# WP 2.0: Signoff-Quality Implementation and Validation of Energy-Efficient Clock-Less Wave Propagated Pipelining

Yehuda Kra
*Faculty of Engineering*
*Bar-Ilan University*
Ramat Gan, Israel
yehuda.kra@biu.ac.il

Tzachi Noy
*Faculty of Engineering*
*Bar-Ilan University*
Ramat Gan, Israel
tzachi.noy@biu.ac.il

Adam Teman
*Faculty of Engineering*
*Bar-Ilan University*
Ramat Gan, Israel
adam.teman@biu.ac.il

***Abstract*—The design of computational datapaths with the clockless wave-propagated pipelining (CWPP) approach is an area and energy-efficient alternative to traditional pipelined logic. Removal of the internal registers saves both area and the toggling power of these complex gates, while also simplifying the clock tree. However, this approach is rarely used in modern scaled technologies, due to the complexity of implementation and the lack of a robust, scalable, and automated design methodology that meets rigid industry standards. In this paper, we present WP 2.0, an extension of the original WP algorithm and automation utility, which demonstrated how to apply CWPP to any generic combinatorial circuit using a CMOS standard cell library. WP 2.0 advances this concept to provide full-flow implementation capabilities, providing a post-layout CWPP-ready design that meets signoff-quality industry timing requirements. The WP 2.0 utility interfaces with commercial design automation software for balancing a post-synthesis netlist to achieve a high CWPP launch rate (frequency). We demonstrate the calculation of an fused dot-product accumulation unit, implemented with a 65nm standard cell library, providing a worst-case launch rate that is comparable to a design implemented with a 3-stage clocked pipeline with a 12% area reduction and between 37%-54% power savings. Furhtermore, the CWPP design is equipped with unique post-silicon field configuration capabilities for optimizing operation and overcoming variation.**

***Index Terms*—Wave Propagation, Clock-less Wave Pipeline, High Throughput, Low Power, Timing Constraints, STA**

## I. Introduction

The continuous growth in demand for reliable integrated circuits throughout every aspect of the modern world has led to the development of universally accepted, robust digital design methodologies to ensure the functionality and yield of these products [1]. The standard digital flow almost invariably dictates the use of synchronized datapaths to eliminate races between signals, while clearly defining the timing relationship required to verify that these signals will propagate correctly under the specified operating conditions. In such a paradigm, performance (throughput) is increased by pipelining the logic paths, thereby shortening the longest paths and enabling a higher frequency. However, the addition of intermediate stages requires the insertion of sampling registers, which consume area and power and lead to a complex clock tree.

As an alternative to conventional clocked pipelines, the clockless wave propagated pipeline (CWPP) approach achieves an increase in throughput without the overhead introduced by the intermediate sampling registers [2]–[7]. This is achieved by balancing the delays through the network such that no race conditions (signal overlap) occur between subsequently launched signals as they propagate from the input to the output. This technique was developed and applied in the early days of computing [8]; however, in recent decades, it was essentially abandoned due to the difficulty in applying it to modern CMOS logic at nanoscaled technology nodes. Recent work [9] has proposed an algorithm for applying CWPP using a CMOS standard cell library. The algorithm was implemented in a custom tool and demonstrated under post-synthesis logic simulation.

In this paper, we extend the work of [9] to enable the full implementation of energy-efficient CWPP data paths to provide a signoff-quality design. The proposed approach was implemented in WP 2.0, a python-based utility that interacts with commercial electronic design automation (EDA) tools to balance a given combinatorial circuit, which meets the skew-based timing requirements of the CWPP approach. We show that standard static timing analysis (STA) can be used to verify and validate these timing constraints and ensure the circuit meets industry standard requirements prior to fabrication. To demonstrate the application and efficiency, a fused dot-product unit was mapped to a 65 nm standard cell library as an input to the tool and balanced to minimize skew. The resulting post-layout design was shown to achieve similar throughput to a three-stage pipeline with power savings of 37%–54% within a smaller silicon footprint. Furthermore, the design was equipped with a configuration mechanism to optimize the operation on-chip to potentially achieve further benefits for a given silicon sample and to overcome variation.

The rest of this paper is organized as follows: Section II introduces the WP 2.0 utility for balancing a combinatorial netlist. This is followed by Section III, which shows how the CWPP approach is validated using standard STA approaches. Section IV presents the implementation of a test case dot-product unit and compares it with a standard sequential implementation. Finally, Section V concludes the paper.

## II. Full-flow Balancing Implementation

Standard sequential clocking constrains the timing paths between startpoints (primary inputs or clocked data sources) and endpoints (primary outputs or clocked data sinks) to meet the target clock period and the setup/hold requirements of the

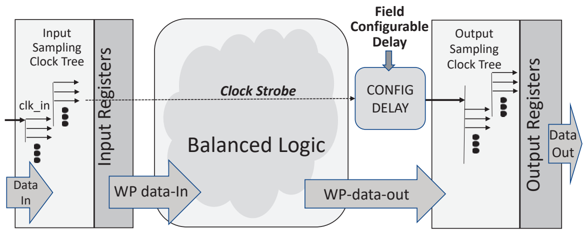


Fig. 1: Wave-Propagation design scheme

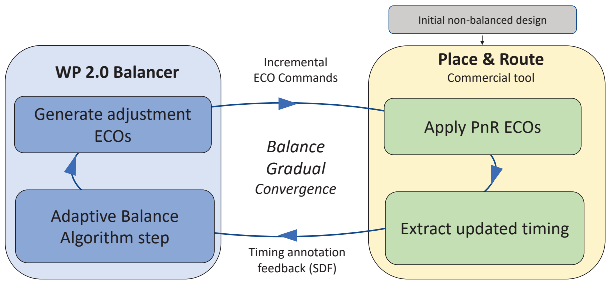


Fig. 2: WP 2.0 – a PnR integrated Balancing Utility.

clocked elements. CWPP, on the other hand, constrains the launch rate of the inputs, i.e., the frequency at which new data is injected into the network, according to the difference between the longest and shortest paths through the network [3]. That is, to ensure overlap-free propagation of data ("waves") through the network, it is sufficient to observe the longest and shortest paths to the endpoints ($t_{\text{logic}}(\max)$ and $t_{\text{logic}}(\min)$) and maintain [3]:

$$T_{\text{rate}} > t_{\text{logic}}(\max) - t_{\text{logic}}(\min) + t_{\text{setup}} + t_{\text{hold}}, \qquad (1)$$

where $T_{\text{rate}}$ is the minimum delay between subsequent data launched and $t_{\text{setup}}$ and $t_{\text{hold}}$ are the setup and hold constraints of the capture registers, respectively. Note that in a CWPP approach, as opposed to a standard clocked pipeline, the launch period is much shorter than the maximum delay ($T_{\text{rate}} \ll t_{\text{logic}}(\max)$), such that many separate inputs are simultaneously propagating through the network as waves.

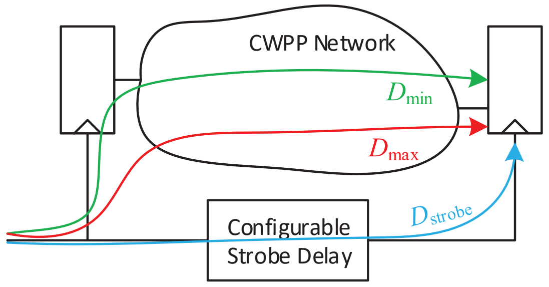
The work in [9] showed that an algorithmic approach to minimize the overall output skew ($t_{\text{logic}}(\max) - t_{\text{logic}}(\min)$) is to iterate over each gate in the network and minimize the skew of the signals through that gate. This approach was implemented in a utility called WP, which demonstrated how a given combinatorial logic netlist can be adapted for CWPP through such a balancing process. The tool was demonstrated on a synthesized logic netlist and various implementation challenges were raised and conceptually addressed, such as how to deal with delay dependence on input pattern, slew and load, clock skew, and process variation. However, since the work was limited to the post-synthesis stage, many of these problems were either negligible or left unaddressed.

The problem setup and balancing approach is illustrated in Fig. 1. The requirement is that the design consists of combinatorial logic, encapsulated between clocked startpoints and endpoints, which we will refer to as input and output registers for simplicity. This logic is provided as an unbalanced structural netlist that can be generated, for example, through logic synthesis. The most distinct difference shown in Fig. 1 from any generic design is the clocking of the output registers. Initially, a clock tree is generated to properly drive both the input and output registers. However, the root of output register clock tree, which we refer to as the "clock strobe", is connected to a net that branches out of the input clock. The clock strobe is treated as an additional input signal to the CWPP network, and thus is balanced with the logic signals that propagate through the network. The result is that the clock strobe arrives at the output registers at the same time that the (balanced) data arrives, providing a sampling edge that is independent of the launch frequency. A configurable delay block is added along the clock strobe to enable fine-grained adjustment of the location of the strobe clock edge to ensure robust sampling of the output within the timing window.

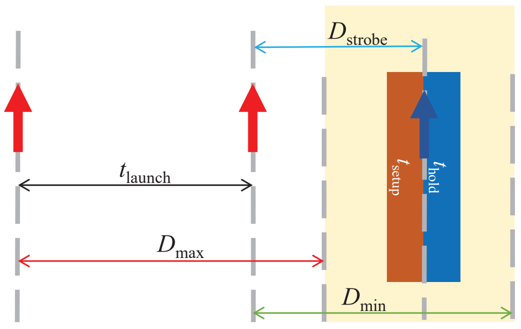
The skew balancing algorithm is implemented in WP 2.0, an extension of the original WP utility [9] that supports full-flow balancing to provide a post-layout implementation that meets industry standard signoff requirements. The operational concept of WP 2.0 is illustrated in Fig. 2. The balancing process starts with an initial non-balanced netlist[1] (output from synthesis or a structural design), including combinatorial logic encapsulated between a set of startpoints and endpoints, as described above. Commercial EDA tools are used to provide a post-layout design, including placement, application of the clocking scheme, and routing the design. Layout parasitics are extracted and used to calculate the gate and wire delays, which are annotated in the standard delay format (SDF). The timing arcs are fed into the balancing utility, which calculates the skew gaps for each cell input and generates corresponding engineering change order (ECO) commands to incrementally reduce the skew. The ECO commands, which preserve the majority of the existing placement and routing in order to maintain consistent timing between iterations, are activated within the place & route (PnR) tool and incremental placement, routing and timing extraction are applied. This process is repeated, gradually reducing the skew in order to adjust to the side effects and converge into a well-balanced design. Since convergence into a balanced design is performed incrementally, the typical overall run-time is reasonably in the order of 30 modern CPU minutes for the test case example described in Section IV.

The mechanism for generating adjustment ECOs is as follows. The utility starts its calculation at each startpoint with the arrival time initialized to the clock insertion delay to that register. It then advances down the connectivity graph – from startpoints to endpoints – individually balancing each gate along the path. This is done by calculating the arrival time at each of the gate inputs plus the timing arc from that input and slowing down the faster paths to equalize their delay with the slowest path through the gate. During early iterations,

[1]Note that for better results, the synthesizer can be driven towards balancing the design, but this is not required.



(a) STA Setup



(b) Valid timing window

Fig. 3: Timing parameters for CWPP STA.

larger delays are applied through buffer insertion and the tool only partially balances the skew, since the actual propagation delays will be significantly affected after incremental PnR due to modified slews, wire loads, etc. During later iterations, fine-grained tuning is applied, primarily by adding side-loaded capacitances.

## III. Standard STA sign-off approval flow

The WP 2.0 utility, described above, has the capability to provide a fully placed and routed digital design, ready for ASIC integration and fabrication. At a first glance, it would seem that the constraints that set the maximum launch frequency, previously shown in (1), are not the "textbook" timing constraints used for validating sequential designs (e.g., in (1), $T_{\text{rate}}$ is dependent on $t_{\text{logic}}(\text{min})$). However, a closer look at these constraints will show that, the standard set of timing checks is also applicable to CWPP, and therefore, standard STA can be used for signoff-quality design validation, required prior to fabrication. To elaborate upon this, we will consider a simplified model of a CWPP timing path from a single startpoint to a single endpoint, as illustrated in Fig. 3(a). For purpose of this discussion, we denote the maximum and minimum logic delays between these points as $D_{\text{max}}$ and $D_{\text{min}}$, respectively, and the delay of the strobe clock until it reaches the output register is denoted as $D_{\text{strobe}}$[2].

### *A. Timing constraints for Valid Timing Window Verification*

Fig. 3(b) shows a conceptual timing diagram of how the data propagates through the network. The red arrows represent two clock events at the clock source. The events are $t_{\text{launch}}$ apart. The calculation launched by the first event is valid at the output of the CWPP network, $D_{\text{max}}$ after the rising edge of the clock, and remains stable until $D_{\text{min}}$ after the following clock event, at which point the result cannot be reliably sampled anymore. The window for sampling is shaded yellow in the diagram, i.e., the destination sampling event with $t_{\text{setup}}$ and $t_{\text{hold}}$ margins must fit into the yellow area. The blue arrow represents the sampling event **at the destination**, $D_{\text{strobe}}$ after it was sourced. This is equivalent to standard sequential logic, where the sampling edge is one cycle after the launching edge; the only difference is that several additional launching edges will occur before the output registers sample the original input. In other words, the diagram is not to scale; generally for CWPP applications $D_{\text{max}}$ and $D_{\text{min}}$ are much larger than $t_{\text{launch}}$. This means there are multiple 'waves' propagating through the network simultaneously, and the clock strobe propagates along with the data delay to correctly sample within the valid window.

From Fig. 3(b), we can derive that the min-delay constraint, which requires that the data is stable at least $t_{\text{hold}}$ after the rising edge of the clock strobe, can be written as:

$$D_{\text{strobe}} + t_{\text{hold}} \leq D_{\text{min}}. \tag{2}$$

This is a standard early (hold) check, verifying the data launched by some clock edge does not interfere with data sampled by the same edge.

At the other side of the data valid window, the max-delay constraint requires that the data must be stable at least $t_{\text{setup}}$ before the clock rises. The data becomes valid and stable $D_{\text{max}}$ after the launch edge, which was once cycle ($t_{\text{launch}}$) before the sampling edge[3]. Accordingly, the max-delay constraint is:

$$D_{\text{max}} \leq t_{\text{launch}} + D_{\text{strobe}} - t_{\text{setup}}. \tag{3}$$

This is a standard late (setup) check, verifying that data launched by some clock edge propagates through the whole combinatorial logic and stabilizes before being sampled by the following edge.

Note that by isolating $t_{\text{launch}}$ in (3) and substituting $D_{\text{strobe}}$ according to (2), we arrive at the maximum launch rate constraint, previously shown in (1). Also note, that as with

[2]Note that $D_{\text{max}}$, $D_{\text{min}}$, and $D_{\text{strobe}}$ are calculated from the clock root, and therefore include the clock insertion delay and the clock-to-q delay of the input registers.

[3]Note that the capture edge rises one cycle ($t_{\text{launch}}$) after the launch edge. However, as mentioned previously, this rising edge (the clock strobe) only captures the data after a delayed propagation period, which is much larger than $t_{\text{launch}}$. Therefore, this can be seen as an extreme case of useful skew.

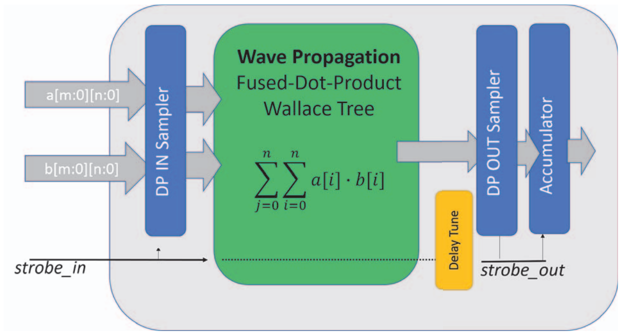


Fig. 4: Accumulated dot-product test case unit

standard sequential logic, the early constraint in (2) is independent of the clock period $t_{\text{launch}}$, while the late constraint in (3) can be met by reducing the launch rate.

## B. Variation Tolerance and Post-silicon Tuning

The standard STA checks, described above, are sufficient for signoff-quality timing validation. However, the non-conventional approach to design implementation may lead to concerns over tolerance to local variation due to both manufacturing and operating conditions. Remarkably, the CWPP concept not only addresses this, but actually provides a mechanism for overcoming such tolerance that is not available in standard sequential designs.

Returning to Fig. 3(b), CWPP requires that the clock strobe edge (blue arrow) must fall within the valid window (yellow shaded area), also taking into account the setup and hold constraints of the output registers. Therefore, it is sufficient to provide a timing window that is marginally larger than $t_{\text{setup}} + t_{\text{hold}}$, as long as the clock strobe can be steered into the middle of this window. Any additional width of the timing window (i.e., distance from the blue arrow to the edges of the yellow shaded area in Fig. 3(b)), provides margin for variation tolerance. Any extra variation tolerance for high yield (e.g., 3-sigma), can be quantified and defined as positive slack requirements in the STA checks.

However, a closer look at (2) and (3) shows that in the proposed implementation of Fig. 1, the CWPP approach has two free variables: $t_{\text{launch}}$ and $D_{\text{strobe}}$. This means that both the width of the valid window and the location of the clock strobe edge can be set externally. This is in contrast to standard sequential design, in which the frequency can be relaxed to meet the max-delay constraint, but the min-delay constraint must be fixed prior to fabrication for all manufactured samples. In other words, in the case of CWPP, both "setup" and "hold" violations can be fixed at the post-silicon stage.

This also raises another interesting point. In a standard sequential design, a "worst-case" (WC) corner is used to find the maximum operating frequency; usually assuming the slowest gates and the slowest operating conditions. Simultaneously, a "best-case" (BC) corner is defined to ensure that no min-delay violations occur[4]; usually assuming fast gates

[4]Note that due to their severity, min-delay violations are often checked at all defined corners.

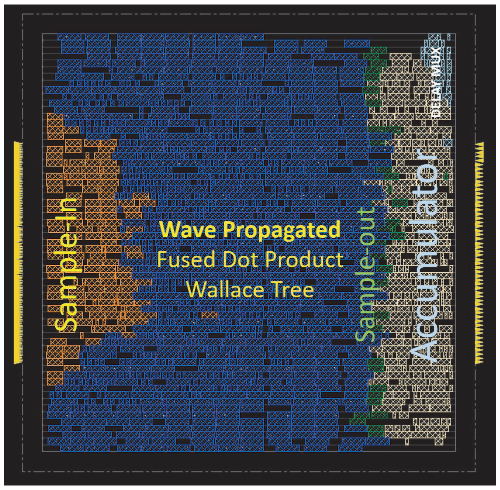


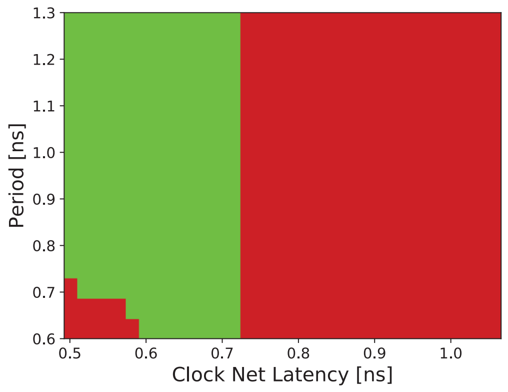
Fig. 5: Layout snapshot of test-case layout.

and the fastest perceivable operating conditions. However, in the case of CWPP with the ability to control the location of the clock strobe edge using the field-configurable delay unit, the maximum launch rate is set according to the width of the valid window. Since this window is particular for a given sample (i.e., one chip in one corner), the maximum launch rate should be estimated using a single corner, rather than checking max-delay at the WC corner and min-delay at the BC corner. Furthermore, a post-silicon binning operation should be applied to set the best launch-rate/delay unit configuration pair for each die – even providing on-chip sensors to adapt this for current operating conditions. While this is not a requirement, it provides a means to achieve better than worst-case estimations. It also means that as a design methodology, balancing could be applied for a typical corner, at which the majority of the chips are expected to be manufactured, rather than having to overdesign by always assuming the worst corner cases.

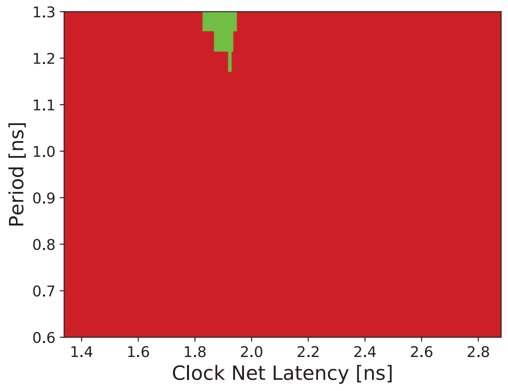
## IV. CWPP Test Case Implementation

Using the WP 2.0 utility, the CWPP approach can be applied to any combinatorial circuit that doesn't have feedback. However, it is especially effective when applied to non-conditional datapath calculations that require high throughput. Therefore, as a test case to demonstrate the feasibility and efficiency of CWPP realization using WP 2.0, we chose a dot-product unit, commonly found in systems such as digital signal processors (DSPs) and neural-network accelerators [10].

The test case design, which performs a vector-vector dot product multiplication of up to 1024 8-bit elements, is illustrated in Fig. 4. Each wave period, the unit performs a partial dot-product operation on 8 pairs of 8-bit numbers, such that for a launch clock period of $t_{\text{launch}} = 1\,\text{ns}$, an overall bandwidth of 8 G multiply-and-accumulate (MAC) operations per second is achieved. The design includes 128 input sampling registers, which drive a fused dot-product Wallace Tree combinatorial circuit whose output is sampled by 48 output registers and fed into a result accumulator. The design was mapped to a 65 nm standard cell library and fed into the WP 2.0 tool to



(a) Best-Case



(b) Worst-Case

Fig. 6: Shmoo plots of clock period vs. latency for BC and WC. BC can operate at much higher frequencies, and with a wider range of clock latency configurations compared to WC.

achieve a CWPP ready circuit. Placement, clock-tree synthesis, routing, parasitics extraction, STA, and power analysis were carried out with a Cadence flow (Innovus, QRC, Tempus, and Voltus, respectively), interfaced with and managed by the WP 2.0 utility. The resulting floorplan and standard cell placement of the balanced design is shown in Fig. 5. For reference and comparison, the design was also implemented with a standard clocked pipeline using the same EDA tools.

### A. *Validation of CWPP Operation*

The design was validated for correct operation through a standard signoff-quality STA flow, according to the methodology described in Section III. However, to demonstrate the validity of this methodology and provide extra confidence, a number of post-layout analyses were performed, by scanning a wide spectrum of launch rates and field-configurable delay unit settings.

Fig. 6 presents Shmoo plots for a range of launch rates and clock strobe latencies. One interesting implementation result that can be clearly seen in these figures is that the skew balancing is much more effective in the BC corner than in the WC corner. This leads to the BC corner having a significantly higher achievable launch rate with a wide valid window (i.e., many valid clock strobe delay settings) at frequencies well over 2 GHz. On the other hand, for the WC corner, the highest attainable frequency is just over 800 MHz. This discrepancy is not inherently clear in the CWPP approach, since the maximum frequency is set according to skew and not according to the propagation delay. However, when using non-balanced CMOS standard cells, the differences between the rising and falling arcs and the various paths through conditional cells are larger at the slow corner than the fast corner and these accumulate to larger skew over the network. In fact, for the implemented design, the skew at the fast corner was about 50% smaller than at the slow corner, despite using the slow corner timing data during WP 2.0 optimization.

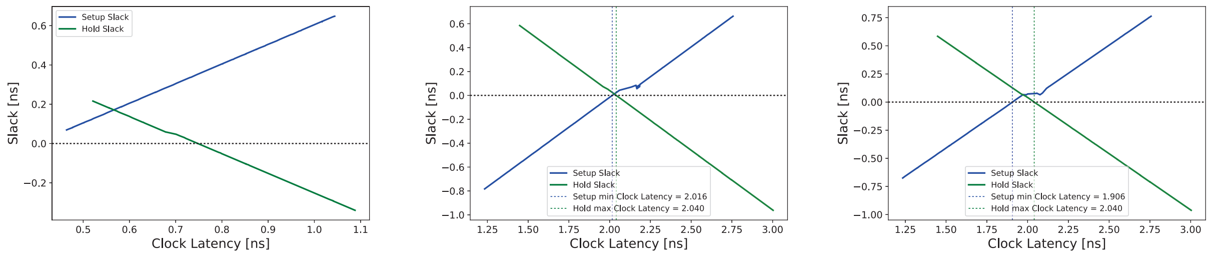
A closer look at the field-configurable delay capability is shown in Fig. 7. These plots show the timing slack reported by min-delay (hold) and max-delay (setup) checks on the post-layout CWPP design with a set launch rate as a function of the configurable delay setting. The requirement for valid CWPP operation at the specified frequency is positive slack for both setup and hold checks at at least one delay setting. For the best-case corner at 1.4 GHz (Fig. 7(a)), a large valid timing window exists and many different delay settings can be used. For the worst-case corner, a valid timing window can be achieved by reducing the frequency to 1 GHz (Fig. 7(b)), albeit with very little additional variation tolerance. By further reducing the frequency to 833 MHz, more margin is provided and several delay configurations can be used to locate the clock strobe within the valid window. Note that the inconsistencies that can be seen along the curves in Fig. 7 are primarily due to signal integrity (coupling noise) that can affect the propagation.

It should be noted that, in reality, most manufactured samples are well in-between the best and worst case assumption. Therefore, our capability to configure the clock strobe delay post-manufacturing is a great advantage, since unlike a conventional clocking scheme, not only setup violations can be fixed (by reducing the clock frequency), but normally unfixable hold violations, as well.

### B. *Comparison with Standard Clocked Design*

Table I compares the area composition of the CWPP design as compared to a 3-stage clocked-pipeline design, which achieves a similar throughput. The reference design has a 37% higher cell count than the CWPP test-case design, including an extra 381 internal sampling registers and 20 clock-tree nodes. This leads to a total area savings of 12%, since the balancing flow of the CWPP design requires lower initial floorplan utilization to allow incremental delay element insertion without significantly impacting the base placement and routing.

In order to compare the power consumption of the two designs, the timing arcs of the post-layout designs were back-annotated into the gatelevel simulation to extract toggling rates, and Cadence Voltus was used for vector-based power estimation. The designs were tested under input vectors with varying activity rates, ranging from 0–100% (100% activity referring to all inputs randomly changing every clock cycle).



(a) Best-Case corner with $t_{\text{launch}}$=0.7 ns (b) Worst-Case corner with $t_{\text{launch}}$=1.0 ns (c) Worst-Case corner with $t_{\text{launch}}$=1.2 ns

Fig. 7: Setup/Hold Slack Clock Latency valid window.

TABLE I: Cell count and area comparison

| | CWPP | Sequential | saving % |
|---|---|---|---|
| Num Logic Cells | 1697 | 3561 | 52% |
| Num Delay Cells | 529 | N/A | N/A |
| Delay + Logic Cells | 2226 | 3561 | 37% |
| Num Flops | 213 | 594 | 64% |
| Num Clock Tree Buf | 13 | 33 | 61% |
| **Total area (µm2)** | **16064** | **18282** | **12%** |

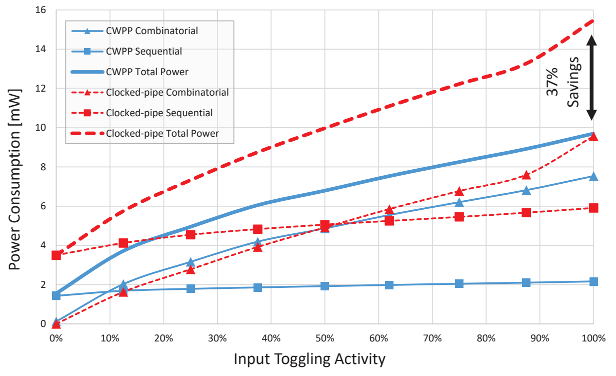


Fig. 8: Power consumption of the CWPP and the clocked-pipeline reference design under varying input toggling activity.

The results are presented in Fig. 8 showing that the CWPP provides a 37% reduction in power consumption for a full toggle rate, and similar benefits throughout the entire spectrum of activity factors, reaching as high as 56% under no toggling. Whereas the combinatorial power is similar between the designs, the clock and sequential power is, as expected, significantly reduced for the CWPP implementation[5].

## V. Conclusions

Clockless wave propagated pipelining is an area and energy-efficient technique for realizing high throughput datapaths. However, due to complex design and the lack of a standard methodology for validating the implementation, this approach has all but been abandoned in advanced processing nodes. Recent work proposed an algorithm for automating CWPP design that is applicable to any generic combinatorial circuit without feedback. In this work, we have extended this algorithm and integrated it into WP 2.0, a utility that enables full-flow implementation of CWPP networks using commercial EDA tools, CMOS standard cell libraries, and meeting industry-standard signoff requirements. The proposed approach is shown to adhere to standard STA checks for timing verification, while also providing a unique post-silicon capability to optimize operation and enable fixing both max-delay and min-delay violations in fabricated samples. The application of the WP 2.0 utility was demonstrated on a test case dot-product calculation unit, which was shown to reduce power consumption between 37%–54% as compared to a 3-stage clocked-pipeline reference design, while also saving 12% of the silicon area.

[5]Note that the clock and sequential power of the CWPP design are consumed by the input and output registers.

## References

[1] D. Rossi, A. Pullini, I. Loi, M. Gautschi, F. K. Gürkaynak, A. Teman, J. Constantin, A. Burg, I. Miro-Panades, E. Beignè, F. Clermidy, P. Flatresse, and L. Benini, "Energy-Efficient Near-Threshold Parallel Computing: The PULPv2 Cluster," *IEEE Micro*, vol. 37, no. 5, pp. 20–31, 2017.

[2] D. C. Wong, G. De Micheli, and M. J. Flynn, "Designing high-performance digital circuits using wave pipelining: Algorithms and practical experiences," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 1, pp. 25–46, 1993.

[3] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: a tutorial and research survey," *IEEE TVLSI*, vol. 6, no. 3, pp. 464–474, 1998.

[4] W. Kim and Y. Kim, "Automating wave-pipelined circuit design," in *IEEE design and test of computers, 20(6)*. IEEE, 2003, pp. 51–58.

[5] C. T. Gray, W. Liu, and R. K. Cavin III, *Wave pipelining: theory and CMOS implementation*. Springer Science & Business Media, 2012, vol. 248.

[6] O. Zografos, A. De Meester, E. Testa, M. Soeken, P.-E. Gaillardon, G. De Micheli, L. Amaru, P. Raghavan, F. Catthoor, and R. Lauwereins, "Wave pipelining for majority-based beyond-cmos technologies," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1306–1311.

[7] A. Najafi, A. Najafi, and A. Garcia-Ortiz, "Stochastic wave-pipelined on-chip interconnect," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 5, pp. 841–845, 2020.

[8] L. Cotten, "Maximum-rate pipeline systems," in *SJCC*, 1969.

[9] Y. Kra, T. Noy, and A. Teman, "Wavepro: clock-less wave-propagated pipeline compiler for low-power and high-throughput computation," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1291–1294.

[10] O. Maltabashi, Y. Kra, and A. Teman, "Physically-aware affinity-driven multiplier implementation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.