A PLA based Asynchronous Micropipelining Approach for Subthreshold Circuit Design

Nikhil Jayakumar‡
nikhil_at_ece.tamu.edu
Rajesh Garg‡
rajeshgarg_at_tamu.edu
Bruce Gamache*
bruce_dot_gamache_at_conexant.com
Sunil P Khatri‡
sunilkhatri_at_tamu.edu

‡Department of Electrical & Computer Engineering,Texas A&M University, College Station, TX
*Conexant Systems, Inc, Colorado

ABSTRACT

Power consumption is a dominant issue in contemporary circuit design. Sub-threshold circuit design is an appealing means to dramatically reduce this power consumption. However, sub-threshold designs suffer from the drawback of being significantly slower than traditional designs. To reduce the speed gap between sub-threshold and traditional designs, we propose a sub-threshold circuit design approach based on asynchronous micropipelining of a levelized network of PLAs. We describe the handshaking protocol, circuit design and logic synthesis issues in this context. Our preliminary results demonstrate that *by using our approach, a design can be sped up by about 7×*, with an area penalty of 47%. Further, our approach yields an energy improvement of about 4×, compared to a traditional network of PLA design. Our approach is quite general, and can be applied to traditional circuits as well.

Categories and Subject Descriptors: B.7.1 [Integrated Circuits]: VLSI

General Terms: Design

Keywords: Sub-threshold, Micro-pipelining, PLA, Asynchronous

1. INTRODUCTION

In recent times, the excessive power consumption of contemporary circuits has become a dominant design concern [1], hampering the relentless march towards smaller, faster and cheaper electronic circuits. In addition to shortened battery life for portable electronics, higher power consumption results in aggravated on-chip temperatures, which can result in a reduced operating life for the chip [2].

Due to their extreme low power consumption, sub-threshold design approaches are appealing for a widening class of applications which demand low power consumption and can tolerate larger circuit delays. Such applications include sensor networks [3, 4, 5] and wearable / portable computing and communication systems, which have generated a significant interest in the area of low power computing. The success of such systems relies on the ability to design low power digital circuits. Although great strides have been made in this arena – one of the important digital design problems of the day – much more remains to be done.

The key contribution of this paper is to come up with a technique which enjoys an extreme low power consumption due to the use of sub-threshold circuitry, but at the same time, compensates for the sub-threshold delay penalty. Such techniques would widen the applicability of sub-threshold circuit design approaches to a broader class of applications. *The proposed approach utilizes a network of PLA (NPLA) based sub-threshold circuit design approach, configured in an asynchronous micropipelined structure to enhance the speed of the circuit.* We provide details about our micropipelined PLA based asynchronous protocol, the logic synthesis approach to decompose a circuit into this circuit paradigm, and also the delay, area and power characteristics of designs which are implemented using our approach.

Sub-threshold (leakage or cut-off) [6, 7] currents are a necessary evil in traditional VLSI design methodologies. These currents increase exponentially as threshold voltage scales, creating a serious problem for traditional design approaches. Our approach is based on the exclusive use of sub-threshold conduction currents to perform circuit operations, turning this problem into an opportunity. This yields a dramatic improvement (previous work in the area demonstrate a 100-500× reduction [8, 9, 10, 11] in power consumption compared to traditional circuit design approaches. This reduction makes it feasible to design extreme low power circuits with such an approach.

The magnitude of sub-threshold currents are significantly smaller than linear or saturation currents since I_{D0} is extremely small. In traditional digital VLSI design, the sub-threshold region of operation is avoided. Circuit operation is based purely on linear or saturation mode currents, and sub-threshold currents are viewed as an attendant evil, since they contribute towards leakage power consumption when the device is in stand-by. As the minimum feature size of processes continues to shrink with each successive process generation (along with the value of supply voltage and therefore V_T), leakage currents increase exponentially, as we can see from the above equation. On the one hand this would suggest the choice of larger V_T values, but this in turn leads to slower circuits since the device (operating in linear or saturation region) has a slower turn-on when V_T is increased. Choosing a lower V_T results in lower delays but increased leakage power dissipation. Leakage power already

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

comprises about 50% of the total power dissipation of modern designs [1, 12], so this option is not desirable either.

We exclusively utilize sub-threshold leakage currents to implement circuits. This is achieved by actually setting the circuit power supply VDD to a value less than or equal to V_T. This choice results in dramatically smaller conduction currents and power, but also larger circuit delays as well. This paper describes asynchronous network of PLA based techniques to reduce these circuit delays, and thereby widen the applicability of sub-threshold circuit approaches.

The **advantages** of a circuit design approach that utilizes sub-threshold conduction are:

- Power is significantly (100-500×) lower [11]
- Circuits get *faster* at higher temperature [13].
- Device transconductance is an exponential function of V_{gs}, resulting in a high ratio of *on* to *off* current in a device stack. As a consequence, circuit noise margins are high.
- Delay gets worse by 10-25×, but the PDP (Power-Delay Product) improves by 10-20× [11]. It has been shown[14] that we can obtain an improvement in the Energy-Delay product as well, by operating the circuit in the near-subthreshold region.

The **disadvantages** of a sub-threshold design methodology are:

- I_{ds} exhibits an exponential dependence on temperature, requiring circuitry to compensate for this effect.
- I_{ds} is highly dependent on process variations. For example, small changes in V_T result in large changes in I_{ds} due to the exponential dependence of I_{ds} on V_T. We therefore require circuitry to compensate for this effect as well.
- I_{D0} is small, resulting in small I_{ds}.

Techniques to dynamically compensate for the process, voltage and temperature dependence of the sub-threshold delay have been proposed [11]. Also, applications such as digital wrist-watches and calculators have utilized extreme low power circuitry based on sub-threshold conduction. However, these applications are analog in nature, or implement very simple digital circuits. The design methodologies used are ad-hoc.

Our paper provides a systematic EDA framework for the design of complex digital systems using sub-threshold NPLA circuits. It utilizes an asynchronous micropipelining approach to speed up the sub-threshold design. Our experiments indicate that this approach yields a circuit speedup of 7×, with a 4× improvement in energy consumption, compared to traditional NPLA designs. Circuit speedup is measured in terms of computational throughput.

The remainder of this paper is organized as follows. The next Section 2 talks about methods similar to our own. Section 3 describes our design flow, while Section 4 describes our experimental results. Finally, in Section 5, we make concluding comments and discuss further work that needs to be done in this area.

2. PREVIOUS WORK

In [8, 9, 10], the authors discuss sub-threshold logic for ultra-low power circuits. They state that their approach would be useful for applications where speed is of secondary importance. In one of two proposed approaches, they describe circuitry to stabilize the operation of their circuit across process and temperature variations. In these papers, the idea of using sub-threshold circuits was introduced from a device standpoint, and candidate compensation circuits were proposed. Also, no systematic design methodology was provided to design circuits in this methodology.

In [15], the authors report a sub-threshold implementation of a multiplier. The methodology utilizes a leakage monitor, and a circuit which compensates the sub-threshold current across process and temperature variations. Another technique, which dynamically compensates for the process, voltage and temperature dependence of the sub-threshold delay has been proposed [11]. It is based on a network of PLA based design approach, where PLAs are clustered, with a common Nbulk per PLA. This bulk voltage is modulated to synchronize the delay of a representative PLA in the cluster to an external clock signal. In general, any practical sub-threshold design approach requires process, voltage and temperature compensation circuitry. For our asynchronous micropipelined approach, we assume the use of the latter compensation technique.

Our approach assumes the use of a sub-threshold circuit design approach, and provides a means to enhance the speed of the sub-threshold design by using asynchronous micropipelines. In this sense, it is orthogonal to the above approaches, which discuss circuit level approaches to design sub-threshold circuits.

3. OUR APPROACH

Our approach to enhancing the speed of sub-threshold circuits is based on implementing the circuit using a micropipelined asynchronous network of PLAs. This implementation has the advantage of increasing the throughput of the circuit to a constant, regardless of the topological depth of the circuit. PLAs with adjacent topological depths in this structure communicate via an asynchronous handshake, which ensures correct operation of the design.

In Section 3.1, we describe the operation of the asynchronous micropipeline, along with its handshaking protocol. Section 3.2 indicates our approach for synthesizing a network of PLAs from a multi-level logic circuit, in a manner which is optimized for an asynchronous micropipeline based implementation. We point out that in addition to PLAs, this methodology requires a specialized circuit block (which we call a *stutter* block) which delays signals that traverse multiple levels in the NPLA. Section 3.3 describes the design of a single PLA in this methodology, and the handshaking logic within each PLA. We also discuss details of each PLA (maximum number of inputs, outputs and rows) used in our approach. We also describe the design of the stutter blocks used in our approach.

3.1 Asynchronous Micropipelined NPLAs

Our asynchronous micropipelined design methodology is based on the use of NPLAs [16, 17]. The choice of PLAs for implementation of the underlying logic is that these structures can be designed to have a constant output delay across all possible input combinations. Also, the use of pre-charged NOR-NOR PLAs results in a compact and fast circuit. It was shown that for a single PLA, the delay was about 48% and the area about 46% compared to a standard cell based design [17], as long as the PLA was medium-sized (with 7-15 inputs, 5-10 outputs and 15-30 rows).

For a robust asynchronous micropipelined implementation, it is critical that the delays of the underlying circuit blocks are extremely predictable. The constant delay of a dynamic PLA over all input combinations makes it a very attractive choice in this context. Also, we utilize PLAs of fixed size in our approach. In this way we satisfy this important requirement of predictable delay. Note that in a sub-threshold design methodology, circuit delays vary significantly as a function of process, temperature and voltage (PVT) variations, as indicated in Section 1. However, we propose to use an on-the-fly, dynamically delay-compensated NPLA structure [11], which was shown to dramatically reduce this variation. The residual variation in NPLA delay after applying this technique is minimal. Therefore,

a simple guardbanding can achieve a predictable PLA delay across PVT variations in a sub-threshold context.

The structure of the asynchronous micropipelined NPLA is shown in Figure 1. Each PLA is a precharged NOR-NOR structure. However, the determination of when a PLA precharges and evaluates is made based on the handshaking protocol. There is no global clock signal in the design. Each PLA has a *completion* signal (which switches high when evaluation of the PLA completes), which indicates that its outputs have been computed. The inputs of a PLA are indicated as D. PLA outputs are marked as O. Each PLA also has two inputs $P1$ and $P2$ which control the asynchronous handshake. The precharge operation of a PLA begins when P1 goes high, while evaluation starts when $P2$ rises, provided the *completion* signal of the PLA is low.

After the *completion* signal of the topologically lowest level (level 1) PLAs goes low (PLA has precharged), the $P2$ signal of the topologically lowest level PLAs is asserted. This causes level 1 PLAs to evaluate. When the *completion* signal of the level 1 PLAs is asserted, the level 2 PLAs begin evaluation. When the level 2 PLAs start evaluating (a short period after the INTCLK signal of the level 2 PLA rises), the level 1 PLAs start precharging. This ensures that the data from PLAs of level 1 to PLAs of level 2 is held until PLAs of level 2 have latched the data from PLAs of level 1. This handshaking mechanism is utilized across all PLA levels.

The micropipelined structure in Figure 1 shows a single PLA at any topological level. In practice, there may be several PLAs at any level, in which case the *completion* signal for any level i would be generated by logically ANDing the *completion* signals of all PLAs of level i.

The screen capture of a Verilog simulation for a series of 4 PLAs showing the working of our handshaking protocol is shown in Figure 2. Note that this figure illustrates the asynchronous nature of the computation. P2 is a signal from outside the micropipeline that signals the level 1 PLA to start evaluating (if the level 1 PLA is precharged). Once the level 1 PLA completes evaluation it signals the level 2 PLA to start evaluating. This happens at the time instant marked a, which occurs a short handshake period after the level 1 PLA completed its evaluation. We call this handshake period the 'evaluation handshake period'. The level 2 PLA completes its evaluation at the time instant marked b and then after a period equal to the evaluation handshake period, the level 3 PLA starts evaluating at the time instant marked c. A short-period after this (at the time instant marked c), the level 2 PLA starts precharging. We call this short-period the 'precharge handshake period'. P1 is the user acknowledgement signal generated (at time instant marked e) after the PLA at level 4 completes its evaluation and the user has latched the data from the PLAs at this level. When the level 4 PLA receives this signal it starts precharging. If the user is late in acknowledging the data from the PLA at the last level, the pipeline is stalled till P1 is asserted again (at time instant marked f).

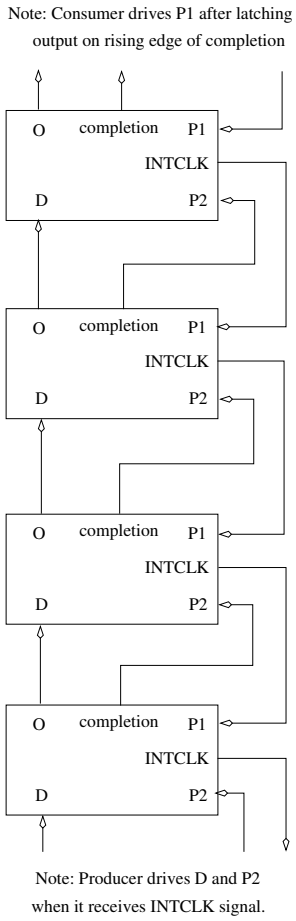

Figure 1: NPLA based Asynchronous Micropipelined Circuit

3.2 Synthesis of Micropipelined PLA Networks

Synthesis of a PLA network for an asynchronous micropipelined implementation consists of a two step process. In the first step, we generate a NPLA from a multi-level logic netlist. In the second, we infer the stuttered signals that are induced by the synthesized result, and augment the netlist of the first part with stutter blocks which delay signals that traverse more than one level of PLAs.

In the first step, we begin by performing technology independent optimizations on the multi-level circuit C. Next, we decompose C into a network C^* of nodes with at most p inputs. In our experiments, $p = 5$. Now C^* is sorted in depth-first manner. The resulting array of nodes is sorted in *levelization*[1] order, and placed into an array L.

Now we greedily construct the logic in each PLA, by successively grouping nodes from L such that the resulting PLA implementation of the grouped nodes N^* does not violate the constraints of PLA width and height. This check is performed in a *check_PLA* routine, which first flattens N^* into a two-level form, P. It then calls *espresso* [18] on the result to minimize the number of cubes in P. Next, *check_PLA* calls a *PLA folding* routine which attempts to fold the inputs of P so as to implement a more complex PLA in the same area. Finally *check_PLA* ensures that the final PLA, after folding and simplification using *espresso*, satisfies the maximum width and height constraints respectively. If so, we attempt to include another node into N^*, otherwise we append the last PLA satisfying the height and width constraints to the result.

The *get_next_element* routine returns the most favorable node n among nodes in the fanout of nodes $n' \in N^*$ and nodes n'' which have the same level as the first node included into N^*, provided that the inclusion of n into N^* would not result in a cyclic PLA dependency graph. If such nodes are not available, the first unmapped node from L is returned. The favorability of a candidate is

[1]Primary inputs are assigned a level 0, and other nodes are assigned a level which is one larger than the maximum level of all their fanins

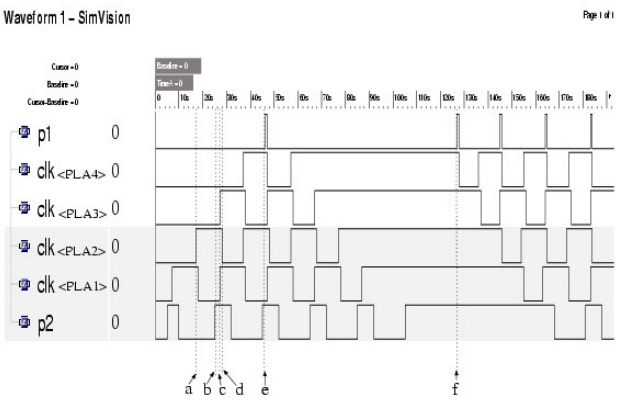

Figure 2: Verilog Simulation of Our Approach

Algorithm 1 Decomposition of a Circuit into a Network of PLAs

```
C = optimize_network(C)
C* = decompose_network(C, p)
L = dfs_and_levelize_nodes(C*)
N* = 0
RESULT = 0
while get_next_element(L) != NIL do
  N* = N* ∪ get_next_element(L)
  P = make_PLA(N*)
  if check_PLA(P,W,H) then
    continue
  else
    Q = remove_last_element(N*)
    RESULT = RESULT ∪ N*
    N* = Q
  end if
end while
```

computed as:

$favorability(n) = 2 * [\#common\ fanins(n,n')] + [\#common\ fanouts(n,n')]$

Nodes with shared fanins and fanouts decrease the number of PLAs created. We also found that shared fanins had a greater effect on this decrease. Hence in evaluating the favorability of a node we gave a greater weight to those nodes that shared a fanin with a node already included in the current PLA.

We implemented the algorithm to decompose a circuit into a network of PLAs in SIS [19]. The pseudo-code is given in Algorithm 1.

The PLAs we used in our experiments had 16 inputs, 14 outputs and 24 rows. A single PLA had a delay of about 225ns, at 25° C, and VDD = 0.2V. Note that we used the BPTM [20] 65nm process cards for our simulations.

Inferring of stuttered signals is performed by traversing the network of PLAs from inputs to outputs. For any output of a PLA of level l, if the PLAs in its fanout have a maximum level of l_{max}, then $l_{max} - l - 1$ stutter signals are inserted for this output, one for every level between l and l_{max}.

3.3 Circuit Details of PLAs and Stutter Blocks

The PLA we use is a precharged NOR-NOR PLA (similar to the ones used in [17, 21, 22, 23]). The schematic view of the PLA circuit is shown in Figure 3. The wordlines of the PLA (which represent the cubes of the function to be implemented), run horizontally through the AND and OR plane of the PLA. The bit lines (which carry the inputs and their complements) run vertically through the AND plane, while the output lines run vertically through the OR plane of the PLA structure. The layout view for this PLA is not shown due to lack of space.

When INTCLK (which is manipulated by the micropipelining handshake protocol) is low, the PLA enters the *precharge* phase. During this time, the horizontal wordlines get precharged. A special wordline (the dummy wordline), which is the maximally loaded wordline also gets precharged. This forces the signal D_CLK to go low, cutting off the OR plane from GND, and causing the output lines to also get precharged. A special output line (which is inverted to produce the signal *completion* shown in Figure 3) also gets precharged. The dummy wordline is designed to be the last wordline to switch (by making it maximally loaded among all wordlines). Similarly, the *completion* line is also the last output line to switch, since it is maximally loaded as well, in comparison to other outputs. The *completion* line switching low signals the completion of the precharge operation of the PLA. In the precharged state, all the wordlines and the output lines of the PLA are precharged. Now, when INTCLK switches high, the PLA enters the *evaluation* phase. In evaluation, if any of the vertical bitlines are high, the wordline that it is connected to, gets pulled low. One of the inputs *and* its complement is connected to the dummy wordline, so that the dummy wordline switches low during *every* evaluate phase. By design, the dummy wordline is the last wordline to switch low. This makes the signal D_CLK go high, as a result of which the GND gating transistor in the OR plane now turns on[2]. The output lines to which, wordlines that have switched low are connected, will switch low. The *completion* line, which is connected to the complement of the dummy wordline is the last line to switch high. This signals the completion of the evaluation operation.

The INTCLK signal is generated from the *completion*, $P1$ and $P2$ signals using the circuit shown in Figure 4. On every rising edge of $P1$, a pulse is generated which makes the INTCLK signal go low. forcing the PLA to enter the precharge phase. In other words, PLA p enters the precharge phase if PLAs at a level above the PLA p have started evaluation (after latching the input data). Once this happens, the *completion* signal of the PLA p falls (after all other signals in p have precharged). At this point, if $P2$ rises, then the PLA p enters the evaluation phase. In other words, if the PLA p has been precharged, and if the PLAs a level below complete their computation, then p enters the evaluation phase. The additional inverter(s) in the path of the *completion* signal are for design guardbanding. In our SPICE [24] simulation of this handshaking block, we found that it had a worst case delay of 25ns for INTCLK to fall, measured with respect to $P1$ rising. We called this the 'precharge handshake period' in Section 3.1. The handshaking block had a worst case delay of 60ns for INTCLK to rise (measured with respect to *completion* falling). We called this the 'evaluation handshake period' in Section 3.1.

Note that each of the PLAs has a set of level sensitive latches on its inputs. When the PLA p has completed its computation, these latches hold their state, ensuring that the precharging of PLAs a level below does not change the state of the outputs of p that have been computed.

In this manner, odd levels of the NPLA precharge while even levels of PLAs evaluate. This pattern alternates continuously.

The stutter block is simply a series of latches, implemented in

[2]Note that in the sub-threshold region a transistor is either *off* or *less off*. For the sake of simplicity, we say that an NMOS transistor is *on* when its gate is at VDD and *off* when its gate is at GND. Similarly we say a PMOS transistor is *on* when its gate is at GND and *off* when its gate is at VDD.

the footprint of a PLA (in terms of height). Its function is to delay signals which traverse across levels of PLAs, in order to guarantee correct operation under asynchronous micropipelining. For example, if there is a signal S_{jump1} that is an output of a level 1 PLA and is an input to a level 3 PLA, then a stutter block, consisting of a single latch, is placed between the two PLAs. The signal S_{jump1} is used as the data input to this latch and the data is latched using the INTCLK signal from level 2 PLA(s). This ensures that all the inputs to the level 3 PLA(s) are ready at the same time. For a signal traversing across n levels, n latches are required.

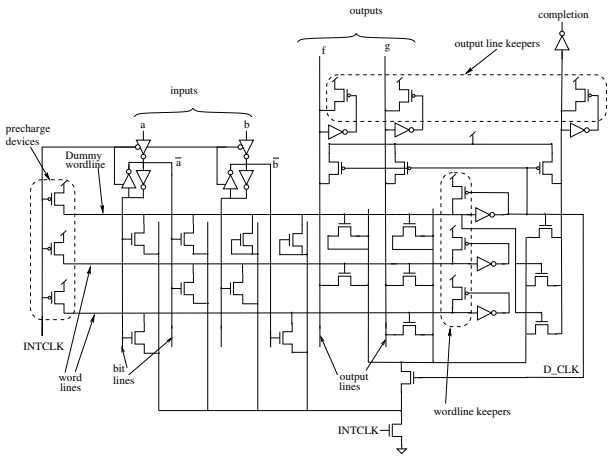

Figure 3: Schematic view of the PLA

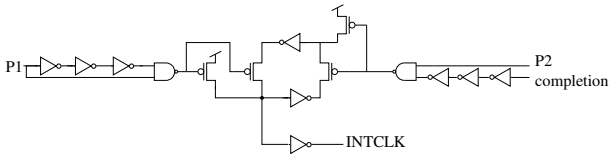

Figure 4: Micropipelined PLA Handshaking Logic

4. EXPERIMENTAL RESULTS

To compare the characteristics of an asynchronous micropipelined network of PLAs with that of a network of PLAs, we performed extensive simulations. All circuit simulations were done in SPICE [24], using 65nm BPTM [20] model cards. The area of any of the two design styles was computed using the sum of the areas of all the PLAs in the design, including the area of any stutter blocks (in the case of the micropipelined network of PLAs).

The asynchronous micropipelined network of PLAs has a throughput of

$T = \frac{1}{T_{eval}+T_{pchg}+2\cdot H_{eval}+H_{pchg}}$

Here T_{eval} is the evaluation delay of the PLA (recall, we utilize fixed sized PLAs in the design), T_{pchg} is the precharge delay of the PLA, H_{eval} is the evaluation handshake period and H_{pchg} is the precharge handshake period. The values of T_{eval}, T_{pchg}, H_{eval}, H_{pchg} are 210ns, 155ns, 60ns and 25ns respectively. As a consequence, the throughput is $\frac{1}{510ns}$. Note that the latency is still proportional to the number of PLA levels in the design, but the throughput is a constant.

In the traditional network of PLA implementation, all levels of PLAs are precharged together and then evaluate in a domino fashion. The timing diagram of this is shown in Figure 5. In case of the traditional network of PLA implementation, the delay is given by the topological depth of the PLA network (in terms of number of PLAs) times the evaluation delay T_{eval} of each PLA. We also add to this the time taken to precharge all the PLAs in the design.

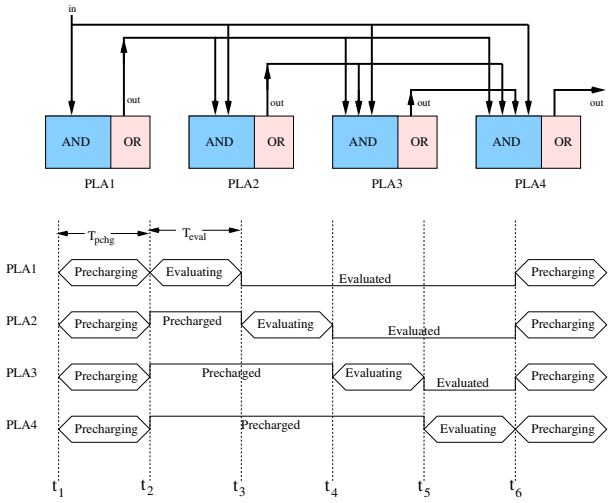

Figure 5: Timing Diagram for Traditional NPLAs

We also compared the energy consumption of the two types of implementations. More specifically we compared the energy consumption per computation in the two types of NPLAs.

For the micropipelined implementation, we first found (through spice simulation) the energy consumption for the operation of 1 PLA (over a period of 510ns) and multiplied this by the number of PLAs. To this, we add the energy consumption of the handshaking logic and the energy consumption in the stutter blocks. This gives us the energy consumption for one computation through the micropipelined NPLA.

While a micropipelined PLA spends very little time (equal to the handshaking periods) in a *precharged* state or *evaluated* state, the traditional NPLA spends substantial periods of time in the *precharged* state and *evaluated* state. This is evident from the timing diagram shown in Figure 5. [3] As a consequence, the micro-pipelined PLA wastes less energy in leakage than traditional PLAs.

Table 1 reports the results of our experiments. The first column represents the circuit under study. The second column reports the number of PLAs required, while the third column reports the number of stutter blocks required to make the network of PLAs micropipelined. The next 3 columns report the delay of the non-micropipelined PLA, the throughput of the micropipelined PLA, and the ratio. Note that the throughput of the micropipelined PLAs is constant. The traditional PLA network delay is computed as described above. We note that the *micropipelined PLA results in a speedup of about 7× over a traditional design*. This is because the micropipelined PLA is pipelined and hence the throughput is what we compare. Further, for network of PLA circuits with larger topological depths, this improvement is more pronounced. The energy consumption of the micropipelined NPLAs is about 4× lower than the energy consumption of the traditional NPLAs. The area penalty for the approach is about 47% on average.

[3] In [14], the authors point out this fact and use it find an optimum supply voltage for minimum energy operation.

Ckt	# PLAs	# Stutter blks	Delay(ns) ↓			Energy(fJ) ↓			Area(μ^2) ↑		
			Non-μpipe	μpipe	Impr.	Non-μpipe	μpipe	Impr.	Non-μpipe	μpipe	Ovh
alu4	14	5	2885	510	5.66	5984.80	1811.43	3.30	9408	12768	1.36
apex6	24	12	2465	510	4.83	9033.09	3261.19	2.77	16128	24192	1.50
C432	11	4	2255	510	4.42	3877.22	1397.00	2.78	7392	10080	1.36
C499	14	4	2255	510	4.42	4961.02	1768.64	2.80	9408	12096	1.29
C880	16	5	2255	510	4.42	6088.11	2052.22	2.97	10752	14112	1.31
C1355	21	10	3305	510	6.48	10198.86	2863.68	3.56	14112	20832	1.48
C1908	24	13	3935	510	7.72	13814.19	3307.96	4.18	16128	24864	1.54
C2670	34	13	3515	510	6.89	18694.33	4472.11	4.18	22848	31584	1.38
C3540	67	46	7505	510	14.72	73900.56	9777.18	7.56	45024	75936	1.69
pair	65	35	4565	510	8.95	44442.77	9047.27	4.91	43680	67200	1.54
rot	19	13	3095	510	6.07	8966.68	2774.15	3.23	12768	21504	1.68
Avg	28.09	14.55			6.78			3.84			1.47

Table 1: Comparison of Micropipelined with Traditional Circuits

5. CONCLUSION

In recent times, power consumption has a dominant issue in VLSI circuit design. Sub-threshold circuit design is an appealing means to dramatically reduce this power consumption. However, sub-threshold designs suffer from the drawback of being significantly slower than traditional designs. This paper describes a means to reclaim the speed penalty associated with sub-threshold designs. The approach is based on the use of a sub-threshold circuit design approach which is based on asynchronous micropipelining of a levelized network of PLAs. We have come up with the handshaking protocol, circuit design and logic synthesis issues in this context, and our preliminary results demonstrate that by using our approach, a design can be sped up by 7×, with an area penalty of 47%. Further, the energy consumption of micropipelined NPLA based circuits is about 4× lower than that of the traditional NPLAs circuits. Our simulations were validated in VERILOG, and circuit level characteristics were extracted using SPICE modeling. Recent work [25, 26, 27, 14] suggests that the optimal VDD for minimum energy operation is slightly above V_T. The techniques described in this paper are equally applicable for these operating conditions as well.

6. REFERENCES

[1] "The International Technology Roadmap for Semiconductors." `http://public.itrs.net/`, 2003.

[2] W. Daasch, C. Lim, and G. Cai, "Design of vlsi cmos circuits under thermal constraint," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, pp. 589–593, Aug 2002.

[3] S.-H. Choi, B.-K. Kim, J. Park, C.-H. Kang, and D.-S. Eom, "An implementation of wireless sensor network," *IEEE Transactions on Consumer Electronics*, vol. 50, pp. 236–244, Feb 2004.

[4] "The MultimodAl NeTworks of In-situ Sensors (MANTIS) project." `http://mantis.cs.colorado.edu`, 2004.

[5] A. Abidi, G. Pottie, and W. Kaiser, "Power-conscious design of wireless circuits and systems," *Proceedings of the IEEE*, vol. 88, pp. 1528–1545, Oct 2000.

[6] J. Rabaey, *Digital Integrated Circuits - a design perspective*. Prentice Hall, 1996.

[7] N. Weste and K. Eshraghian, *Principles of CMOS VLSI design - a systems perspective*. Addison-Wesley, 1988.

[8] H. Soeleman, K. Roy, and B. Paul, "Robust subthreshold logic for ultra-low power operation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, pp. 90–99, Feb 2001.

[9] H. Soeleman and K. Roy, "Digital CMOS logic operation in the sub-threshold region," in *Tenth Great Lakes Symposium on VLSI*, pp. 107–112, Mar 2000.

[10] H. Soeleman and K. Roy, "Ultra-low power digital subthreshold logic circuits," in *International Symposium on Low Power Electronic Design*, pp. 94–96, 1999.

[11] N. Jayakumar and S. Khatri, "A variation-tolerant sub-threshold design approach," in *Proceedings, Design Automation Conference*, pp. 716–719, June 2005.

[12] M. Mui, K. Banerjee, and A. Mehrotra, "Power supply optimization in sub-130 nm leakage dominant technologies," in *Proceedings. 5th International Symposium on Quality Electronic Design*, pp. 409–414, Mar 2004.

[13] K. Kanda, K. Nose, K. Kawaguchi, and T. Sakurai, "Design impact of positive temperature dependence on drain current in sub-1-V CMOS VLSIs," *IEEE Journal of Solid-State Circuits*, vol. 36, p. 1559=1564, Oct 2001.

[14] N. Jayakumar and S. P. Khatri, "Minimum energy near-threshold network of pla based design," *Proceedings, IEEE International Conference on Computer Design*.

[15] B. Paul, H. Soeleman, and K. Roy, "An 8x8 sub-threshold digital CMOS carry save array multiplier," in *European Solid State Circuits Conference*, Sept 2001.

[16] S. P. Khatri, *Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics*. PhD thesis, UC Berkeley, Dec 1999.

[17] S. Khatri, R. Brayton, and A. Sangiovanni-Vincentelli, "A VLSI design methodology using a network of PLAs embedded in a regular layout fabric," Tech. Rep. UCB/ERL M99/50, Electronics Research Laboratory, University of California, Berkeley, May 1999.

[18] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[19] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. UCB/ERL M92/41, Electronics Research Laboratory, Univ. of California, Berkeley, CA 94720, May 1992.

[20] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," in *Proc. of IEEE Custom Integrated Circuit Conference*, pp. 201–204, Jun 2000. http://www-device.eecs.berkeley.edu/ ptm.

[21] F. Mo and R. Brayton, "PLA-based regular structures and their synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, pp. 723–729, June 2003.

[22] F. Mo and R. Brayton, "River PLAs: a regular circuit structure," in *Proceedings, Design Automation Conference*, pp. 201–206, June 2002.

[23] F. Mo and R. Brayton, "Whirlpool PLAs: a regular logic structure and their synthesis," in *IEEE/ACM International Conference on Computer Aided Design*, pp. 543–550, Nov 2002.

[24] L. Nagel, "Spice: A computer program to simulate computer circuits," in *University of California, Berkeley UCB/ERL Memo M520*, May 1995.

[25] R. Gonzalez, B. M. Gordon, and M. A. Horowitz, "Supply and threshold voltage scaling for low power cmos," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, 1997.

[26] A. Wang, A. Chandrakasan, and S. Kosonocky, "Optimal supply and threshold scaling for subthreshold CMOS circuits," *Proceedings, IEEE Computer Society Annual Symposium on VLSI*, pp. 5–9, Apr 2003.

[27] B. H. Calhoun, A. Wang, A. Chandrakasan, and S. Kosonocky, "Device sizing for minimum energy operation in subthreshold circuits," *Proceedings, IEEE Custom Integrated Circuits Conference*, pp. 95–98, Oct 2004.