_{dd} optimization, we consider high and low V_{dd} within the range from 0.4V to 1.0V. We only consider 2-stage pipelining in our experiment.

We evaluate a subset of ISCAS85 benchmarks and synthesize each netlist using Cadence Encounter for generating parasitics capacitances for all considered netlists [22]. We develop an in-house timer in C++ for flexibility and robustness in computing load and slew dependent delays. Each design is optimized in accordance to the ISPD2012 design contest suite in satisfying each slew and cell load restrictions.

We show our simulation work flow in Figure 5. We start from the characterization using the cell library, afterwards, we use the gate-level simulation to quantify the delay, switching power, and leakage power. The next step is to apply the circuit optimization. We first apply our 2-stage pipelining on the benchmark circuits, and use its delay under the initial V_{dd} as the target delay. Then we further adopt dual-V_{dd} assignment on the pipeline circuit under the target delay. For both cases before and after dual-V_{dd} assignment, we test the switching power and the leakage power of the circuit and compare the total power consumption. Moreover, we have also introduced a baseline pipelining which achieves the optimal critical path delay under the single V_{dd}. But the key difference compared to our pipeline design is that the baseline pipelining randomly decides which stage the non-critical path logic belongs to. We apply the dual-V_{dd} assignment on the baseline pipelining under the same target delay and calculate the total power consumption.

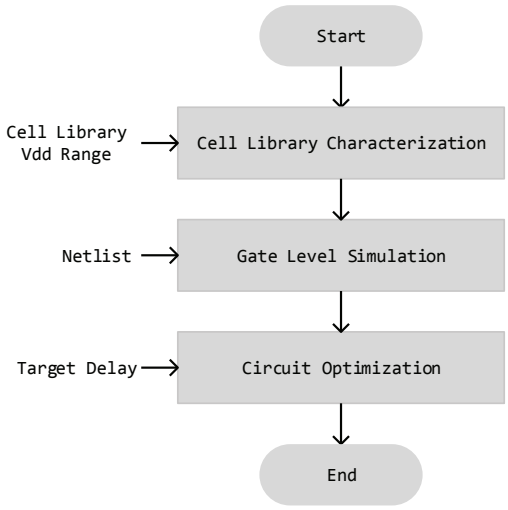

Fig. 5: Work flow of our simulation.

B. Results

We test the power consumption of each benchmark circuit under target delay for three different cases, pipelining with initial V_{dd}, baseline pipelining with dual-V_{dd}, and our pipelining with dual-V_{dd}. The results are presented in Table I. Compared to the power consumption under initial V_{dd}, the introduce of dual-V_{dd} together with baseline pipelining can provide 3.9% to 18.5% (Avg. 11.4%) power saving while the combination of dual-V_{dd} with our pipelining can provide 3.9% to 29.2% (Avg. 20.5%) power saving. If we directly compare our pipelining with baseline pipelining under dual V_{dd} optimization, the improvement is from 0% to 24.0% (Avg. 10.4%). Note that the low/high voltage pairs presented in the table are calculated based on Algorithm 2. We draw the following conclusions from the results.

First of all, the power consumption of our pipeline design achieves better performance compared to the baseline pipeline design when combined with the dual-V_{dd} optimization. The power saving from the baseline pipelining to our pipelining is even comparable to the power saving from single V_{dd} to dual V_{dd} with baseline pipelining. In all the tested circuits, our pipelining algorithm is at least equally well performed as the standard pipelining algorithm regarding power consumption.

Circuit	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
Number of gates	160	202	383	546	880	1193	1669	2307	2416	3512
Target Delay(ns)-pipeline+initial V_{dd}	2.03	2.12	1.91	2.16	2.47	3.09	3.54	3.38	13.58	2.71
Power(μW)-pipeline+initial V_{dd}	**374.8**	**536.9**	**798.2**	**1158.5**	**1816.7**	**2355.8**	**3339.9**	**4582.0**	**5376.8**	**7026.1**
Power(μW)-**baseline pipeline**+dual V_{dd}	**342.4**	**516.2**	**683.8**	**1067.3**	**1480.4**	**1925.2**	**3001.3**	**3857.9**	**5119.9**	**6186.2**
$< low\ V_{dd}, high\ V_{dd} >(V)$-**baseline pipeline**+dual V_{dd}	$< 0.55, 0.76 >$	$< 0.52, 0.75 >$	$< 0.54, 0.74 >$	$< 0.55, 0.76 >$	$< 0.55, 0.75 >$	$< 0.51, 0.77 >$	$< 0.53, 0.75 >$	$< 0.57, 0.76 >$	$< 0.53, 0.75 >$	$< 0.55, 0.75 >$
# of gates: <stage1, stage2> -**baseline pipeline**	$< 58, 102 >$	$< 80, 122 >$	$< 74, 309 >$	$< 296, 250 >$	$< 435, 445 >$	$< 393, 800 >$	$< 507, 1162 >$	$< 398, 1909 >$	$< 1453, 963 >$	$< 723, 2789 >$
Power(μW)-**our pipeline**+dual V_{dd}	**282.1**	**516.2**	**678.9**	**955.3**	**1434.8**	**1667.6**	**2281.8**	**3234.5**	**4873.5**	**5322.9**
$< low\ V_{dd}, high\ V_{dd} >(V)$-**our pipeline**+dual V_{dd}	$< 0.51, 0.77 >$	$< 0.52, 0.75 >$	$< 0.53, 0.76 >$	$< 0.55, 0.76 >$	$< 0.54, 0.75 >$	$< 0.53, 0.76 >$	$< 0.54, 0.76 >$	$< 0.55, 0.74 >$	$< 0.53, 0.75 >$	$< 0.52, 0.76 >$
# of gates: <stage1, stage2> -**our pipeline**	$< 126, 34 >$	$< 80, 122 >$	$< 82, 301 >$	$< 375, 171 >$	$< 513, 367 >$	$< 736, 457 >$	$< 898, 771 >$	$< 1081, 1226 >$	$< 1809, 607 >$	$< 2368, 1144 >$
Power Save-initial V_{dd} to baseline pipeline+dual V_{dd}	8.6%	3.9%	14.3%	7.9%	18.5%	18.3%	10.1%	15.8%	4.8%	12.0%
Power Save-initial V_{dd} to our pipeline+dual V_{dd}	24.7%	3.9%	14.9%	17.5%	21.0%	29.2%	31.7%	29.4%	9.4%	24.2%
Power Save-dual V_{dd}, baseline pipeline to our pipeline	**17.6%**	**0%**	**0.7%**	**10.5%**	**3.1%**	**13.4%**	**24.0%**	**16.2%**	**4.8%**	**14.0%**

TABLE I: Experimental results of ISCAS-85 benchmark circuits with 2-stage pipelining and dual-V_{dd} optimization. Two types of pipelining are presented: baseline pipelining and our pipelining. Stage 1 represents the pipeline stage close to inputs, and stage 2 represents the pipeline stage close to outputs.

Secondly, our pipelining achieves more power saving when the circuit has a larger number of gates. It is because for a larger circuit, there normally exists more non-critical path logic near the optimal critical path cut. Therefore in our pipelining, such logic is merged to the first stage of pipelining. The gates in such logic are now switched from high V_{dd} to low V_{dd} without compromising circuit delay. From the results, we can clearly see that if many more gates are assigned to S_1 (upper stage) in our pipelining than the baseline pipelining, it normally suggests that the power saving between the two pipeline designs is also large.

Lastly, there also exists some situation when our pipelining helps little compared to the baseline pipelining, such as circuit $c499$ and $c880$ where the power savings are 0% and 0.7%. When taking a closer look at both circuits, it is because the area of the circuit near the optimal critical path cut is ultra "thin", which in other words, exists almost no non-critical path logic. As a result, the stage split between the baseline pipelining and our pipelining stays almost the same, causing no power saving available.

Our pipelining is also applicable to an n-stage pipeline circuit design. The key idea remains the same, which is to reduce the number of non-critical path gates at the start of each stage so that these gates do not need to be assigned to high V_{dd}. The same algorithm can be applied and easily extended to an n-stage pipelining. Another extension of our current work is to apply our pipelining on sequential circuits in which case we can no longer arbitrarily "add" flip-flops to the circuit. Instead, we need to move the flip-flops around the circuits with the technology of retiming.

To visualize the effect of our algorithm, Figure 6 presents the final standard cell layout using our pipelining and dual-v_{dd} assignment on benchmark circuits c7552 (3512 cells). The red cells include all the flip-flops and the gates placed at the high V_{dd} and the blue corresponds to the low V_{dd} gates. For benchmark c7552, 23% of the cells were placed at the higher V_{dd}. Although we do not consider cell placement optimization in this paper, in addition to energy minimization, applying a pipelining+dual-V_{dd}-aware procedure can be used in a generic cell placement tool for power grid optimization to ensure circuit reliability across the power network of a design. Under these scenarios, the location of higher voltage cells directly impacts the placement of high-voltage power rails of a design. We leave this direction to future work.

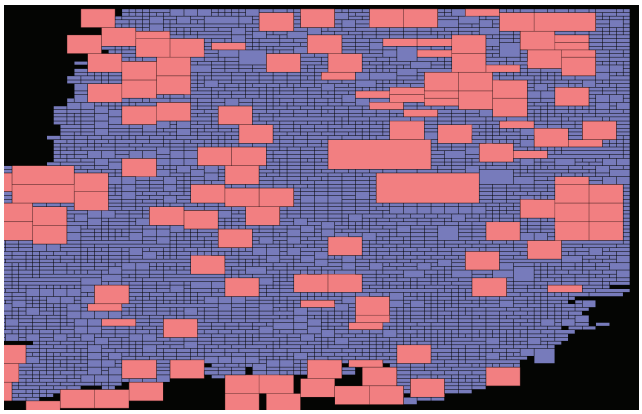

Fig. 6: Benchmarks c7552 layout after applying the pipelining and dual-voltage assignment. The red cells are the gates/flip-flops in high V_{dd} and the blue cells are the ones in low V_{dd}.

VIII. Conclusion

A novel design of pipelining is proposed to facilitate dual-V_{dd} assignment to enable low power consumption. While our pipeline design maintains the optimal pipeline delay on the circuit, it allocates as much non-critical path logic as possible from high V_{dd} to low V_{dd} to reduce power consumption. Our approach is by far the first attempt to optimize circuit pipelining in conjunction with dual-V_{dd} optimization. We have tested our approach on the ISCAS-85 benchmark circuits, and the results suggest that in all benchmark circuits, our pipelining algorithm performs at least equally well as the standard pipelining algorithm after dual-V_{dd} assignment. An average power saving of 10.4% is observed when comparing our pipelining to the baseline pipelining.

IX. Acknowledgment

This work was supported in part by the NSF under Award CNS-0958369, and Award CNS-1059435.

References

[1] K. Usami, M. Igarashi, F. Minami, T. Ishikawa, M. Kanzawa, M. Ichida, and K. Nogami, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *Solid-State Circuits, IEEE Journal of*, vol. 33, no. 3, pp. 463–472, 1998.

[2] M. J. Flynn, "Computer Architecture Pipelined and Parallel Processor Design, 1995."

[3] D. A. Patterson and J. L. Hennessy, *Computer organization and design: the hardware/software interface*. Newnes, 2013.

[4] D. Kroening and W. J. Paul, "Automated pipeline design," in *Design Automation Conference, 2001. Proceedings*, pp. 810–815, IEEE, 2001.

[5] J. Cong, Y. Fan, and Z. Zhang, "Architecture-level synthesis for automatic interconnect pipelining," in *Proceedings of the 41st annual Design Automation Conference*, pp. 602–607, ACM, 2004.

[6] M. Galceran-Oms, J. Cortadella, D. Bufistov, and M. Kishinevsky, "Automatic microarchitectural pipelining," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 961–964, European Design and Automation Association, 2010.

[7] Y. Ma, Z. Li, J. Cong, X. Hong, G. Reinman, S. Dong, and Q. Zhou, "Micro-architecture pipelining optimization with throughput-aware floorplanning," in *Design Automation Conference, 2007. ASP-DAC'07. Asia and South Pacific*, pp. 920–925, IEEE, 2007.

[8] E. Nurvitadhi, J. C. Hoe, S.-L. L. Lu, and T. Kam, "Automatic multi-threaded pipeline synthesis from transactional datapath specifications," in *Proceedings of the 47th Design Automation Conference*, pp. 314–319, ACM, 2010.

[9] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design," in *Proceedings of the 1995 international symposium on Low power design*, pp. 3–8, ACM, 1995.

[10] M. Igarashi, K. Usami, K. Nogami, F. Minami, Y. Kawasaki, T. Aoki, M. Takano, C. Mizuno, T. Ishikawa, M. Kanazawa, *et al.*, "A low-power design method using multiple supply voltages," in *Proceedings of the 1997 international symposium on Low power electronics and design*, pp. 36–41, ACM, 1997.

[11] S. Raje and M. Sarrafzadeh, "Variable voltage scheduling," in *Proceedings of the 1995 international symposium on Low power design*, pp. 9–14, ACM, 1995.

[12] J.-M. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 436–443, 1997.

[13] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics," in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, pp. 42–50, ACM, 2004.

[14] Y. Lin and L. He, "Statistical dual-Vdd assignment for FPGA interconnect power reduction," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pp. 1–6, IEEE, 2007.

[15] F. Ishihara, F. Sheikh, and B. Nikolić, "Level conversion for dual-supply systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, no. 2, pp. 185–195, 2004.

[16] A. Srivastava and D. Sylvester, "Minimizing total power by simultaneous V dd/V th assignment," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, no. 5, pp. 665–677, 2004.

[17] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "An ILP algorithm for post-floorplanning voltage-island generation considering power-network planning," in *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pp. 650–655, IEEE, 2007.

[18] T. Xu and M. Potkonjak, "Retiming and dual-supply voltage based energy optimization for dsp applications," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1055–1059, IEEE, 2016.

[19] D. Marković, C. C. Wang, L. P. Alarcon, T.-T. Liu, and J. M. Rabaey, "Ultralow-power design in near-threshold region," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 237–252, 2010.

[20] T. Kirkpatrick and N. Clark, "PERT as an aid to logic design," *IBM Journal of Research and Development*, vol. 10, no. 2, pp. 135–141, 1966.

[21] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo, "The ISPD-2012 discrete cell sizing contest and benchmark suite," in *Proceedings of the 2012 ACM international symposium on International Symposium on Physical Design*, pp. 161–164, ACM, 2012.

[22] F. Brglez, "A neutral netlist of 10 combinational benchmark circuits and a target translation in FORTRAN," in *ISCAS-85*, 1985.