Physical vs. Physically-Aware Estimation Flow: Case Study of Design Space Exploration of Adders

Ivan Ratković*‡, Oscar Palomar*‡, Milan Stanić*‡, Osman Unsal*, Adrian Cristal*‡§, Mateo Valero*‡
* Barcelona Supercomputing Center {first}.{last}@bsc.es ‡ UPC § CSIC-IIIA

***Abstract*—Selecting an appropriate estimation method for a given technology and design is of crucial interest as the estimations guide future project and design decisions. The accuracy of the estimations of area, timing, and power (metrics of interest) depends on the phase of the design flow and the fidelity of the models. In this research, we use design space exploration of low-power adders as a case study for comparative analysis of two estimation flows: Physical layout Aware Synthesis (*PAS*) and Place and Route (*PnR*). We study and compare post-PAS and post-PnR estimations of the metrics of interest and the impact of various design parameters and input switching activity factor (α_I). Adders are particularly interesting for this study because they are fundamental microprocessor units, and their design involves many parameters that create a vast design space. We show cases when the post-PAS and post-PnR estimations could lead to different design decisions, especially from a low-power designer point of view. Our experiments reveal that post-PAS results underestimate the side-effects of clock-gating, pipelining, and extensive timing optimizations compared to post-PnR results. We also observe that PnR estimation flow sometimes reports counterintuitive results**

I. Introduction

Contemporary submicron technologies have enabled fabricating chips with billions of transistors. At the same time circuit models have become more complex to simulate. With each new generation of process technology, novel and improved methods are necessary for the estimation of different metrics of interest for the design process, such as area, timing, and power. Although it is not possible to measure precisely the metrics of interest of a design until it is fabricated, there are various estimation methods aimed for different design phases, which differ in levels of accuracy and estimation speed. The most accurate way to simulate a design is to use transistor-level post-Place and Route (**PnR**) data and SPICE. However, such detailed simulation is not possible in early phases of the design process. Moreover, it is impractical for a large number of test vectors and numerous design points as its computational complexity leads to an unaffordably long time frame to get the results. On the other hand, estimation processes done in early phases are less detailed which makes them faster. Yet, decreased accuracy of these processes may lead a designer to wrong conclusions.

Currently, the most widespread estimation flows work at the gate-level. In this study, we consider synthesis estimation flow that consists of synthesis and post-synthesis estimation, and PnR flow that consists of synthesis, PnR and post-PnR estimations. PnR models are more accurate than synthesized ones as they are closer to fabricated chips. Conversely, synthesis estimation flow deals with simpler models and requires less steps, which makes it faster but less accurate than PnR flow. Traditional synthesis tools use wire-load models based on fanouts which do not provide accurate wire delay information. Wiring delay cannot be ignored, and it even increases its importance with further technology scaling [1]. Physical layout Aware Synthesis (**PAS**) claims to overcome the drawbacks of traditional synthesis tools and to provide post-PAS results which are closer to post-PnR ones. The main advantage of PAS tools over the traditional synthesis flows is their floorplan awareness that provides more realistic interconnect modeling.

We focus our study of estimation flows on adders. Addition is one of the most used operations in both general-purpose and application-specific processors and the adder is often in the processor's critical path [2]. Low-power and energy-efficient design is important for highly utilized units like adders, which are often found to be in a thermal hot-spot of microprocessors [3] where energy-efficiency has become a first-order design constraint. Adder design involves different parameters, such as adder family and number of pipeline stages, that affect the metrics of interest: area, timing, and power. The combination of parameters creates a vast design space and in order to explore adders in that space and examine adequate trade-offs, we need fast and accurate estimation flows (methods) for these metrics.

In this study, we perform an extensive comparison of PnR and PAS estimation flows for 32-bit adders, generated mostly in an automated way using commercial tools. Specifically, we compare estimations in terms of area, timing, and power for several design parameters: clock cycle, adder family, number of pipeline stages, and clock-gating. Design space exploration of adders as a case study is chosen due to importance of adders and as a design space exploration with a large number of design points allows observing advantages and drawbacks of estimation flows in a comprehensive way. Since power has become the primary design constraint for the majority of computer systems, we pay special attention to power dissipation. In contrast to area and timing estimation of a VLSI design that does not require knowledge of the inputs, we cannot have a solid total power estimation without information about the inputs. Therefore, we evaluate how the power numbers obtained in post-PAS and post-PnR estimations vary with input switching activity factor (α_I). To the best of our knowledge such a study has never been addressed in open literature before. Moreover, we observe how important the differences between post-PAS and post-PnR power results are when α_I is extracted from application benchmarks (SPEC CPU2006). In our experiments, we use a low-power 40nm technology (TSMC40LP).

In the context of adder characteristic evaluation, there is considerable prior research [2], [4]–[8]. Our research differs in that we perform evaluation of adders from the estimation flow perspective. Additionally, some of the previous research efforts aim to connect statistics of the input signals with power dissipation [9], [10]; however, their goals are different as they do not include design space exploration nor a comparison of estimation flows.

This research presents the following contributions:

- We study how adder area, timing and power differ with respect to the applied estimation flow. We apply PAS and PnR estimation flows for various design parameters and input switching activity factor (α_I). We examine and

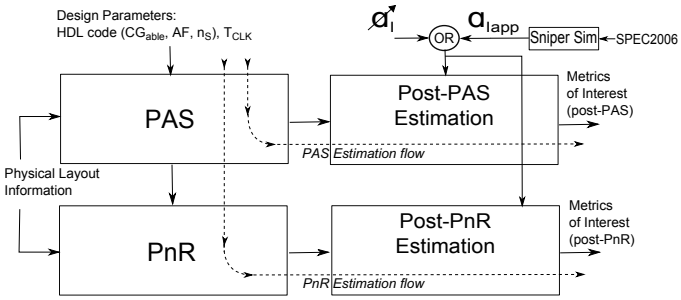

Fig. 1. Block diagram of the framework used to perform the comparative analysis of PnR and PAS estimation flows in terms of area, timing and power (metrics of interest).

explain in which cases post-PAS results are (or are not) reliable, compared to post-PnR (Section III).

- Among other observations, we establish that in terms of timing, post-PAS results fairly underestimate overheads of pipelining and clock-gating due to idealistic clock tree modeling. We also observe that PnR estimation flow sometimes reports counterintuitive results (Section III-B).
- Regarding power dissipation, we observe significant differences, especially for high frequency and high α_I. According to post-PnR estimations carry-lookahead has the lowest power of the studied adders for higher frequencies, while Brent-Kung adder structure performs better according to PAS. Additionally, our results reveal that post-PnR power estimations increase more with α_I than post-PAS ones (Section III-C).

II. Methodology

This section describes the methodology, framework, and flow parameters used to perform this extensive comparison of PnR and PAS estimation flows. By flow parameters we assume design parameters and α_I (explained in Sections II-B and II-C respectively).

A. Framework

A simplified block diagram of the framework is depicted on Figure 1. As a PAS tool and post-PAS results estimator we use Cadence RTL compiler in Physical Layout Estimation mode while for PnR and post-PnR estimations we use Cadence Encounter Digital Implementation System. We developed several scripts for automation and interfacing between tools. PAS is the first common part of both PAS and PnR estimation flows. Each design point is defined by four parameters, *AF*, n_S, CG_{able} and T_{CLK}, which are described in Section II-B. We wrote all the HDL code by hand for the adders of each valid combination of the *AF*, n_S, and CG_{able} parameters. For each design point, we supply the HDL code of the adder together with a target clock cycle (the T_{CLK} parameter) to the PAS tool to produce adder's mapped netlists. We perform PAS for all combinations of our design parameters in an automated way. The PAS tool tries to meet timing constraints (T_{CLK}) while prioritizing power over area optimization. Once we have all the netlists produced using PAS, we perform post-PAS estimation of the metrics of interest (area, timing and power), and complete the PAS estimation flow. Both post-PAS and post-PnR α_I-based power estimations are explained in Section II-C. In order to complete the PnR estimation flow we further process our designs and perform PnR. The last stage of the PnR estimation flow is post-PnR estimation of the metrics of interest.

We perform PAS and PnR under the same conditions. All designs are implemented using TSMC40LP library for *typical* operating conditions. PAS tool requires basic physical layout information in order to perform its physical layout aware synthesis. This is a subset of the complete set of physical layout parameters required by the PnR tool. For example, core density is a fundamental and basic layout parameter required by both tools. An initial core density of 70% is selected as a sensible balance between timing improvement and shrinking area for the wide set of adder designs parameters that we use. We experimentally found that, in PnR stage in general, density below 70% gives negligible faster timing for a non-negligible area overhead while densities higher than 70% spoil timings noticeably as the tool suffers from lack of free space for optimizations and routing. Additionally, the initial densities below 70% even cause DRC errors and density violations in some cases. Regarding another fundamental parameter used by both flows, number of routing layers, we use 4 since we experimentally found that using more would be a waste of resources as the additional layers do not improve the results.

The optimization techniques that are applied in the PAS stage are: adding buffers, resizing gates, remapping logic, swapping pins, deleting buffers, and boundary optimizations (removing undriven or unloaded logic connected, propagating constants, collapsing equal pins, and rewiring of equivalent signals across hierarchy). However, we prevent restructuring optimization techniques that could change the adder structure, in order to maintain the original topology of each adder family. The optimization techniques done in the PAS stage are applied again in PnR since there is more optimization space in PnR than in PAS. Additionally, two more techniques are applied in PnR: moving instances and applying useful skew. PAS optimizations are run with high effort, while PnR ones with medium effort rather than high in order to keep PnR runtime affordable for a large number of design points. More details about the optimizations are available in the documentation of the aforementioned Cadence tools.

The necessary times for the complete PAS and PnR estimation flow for a single design are around 70 and 400 seconds in average, respectively. Thus, PAS is around 5.7 times faster than PnR estimation flow. The time needed to perform PnR is fairly reasonable for a small number of designs. However, the difference in execution time of the estimation flows becomes an important issue when we have experiments with a large number of design points. For example, in this design space exploration the number of design points is over 2000, which puts the difference between times needed for the flows completion into days. Moreover, for larger and faster designs this ratio increases as the PnR tool makes more effort on routing and optimizations. Therefore, we can expect the difference between these flows to be higher for large and more complex designs (e.g. accelerator or processor's core).

The framework is built for adders, but it can be easily accommodated for any other kind of arithmetic block. For example, in order to adapt the current framework to perform the same kind of comparative analysis for multipliers, practically the only change would be to supply the HDL code of the multipliers instead of current HDL code. To make it fully compatible with this framework, the new HDL code should, of course, incorporate the same design parameters (*AF*, n_S, and CG_{able}).

B. Design Parameters

In this section we present design parameters used in this comparative analysis of estimation flows:

- $\boldsymbol{CG_{able}}$ indicates whether an adder is implemented with or without support for clock-gating (*CG* or *noCG*, respec-

tively). Clock-gating prevents useless switching activity in circuits and makes pipelining more effective. It is done at the stage level, i.e. each stage (register) is clock-gated independently. We implement clock-gating in the HDL code and use clock-gating cells from TSMC40LP library.

- ***AF*** is the adder family (algorithm). We choose five adder families: Brent-Kung (*bk*), Kogge-Stone (*ks*), carry-lookahead (*cla*), ripple-carry adder (*rca*), and conditional sum adder (*cosa*) [11], all implemented in HDL code. We choose *rca* as it is a basic algorithm, *cla* as a basic model of fast adders, and parallel prefix (*bk* and *ks*) and *cosa* adders as they are fast and suitable for pipelining.
- $\boldsymbol{T_{CLK} = (1/f)}$ is the clock period of the adder, while f is its clock frequency. We define 0.1-5.0 GHz as a practical frequency range in order to explore a vast design space.
- $\boldsymbol{n_S}$ indicates the number of pipeline stages in the addeer (pipeline depth), and it ranges from 1 to 8. We only increase n_S for a particular AF if it provides shorter T_{CLK} with PAS. In particular, we implement *bk*, *ks*, *rca* for 1-8, *cosa* for 1-7, and *cla* for 1-4 stages. Since the adders are pipelined and have a clocked input register (first pipeline stage) they can be easily incorporated into a datapath. The input register of an *n-stage* adder consists of 2x32 DFlipFlops (*DFF*) for the input operands and one DFF for the carry-in bit, and in case of *CG* we have *n* additional DFFs for the *ClockEnable* signal. Additionally, each *n-stage* adder has *n-1* pipeline clocked registers (consisting of DFFs) which are used to save intermediate results. The size of each register depends on the AF structure and the place inside the adder where the register is inserted (i.e. on the way the adder is pipelined). We include in this study high numbers of pipeline stages because we need high frequencies for vast design space exploration, and as we want to study the impact of high pipelining.

C. Power Estimation

Power estimation is specially analyzed as it depends on data inputs. The total power of a VLSI circuit is the sum of dynamic (P_{dyn}) and static (P_{stat}) power:

$$P_{tot} = P_{dyn} + P_{stat} \tag{1}$$

For arithmetic circuits P_{dyn} is typically dominant. It consists of switching (P_{switch}), short-circuit (P_{sc}) and glitching (P_{glitch}) power:

$$P_{dyn} = P_{switch} + P_{sc} + P_{glitch} \tag{2}$$

From [12]–[14] we have:

$$P_{switch} = \alpha f C_L V_{DD}^2 \tag{3}$$

$$P_{sc} \propto \alpha f V_{DD}^3 / C_L \tag{4}$$

$$P_{glitch} \propto \alpha f V_{DD}^2 \tag{5}$$

where α, f, V_{DD}, and C_L are the switching activity factor, clock frequency, voltage supply, and load capacitance of a circuit node respectively. Therefore, all P_{dyn} components are proportional to α. The probability of changing the state from 0 to 1 at a given clock tick for circuit input and node is denoted α_I and α respectively. α is a function of the circuit topology and aforementioned α_I. A probability of 0 means there is no activity (or negligible activity), 0.5 means signal changes its state every cycle while 1 means the signal has the same frequency as the reference clock.

However, for power estimation purposes, both PAS and PnR tools divide total power into net, internal, and leakage power. Net power is the power dissipated when charging or discharging the capacitive load, which includes the net capacitance and the capacitance of the input pins, and is calculated using Formula 3. The internal power consists of two components: (*i*) the power dissipated in an instantaneous short-circuit connection between the voltage supply and the ground when the gate transitions (P_{sc}) and (*ii*) the internal net power dissipated during charging or discharging internal capacitance (P_{switch}). It is calculated by using internal power tables provided in the library, which are generated as a result of SPICE simulations for a range of input slew rates and external loadings, the effective frequency of a given circuit input (α_I / T_{CLK}), and the probability that the input signal is high. P_{glitch} is modeled inherently by propagating switching activity through the circuit. The leakage power computation depends on the state of the gate, and is calculated using the power tables provided in the library.

We perform post-PAS and post-PnR power estimation by providing the outcome of PAS and PnR stages to post-PAS and post-PnR estimators respectively and changing the α_I of the input signals of the adders. This statistically-based power estimation method is suitable in cases in which a large number of design parameters are used, i.e. a large number of design points. We perform power experiments in two ways:

- By varying α_I (shown as $\not\alpha_I$ in Figure 1) from 0 to 0.5, with increments of 0.025. Since we study pipelined adders with an input register on its inputs, the maximum possible α_I of our input signals is 0.5. We assign the input signals the same probability (p) of being '1' or '0', as we assume they have independent and uniformly distributed random values.
- By assigning the input ports of the adder utilizing signal statistics ($\alpha_I = \alpha_{Iapp}$ and p) obtained using Sniper simulator [15]. Using Sniper we collect the data from input registers of the adder and information about adder usage. We further process these data in order to calculate α_I (α_{Iapp}) and p of each input port of the adder separately (*A[31:0], B[31:0], CarryIn*, and *ClockEnable* signals). We run Sniper for four SPEC CPU2006 benchmarks: mcf, bzip2, libquantum, and h264ref. Sniper implements an Intel's x86 microarchitecture, and we configure it to use a single adder in our experiments.

III. Results

A. Area

Table I shows the statistics of the ratio of area for all designs obtained using PnR and PAS. Area estimated using PnR and PAS estimation flows is compared for all combinations of the design parameters. For both flows, the effective area occupied by cells is considered. The geometrical mean of the ratios is high (1.64). The main reason is that clock tree synthesis with PnR is more realistic (but still imperfect) so with all its buffers and clock routing, it occupies more area than in PAS case. Clock tree in *CG* designs is more complicated, and they suffer from higher area increases than *noCG* ones. We observe, for a given *AF* and T_{CLK}, the ratio decreases with n_S. This happens as the PnR tool does more effort in timing closure (that occupies extra area) for adders with fewer stages as they are inherently slower. Regarding *AF*, *bk* has the highest ratio,

TABLE I. PNR VS. PAS RATIO FOR AREA, TIMING, AND POWER

	Area			Timing			Power		
	All	*noCG*	*CG*	All	*noCG*	*CG*	All	*noCG*	*CG*
Min	1.43	1.43	1.45	0.84	0.84	0.88	0.49	0.56	0.49
Max	1.94	1.73	1.94	2.07	1.38	2.07	7.52	2.07	7.52
GEOM	1.63	1.58	1.69	1.18	1.02	1.38	1.12	1.02	1.27
GStDev	1.06	1.04	1.06	1.25	1.12	1.21	1.35	1.21	1.45

TABLE II. RATIOS OF MINIMAL ACHIEVABLE CLOCK PERIODS FOR PNR AND PAS (T_{minPnR}/T_{minPAS}). IN THE UPPER PART OF THE TABLE ARE PRESENTED RESULTS WITHOUT CLOCK-GATING, WHILE IN THE BOTTOM ARE THE RESULTS WITH CLOCK-GATING SUPPORT.

AF \ n_S	1	2	3	4	5	6	7	8	
bk	1	0.97	0.99	0.94	1	1.06	1.12	1.19	
ks	0.97	0.90	0.92	0.94	0.97	1.06	1.17	1.38	
cla	0.96	0.98	0.90	0.84	N/A	N/A	N/A	N/A	*noCG*
cosa	1	1.11	0.97	0.99	1.13	1.22	1.36	N/A	
rca	0.99	1.02	0.92	1.01	0.89	0.91	0.9	1.11	
bk	1.29	1.31	1.36	1.28	1.37	1.56	1.76	2.06	
ks	1.47	1.30	1.52	1.31	1.35	1.56	1.76	1.93	
cla	1.40	1.22	0.88	1.02	N/A	N/A	N/A	N/A	*CG*
cosa	1.11	1.26	1.07	1.38	1.58	1.64	2.07	N/A	
rca	1.05	1.15	1.17	1.46	1.13	1.22	1.24	1.71	

TABLE III. n_S OF THE FASTEST DESIGN PER *AF* FOR PNR. THE MAXIMUM n_S FOR A GIVEN *AF* IS IN PARENTHESIS

CG_{able} \ *AF*	*bk*	*ks*	*cla*	*cosa*	*rca*
noCG	7 (8)	5 (8)	4 (4)	5 (7)	7 (8)
CG	5 (8)	5 (8)	4 (4)	3 (7)	7 (8)

as due to its high max-fanout, additional circuit sizing is done in PnR. For any given T_{CLK} and n_S, *AF*s have the same relative area order in both flows: *Area(cosa)>Area(ks)>Area(bk)>Area(cla)>Area(rca)*.

B. Timing

Table I provides statistics of ratios of T_{CLK} for all designs. In the rest of this section we focus on the minimal achievable clock period (T_{min}) for a given n_S for all *AF* according to post-PAS (T_{minPnR}) and post-PnR (T_{minPAS}) timing results. In other words, T_{min} represents the critical path of the fastest achievable adder structure for given n_S and *AF*. The comparison is shown on Figures 2(a) and 2(b) for *noCG* and *CG* respectively. Additionally, Figure 2 indicates the clock network latency (t_{CNL}) of post-PnR adders. In PAS, for timing analysis purpose, the clock tree is not modeled so clock network latency is not considered properly. Therefore we present clock latency only for post-PnR designs. The ratios of T_{min} for PnR and PAS (T_{minPnR}/T_{minPAS}) are presented in Table II. A ratio below 1 means that for the same n_S and *AF*, PnR adder is able to achieve higher speed (lower clock period) than PAS adder.

From Figure 2(a) we notice that post-PnR estimations report faster timings for designs without clock-gating support (*noCG* case) in most cases. However, both post-PnR and post-PAS provide very similar estimations on average. Except for highly pipelined designs, the impact in overall timing of clock network latency in post-PnR is not very high. This is because the PnR tool is able to overcome the clock tree network latency by applying optimization techniques. As a conclusion, timing results obtained with PAS are reliable in most cases for designs without clock-gating (*noCG*).

The impact of the clock tree for *CG* (Figure 2(b)) is much higher since the clock tree is significantly larger than in designs without clock-gating. This results in an increase of clock network latency which causes timing degradation since with high clock network latency, register-to-output critical paths become hard to satisfy. The final result is that post-PnR estimations report consistently slower

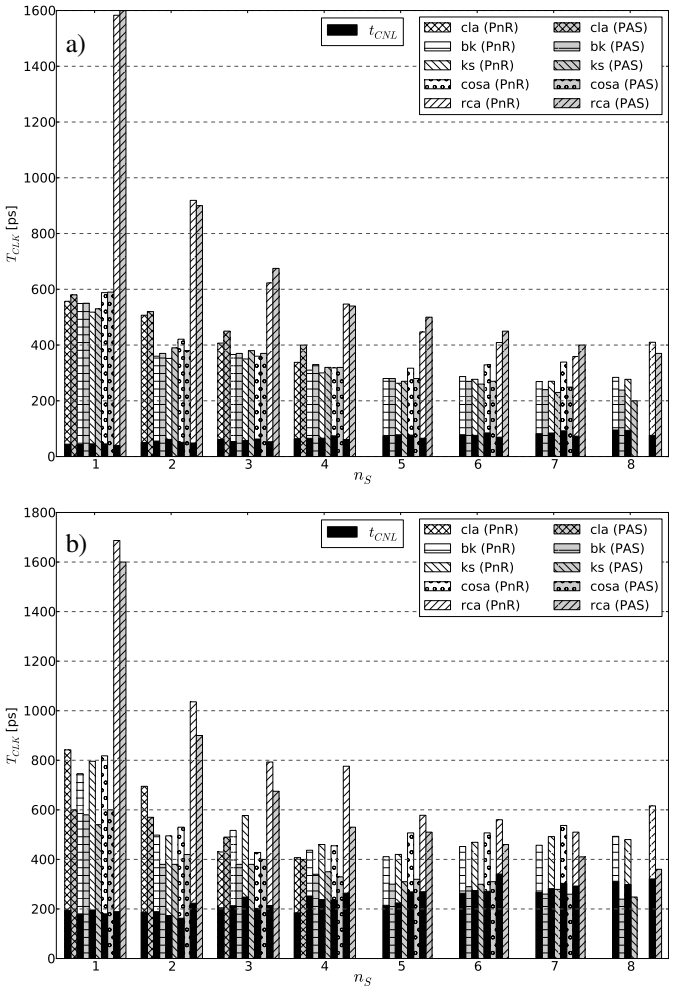

Fig. 2. PnR vs. PAS: minimal achievable clock periods (T_{min}) for given n_S and *AF*, for (a) *noCG* and (b) *CG*. Black part of PnR bars indicates the clock network latency (t_{CNL}).

timings than post-PAS estimations. Therefore, we can say that PAS flow underestimates the negative effect of including clock-gating logic. We also notice that the ratio (T_{minPnR}/T_{minPAS}) is higher when n_S increases in case of high area *AFs* (*bk, ks, cosa*). For low area *AFs* (*cla, rca*) this trend does not exist but still the 8-stage design (*rca*) has the highest ratio. Additionally we observe that, according to post-PnR estimations, *cla* performs faster for *CG* than for *noCG* compared to other *AF*. Due to its small area and simple structure, the delays introduced with routing and clock tree synthesis are smaller than for high area parallel adder structures.

From Figure 2 we observe that for PnR, there are cases in which designs with fewer stages are faster than designs with more stages. For example, the 2-stage *bk* adder with *CG* can achieve shorter T_{CLK} than the 3-stage one. These counterintuitive results happen due to two reasons. The first and the main reason is that with more pipeline stages, the clock tree is larger, so clock network latency is higher. In this case, the clock network latency increase is higher than timing shortening achieved with additional stages. The second, less dominant, but also less obvious reason can be observed in the case of *noCG* 2- and 3-stage *bk* PnR adders. The combinational path delay ($T_{PnR}-t_{CNL}$) in the 2-stage case is shorter than in the 3-stage case (even if we allow more space for clock and signal routing). Most likely, the reason why the PnR tool routes less efficiently the 3-stage case is that it gets stuck in a local minima. Finding optimal routing

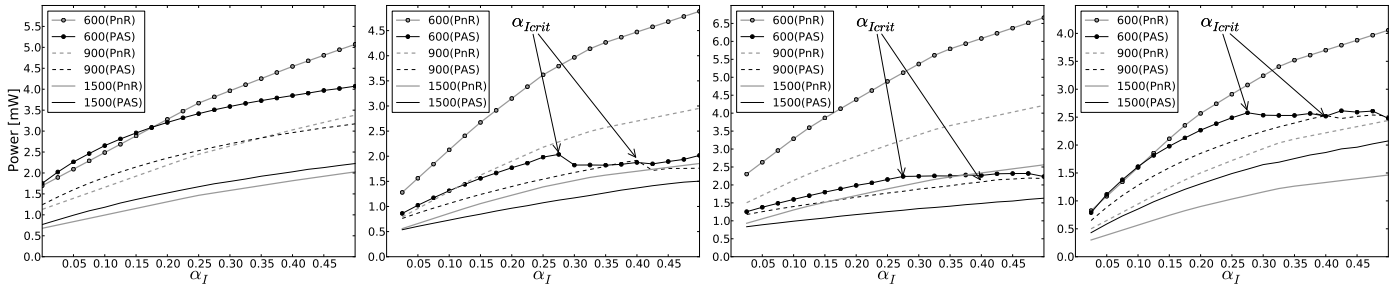

Fig. 3. (α_I,P) graph for post-PnR and post-PAS estimations. From left to right: (a) *noCG, ks, 3-stage*, (b) *CG, cla, 4-stage*, (c) *CG, rca, 6-stage* and (d) *CG, cosa, 2-stage* adders.

for a given placement is an NP-complete problem, so tools typically have to resort to use heuristics, which cannot always guarantee to find the optimal solution. The mentioned counterintuitive result is not an isolated case. For example, lower clock tree latency for a higher n_S, also can be seen in 4- and 5-stage *CG, bk* adders. We suppose this is a consequence of the irregularity of adder structures, and that for regular structures like caches the results would be more expectable. Reducing clock network latency can be achieved by manual clock-tree optimizations, but for a design space exploration it is not practical to optimize by hand a large number of design points. If we optimized by hand only some of them, the comparison would not be fair.

We observe two important consequences of aforementioned facts. The first one is that the fastest adders produced using PnR and PAS flows have fairly different design parameters. The fastest adders are 5-stage *ks* (263ps, PnR) and 8-stage *ks* (200ps, PAS) for *noCG*, while 4-stage *cla* (407ps, PnR) and 8-stage *bk* (240ps, PAS) for *CG*. Accordingly, when clock-gating and PnR are assumed, *cla* outperforms parallel-prefix AF (*bk* and *ks*), that are traditionally known to be fast. The second observation is that according to post-PnR estimations, in terms of T_{min} there is an optimal point beyond which pipelining stops to give the desired speedup but rather slowdowns for all *AF* except *cla*. On the contrary, in post-PAS estimations more pipeline stages always result in shorter T_{CLK}. Table III shows the optimal number of stages per *AF*, for *noCG* and *CG*.

C. Power

Since adders' topologies produce high α, and we implement them using a low leakage technology, P_{stat} is negligible. We observe from the results that it is two orders of magnitude less than P_{dyn}, even for *CG*. Therefore, the power of an adder (P) is practically equal to its dynamic component ($P \approx P_{dyn}$). Furthermore, P consists of two parts: power of adder's clock tree and of the rest of the adder. Due to lack of space we only present the aggregated power P. An important observation is that post-PAS estimation does not model clock tree power accurately as in post-PAS results it does not depend on α_I for *CG*.

Table I shows the statistics of the ratio of power for designs generated for all combinations of design parameters and α_I obtained using PnR and PAS flows. Regarding *AF*, we observe that post-PAS power estimations are the most similar to the post-PnR ones for *rca* (simple topology) while the highest deviations are found for parallel prefix adder structures. For *CG*, we exclude the $\alpha_I = 0$ case from the statistics and further analysis as it is a special case when clock-gating logic is always enabled. Results for this case are fairly different and the ratios range from 0.4 to 15.4. Therefore, we can deduce that this case is obviously modeled very differently in post-PnR and post-PAS estimations.

In order to examine how post-PAS and post-PnR power change with α_I, we present (α_I,P) graphs for different T_{CLK} (Figure 3). As a representative *noCG* case, *ks, 3-stage* adder is selected (Figure 3(a)). The graph shows that post-PAS overestimates power dissipation, comparing to post-PnR results, for low α_I. For high α_I this remains true for longer T_{CLK} designs while for designs with shorter T_{CLK} post-PAS power results are lower than post-PnR. We observe that, for higher n_S, the post-PnR and post-PAS power ratio is higher. The trends are similar for all *AF* except for *rca*. In this case post-PAS power is typically lower than post-PnR power.

Figures 3(b)-(d) show graphs for three representative *CG* cases. For *CG*, the post-PAS power estimations are usually below post-PnR ones. The exceptions are designs for low n_S and long T_{CLK} where post-PAS results are typically above post-PnR ones (Figure 3(d)). As can be expected, post-PnR power estimations increase with α_I. On the contrary, for post-PAS, we observe that there is a critical α_I (α_{Icrit}) from which estimation starts decreasing and contradicting the formulas for power dissipation (Formulas 3-5). This α_{Icrit} does not depend on *AF* but increases with T_{CLK}. It starts at 0.175 for the shortest T_{CLK} (around 400ps), and it disappears when T_{CLK} gets around 1200ps. The trend is the same for other *AF*, but for *rca* it is much less noticeable as it can be seen from Figure 3(c). For *CG*, n_S has the same impact on post-PnR and post-PAS power estimations as for *noCG*. The unexpected effect of α_I ($\alpha_I > \alpha_{Icrit}$) is related to the way how the post-PAS tool estimates clock-gating power reduction. Since we do not have access to the source code of the tools, we are not able to further investigate the problem.

From a low-power processor's designer point of view, the most relevant adders are those that have the lowest power for a given frequency. Figure 4(a) shows two graphs, (f,P) and (f,n_S), of the lowest power adder design for a given frequency for both post-PnR and post-PAS. The graph shows the *AF* and the n_S of the PnR adders and the *AF* of PAS adders. Bar colors indicate whether the PnR and PAS adders have the same n_S. The graph includes only *CG*, as we target low power, while $\alpha_I = 0.5$ is chosen, as it is the worst case from low-power design point of view. We can observe from the graph that post-PAS power drops for high T_{CLK}, and this is a consequence from the aforementioned effect of α_{Icrit}. For f below 0.5 GHz according to both post-PnR and post-PAS estimations *rca* is selected. In the next range (0.5-1 GHz) in post-PnR power estimations, *cla* has the lowest power, while in post-PAS case it varies between *cla* and *bk*. For medium f range (1-2 GHz) in both cases *bk* is typically the lowest power family. *Bk* continues to be the lowest power *AF* over 2 GHz for post-PAS, while for post-PnR the most power efficient *AF* are typically *cosa* and *cla*. Regarding n_S, it is different for PAS and PnR for around 30% of f range, and when it is different, usually PnR adders have a higher n_S as follows from Section III-B.

Figure 4(b) shows the results for α_I extracted from the bench-

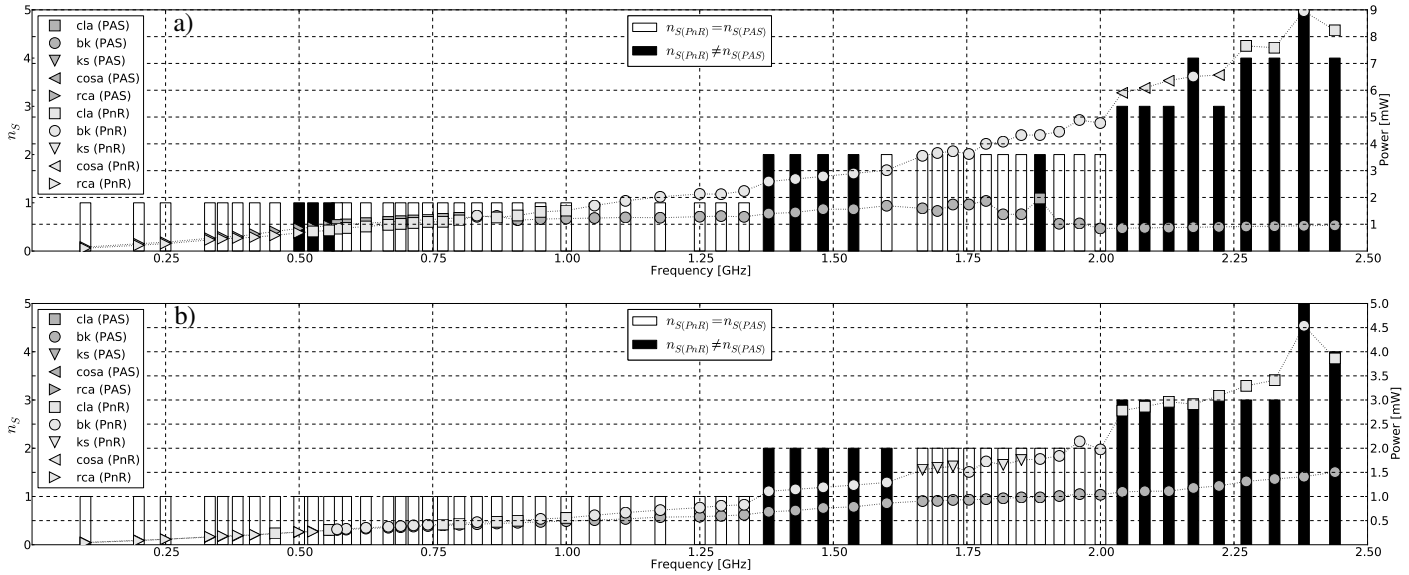

Fig. 4. Post-PnR and post-PnR power estimations graph of the lowest power adder designs with clock-gating support (*noCG*) for given frequency, for (a) $\alpha_I = 0.5$, and (b) $\alpha_I = \alpha_{Iapp}$ extracted from the benchmarks. Bars represent n_S of lowest power PnR adder, while their color indicate whether the PnR and PAS adders have the same n_S.

marks ($\alpha_I = \alpha_{Iapp}$). There are important differences compared to the experiment with $\alpha_I = 0.5$. Due to lack of space we do not present results for α_I of each benchmark separately but we present results for averaged α_{Iavg}. In this averaged case, we can notice that post-PAS power drops, observed for $\alpha_I = 0.5$, are not visible for $\alpha_I = \alpha_{Iavg}$. This means that α_{Iavg} of the input bits does not go above α_{Icrit}. Still, when benchmarks for which the data supplied on adders have high α_I ($\alpha_I > \alpha_{Icrit}$) are considered (e.g. bzip2), the aforementioned post-PAS power drops are observable. Although for α_{Iavg} there are not post-PAS power drops as for $\alpha_I = 0.5$, post-PAS power estimations are still significantly below post-PnR ones for high frequencies (around 3 times). Additionally, the lowest power adders according to post-PnR and post-PAS are fairly different. The most interesting observation is that *cla* is the lowest power *AF* for high frequency range (> 2 GHz) according to post-PnR. As it is presented in Section III-B, *cla* is able to achieve high frequencies with less n_S than other *AF* for *CG*, being in that way more power-efficient. The lowest power designs for post-PAS are similar in both experiments. Therefore, we can observe that PAS estimation flow is less dependent on α_I.

IV. Conclusion

In this research, we examine and analyze the differences between results obtained using PnR and PAS estimations flows. As a case study we use design space exploration of low-power adders that include various design parameters and α_I. Regarding the timing and power, we found that differences between post-PAS and post-PnR flows are significant for *CG* and high n_S. Nevertheless, for *noCG* and low n_S the differences are not substantial. The results of two observed estimation flows are the most similar for the simple topologies like *rca*. According to post-PnR estimations, *cla* often perform better, in terms of power and timing, than parallel prefix adders (*bk* and *ks*), which typically perform the best in post-PAS estimations. The main reason for the differences is that clock tree is modeled more realistically in PnR than in PAS. Additionally, we observe that PAS estimation flow provides fairly unreliable power results for high α_I, compared to post-PnR estimations. The reason is that post-PnR power estimations increase more with α_I than post-PAS ones. The differences between power estimations are especially high for *CG*, where post-PAS power is typically less than post-PnR. Moreover, post-PAS power estimations can decrease for $\alpha_I > \alpha_{Icrit}$, contradicting the formulas for power. We conclude that for most designs, the ratio of post-PnR and post-PAS power estimations is proportional to α_I and n_S, while it is inversely proportional to T_{CLK}. We also find that the PnR tool sometimes synthesizes the clock tree in a counterintuitive way. Additionally, we observe that post-PAS always underestimates area compared to post-PnR. The difference is 39% on average.

This work has been partially funded by the Spanish Government (TIN2012-34557 and FPU12/06157 grant).

References

[1] Z. Huang and M. D. Ercegovac, "Effect of wire delay on the design of prefix adders in deep-submicron technology," in *ACSSC*, 2000.

[2] C. Zhou, B. Fleischer, M. Gschwind, and R. Puri, "64-bit prefix adders: Power-efficient topologies and design solutions," in *CICC*, sept. 2009.

[3] M. R. Stan, K. Skadron, M. Barcella, W. Huang, K. Sankaranarayanan, and S. Velusamy, "Hotspot: A dynamic compact thermal model at the processor-architecture level," *Microelectronics Journal*, vol. 34, no. 12, 2003.

[4] R. Zimmermann, "Binary adder architectures for cell-based vlsi and their synthesis," 1997, *PhD Thesis*, ETH Zurich.

[5] A. Åslund, O. Gustafsson, H. Ohlsson, and L. Wanhammar, "Power analysis of high throughput pipelined carry-propagation adders," in *Norchip*, 2004.

[6] S. Sun and C. Sechen, "Post-layout comparison of high performance 64b static adders in energy-delay space," in *ICCD*, 2007.

[7] B. Zeydel, D. Baran, and V. Oklobdzija, "Energy-efficient design methodologies: High-performance vlsi adders," *IEEE SSC*, vol. 45, no. 6, 2010.

[8] I. Ratković, O. Palomar, M. Stanić, O. Unsal, A. Cristal, and M. Valero, "On the selection of adder unit in energy efficient vector processing," in *ISQED*, 2013.

[9] D. Baran, M. Aktan, H. Karimiyan, and V. Oklobdzija, "Exploration of switching activity behavior of addition algorithms," in *MWSCAS*, 2009.

[10] S. Alipour, B. Hidaji, and A. S. Pour, "Circuit level, static power, and logic level power analyses," in *EIT*, 2010.

[11] M. Ercegovac and T. Lang, *Digital Arithmetic*. MKP, 2003.

[12] H. J. Veendrick, "Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits," *Solid-State Circuits, IEEE Journal of*, 1984.

[13] M. Favalli *et al.*, "Analysis of glitch power disp. in CMOS ICs," in *ISLPED'95*.

[14] D. Rabe *et al.*, "Short circuit power consumption of glitches," in *ISPLED'96*.

[15] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sampled simulation of multi-threaded applications," in *ISPASS'13*, 2013, pp. 2–12.