The Wave Pipeline Effect on LUT-based FPGA Architectures

Eduardo I. Boemo, Sergio López-Buedo, and Juan M. Meneses
E.T.S.I. Telecomunicación. 28040 Ciudad Universitaria, Madrid - España.
e-mail: ivan@die.upm.es, fax: +34 (1) 336 73 23

Abstract

The wave pipeline effect is based on the equalization of all path delays in order to allow several "waves" of data to travel along the circuit with a separation several times smaller than the maximum combinational delay of the circuit. The construction of wave pipelines requires gates and buffers with data-independent delay, and a well-characterized interconnection network delay model, in order to allow the equalization process to be managed by the designer. These features are inherently present in several RAM-based FPGAs architectures. Look-up tables (LUTs) permit the delay of digital blocks with different types of gates or different logic depth to be equalized; moreover, the delay of a FPGA interconnection network is completely parameterized and is a priori known. This paper describes a LUT-based wave pipeline array multiplier manually implemented using a Xilinx chip. The results show that, even for a single-phase non-skewed clocking strategy, a throughput as high as 85 MHz (measured) can be achieved, with 8 waves running in a 13-LUT logic depth combinational array with registered I/O, producing an initial latency of 9 clock cycles. For the FPGA architecture and the topology selected, such a large throughput/latency ratio would be impossible using classical pipelininig.

1 Introduction

In classical pipelines, registers not only increment the chip area, but also limit the minimum size of the stages, because of their setup time and propagation delay. Thus, for fine-grain pipelines, increasing their frequency operation follows the law of diminishing returns: the number of register (area) must be almost duplicated in each additional logic depth reduction in order to obtain a small circuit speedup. Wave pipelines or maximum-rate circuits [Cot65] provide a possible way to avoid this extra cost: they can increase the throughput of a circuit without making use of intermediate registers.

The wave pipeline effect is based on the equalization of all combinational path delays between consecutive register banks, allowing several waves of data to travel along the circuit without interference, using the capacitance of each node as a virtual intermediate register. The minimum clock period is not limited by the longest path delay but by the difference between the maximum and minimum path delays, plus the clock skew, the rise/fall time, and the setup/hold time of the I/O registers [Won92].

Recent applications of wave pipelines are diverse: digital filters [Bur94], adders [Liu94], memories [Now94], multipliers [Gho95], [Kla94], and other arithmetic modules [Fly95]. These circuits have been implemented using special ECL and CMOS cells, designed to achieve data-independent propagation gate delays. However, to the best of our knowledge, the usefulness of look-up tables as building blocks for the equalization process have not been reported in the scientific literature. In this paper, this alternative has been explored by using Xilinx FPGAs whose characteristics are suited to wave pipeline requirements: their LUTs have been designed with data-independent delay in order to improve simulation accuracy [Xil94], and the layout editor allows additional delay elements to be inserted (LUT or wires) in the datapath. These two factors facilitate the equalization task.

The use of the wave pipeline technique in FPGAs could seem inapropiate: since registers are included on most of commercial chips, they are "free" components and their use does not incur an area penalty, in contrast to other VLSI technologies. However, some peculiarities of wave pipelines justify an exploration of this technique even on FPGAs: the possibility of reducing power consumption (the equalization not only eliminates registers, but also diminishes datapath spurious activity), and the obtention of a high throughput/latency ratio.

In this paper, a Xilinx XC4000 series chip has been utilized as a platform to run wave pipeline prototypes. In the next section, the main concepts and drawbacks of wave pipelines are reviewed. In Section 3 the experimental results are summarized.

2 Elements of Wave Pipelines

The construction of a wave pipeline usually requires the addition of both extra cells and wiring in order to slow down all the fastest paths so that their delay matches that of paths with longest delay. Thus, wave pipelines are just another way to trade speed for additional area and design time: their potential benefits must be evaluated for each technology.

The main advantages of wave pipelines in FPGAs are consequences of the avoidance of the setup and propagation register delays. LUT, wire and register delays are similar in most of the FPGA chips; then, for fine-grain conventional pipelines, the stage delay is dominated by wires and registers, which can constitute as much as the 70% of the clock period. Moreover, the delay of each register inserted in the datapath must be added to calculate the initial latency. On the contrary, the data storage on wave pipelines makes use of node capacitances instead of registers; thus, the delay of each "virtual" pipeline stage does not include the fraction of propagation delay corresponding to the registers. In the same way, the wave design mode does not increase the initial latency since the longest I/O path of the combinational circuit is not modified.

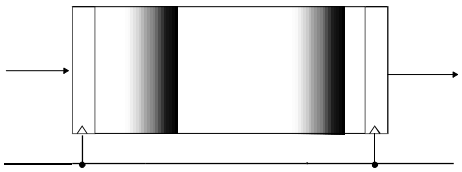

Fig.1a: The wave emitted in t=kT reaches the reception registers after the (k+1)T edge and before the (k+2)T setup window. Thus, the results are correctly latched. Two waves travel along the stage.

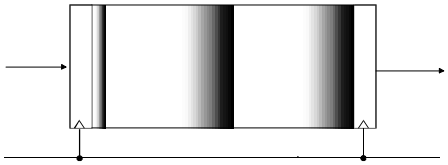

Fig.1b. A frequency increment produces a setup window violation in the reception register bank.

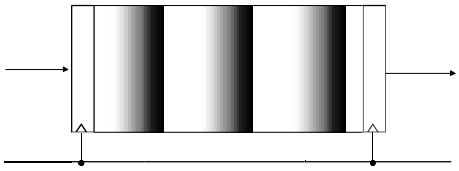

Fig.1c: For this frequency, the wave that enters in t=kT reaches the reception register after (k+2)T, but it is stable before the clock edge (k+3)T. Once again, each result is latched properly.

The use of a single-phase non-skewed clock (as usual on FPGAs) gives rise to an important wave pipeline drawback: a set of frequency bands where the circuit does not work. This phenomenon occurs every time the operation mode (that is, the number of waves that run together inside the circuit) is changed. For example, in Fig.1a the circuit is running with two waves inside; however, if the clock frequency is increased, there is an interval in which three waves start to enter and the output register setup time is violated (Fig.1b). This problem can be corrected simply by an additional increment of the clock frequency. Thus, the three waves become closer and the setup problem disappears. The circuit can run in this mode until a new clock frequency increment produces the entrance of four waves and a new setup violation occurs. Depending on the equalization grade, a further frequency increment can solve the problem once again. However, note that each new operation band is narrower than the previous one.

The main disadvantage of wave pipelines is their sensitivity to parameters that affect propagation delay such as power supply voltage, process variations, or temperature [Now94a]. Although these problems can be severe when circuits are used in production, they can be bearable when FPGAs are employed to prototype in-laboratory future ASICs.

3 Experimental Results

The design of wave pipelines requires exact models rather than the usual conservative worst-case delay specification of VLSI foundries. Nevertheless, it is possible to make use of commercial technologies by applying a strategy called categorical matching [Liu94], which allows the path inbalance caused by the different grades of accuracy of the simulation models of each circuit component (gates, wires, vias, etc.) to be minimized. Using this approach, all data in the FPGA wave pipeline must pass through the same number of LUTs, in the same way that all data passes through the same number of registers in a conventional pipeline. Moreover, all paths should be composed of the same number of the other FPGA elements: "pips", "magic boxes", and so on, as far as possible.

In order to quantify the LUT-based wave pipeline option, a Guild multiplier [Gui69] was implemented and characterized by using a Xilinx XC4005PC84-6 chip. This circuit, depicted in Fig.2 for n=7 bits, consist of an array of n^2 cells, that receive the input data on global lines (the vertical and horizontal wires of the array) and pass the sum and carry bits using local lines (the diagonal wires). Each cell is composed of an AND gate plus a full-adder, and has 4 inputs and 2 outputs. Just one cell can be fitted in one XC4000 CLB. The array has been wave pipelined in horizontal stages.

The layout of the circuit was performed manually using the Xact editor. First, the cells were placed as similarly as possible to the original topology, in order to take advantage of the spacial regularity. Second, the horizontal global wires of data were assigned to long lines. Finally, the rest of the interconnections were mapped in a regular pattern of segment wires, that was repeated along the array. This methods achieved equalization and avoided errors. In Fig.3, the final layout is depicted, demonstrating the uniform use of additional resources (LUTs, wires, and magic boxes) in order to delay the LSBs of the product (right columns) so that they are equalized with the longest paths.

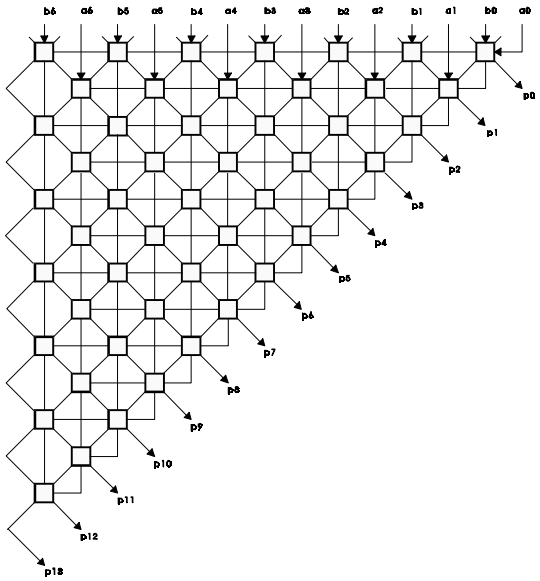

Fig.2: 7-bit Guild array multiplier.

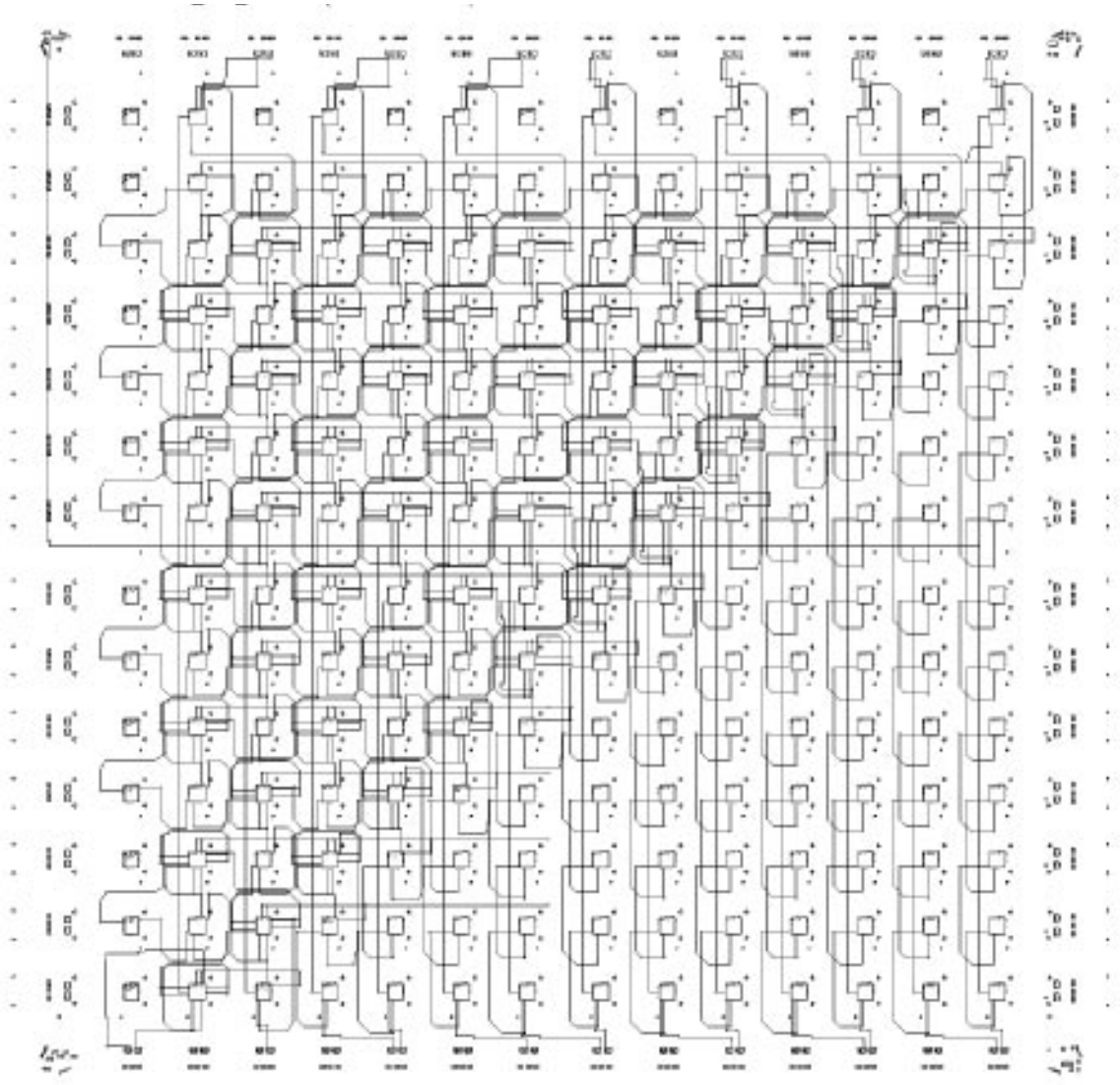

Fig.3: XC4005PC8-6 wave pipeline array multiplier layout.

Equalization in circuits with both global and local communications is difficult. In this case, the cost of using long lines (around 3 ns for the chip selected) to implement the horizontal global channels was balanced by the routing of the other data lines through two magicboxes. In situations where the long line was too heavily loaded (so that its delay approached 4 ns), the preceding local interconnection in the same path was routed through only one magicbox, so that the difference was compensated. In the first segment of the pipeline, the long line delay of 4.1 ns could not be reduced due to the lack of routing resources. These lines were balanced by routing the rest of wires throuh three magicboxes.

By using the methodology described above, a 7-bit wave pipeline Guild multiplier with a constant logic depth of 13 LUT in all paths was obtained. It makes use of 182 CLBs. The maximum unbalance between all the I/O paths was less than 2.1 ns (simulation result). The components of the slower (left column) and faster (right column) paths, determined using the Xdelay tool, are shown in Table 1 (note that the longest path delay would limit the speed below 8 MHz). In Fig.4 the histogram of wire delay is depicted and the homogeneity of the interconnection used can be observed. It is very different to the Pareto-Levy net delay distribution of regular arrays processed using an automatic partitioning, placement and routing strategy.

```
Source clock net : "CLK" (Rising edge)
From: Blk a0/b0 CLOCK         to CLB_R1C14.XQ: 5.0ns ( 5.0ns)
Thru: Net a0                  to CLB_R2C2.G4 : 4.3ns (  9.3ns)
Thru: Blk <0-6>               to CLB_R2C2.Y  : 6.0ns ( 15.3ns)
Thru: Net so<0-6>r            to CLB_R3C3.F3 : 3.2ns ( 18.5ns)
Thru: Blk <1-6>               to CLB_R3C3.X  : 6.0ns ( 24.5ns)
Thru: Net co<1-6>r            to CLB_R4C2.G1 : 3.0ns ( 27.5ns)
Thru: Blk <2-6>               to CLB_R4C2.Y  : 6.0ns ( 33.5ns)
Thru: Net so<2-6>r            to CLB_R5C3.F3 : 3.2ns ( 36.7ns)
Thru: Blk <3-6>               to CLB_R5C3.X  : 6.0ns ( 42.7ns)
Thru: Net co<3-6>r            to CLB_R6C2.G1 : 3.1ns ( 45.8ns)
Thru: Blk <4-6>               to CLB_R6C2.Y  : 6.0ns ( 51.8ns)
Thru: Net so<4-6>r            to CLB_R7C3.F3 : 3.1ns ( 54.9ns)
Thru: Blk <5-6>               to CLB_R7C3.X  : 6.0ns ( 60.9ns)
Thru: Net co<5-6>r            to CLB_R8C2.G1 : 3.1ns ( 64.0ns)
Thru: Blk <6-6>               to CLB_R8C2.Y  : 6.0ns ( 70.0ns)
Thru: Net so<6-6>r            to CLB_R9C3.F3 : 3.2ns ( 73.2ns)
Thru: Blk <7-6>               to CLB_R9C3.X  : 6.0ns ( 79.2ns)
Thru: Net co<7-6>r            to CLB_R10C2.G1: 3.1ns ( 82.3ns)
Thru: Blk <8-6>               to CLB_R10C2.Y : 6.0ns ( 88.3ns)
Thru: Net so<8-6>r            to CLB_R11C3.F3: 3.1ns ( 91.4ns)
Thru: Blk <9-6>               to CLB_R11C3.X : 6.0ns ( 97.4ns)
Thru: Net co<9-6>r            to CLB_R12C2.G1: 3.1ns (100.5ns)
Thru: Blk <10-6>              to CLB_R12C2.Y : 6.0ns (106.5ns)
Thru: Net so<10-6>r           to CLB_R13C3.F3: 3.2ns (109.7ns)
Thru: Blk <11-6>              to CLB_R13C3.X : 6.0ns (115.7ns)
Thru: Net co<11-6>r           to CLB_R14C2.G1: 3.0ns (118.7ns)
Thru: Blk <12-6>              to CLB_R14C2.Y : 6.0ns (124.7ns)
Thru: Net p12                 to P35.O       : 2.6ns (127.3ns)
  To: FF Setup (D), Blk c12             : 8.0ns (135.3ns)
```

```
Source clock net : "CLK" (Rising edge)
From: Blk a6/b6 CLOCK         to CLB_R1C2.XQ : 5.0ns (  5.0ns)
Thru: Net a6                  to CLB_R2C3.F2 : 4.1ns (  9.1ns)
Thru: Blk a6r                 to CLB_R2C3.X  : 6.0ns ( 15.1ns)
Thru: Net a6r                 to CLB_R4C3.G4 : 3.0ns ( 18.1ns)
Thru: Blk a6r3                to CLB_R4C3.Y  : 6.0ns ( 24.1ns)
Thru: Net a6r2                to CLB_R4C3.F2 : 3.0ns ( 27.1ns)
Thru: Blk a6r3                to CLB_R4C3.X  : 6.0ns ( 33.1ns)
Thru: Net a6r3                to CLB_R6C3.G4 : 3.0ns ( 36.1ns)
Thru: Blk a6r5                to CLB_R6C3.Y  : 6.0ns ( 42.1ns)
Thru: Net a6r4                to CLB_R6C3.F2 : 2.9ns ( 45.0ns)
Thru: Blk a6r5                to CLB_R6C3.X  : 6.0ns ( 51.0ns)
Thru: Net a6r5                to CLB_R8C3.G4 : 3.1ns ( 54.1ns)
Thru: Blk a6r7                to CLB_R8C3.Y  : 6.0ns ( 60.1ns)
Thru: Net a6r6                to CLB_R8C3.F2 : 2.9ns ( 63.0ns)
Thru: Blk a6r7                to CLB_R8C3.X  : 6.0ns ( 69.0ns)
Thru: Net a6r7                to CLB_R10C3.G4: 3.1ns ( 72.1ns)
Thru: Blk a6r9                to CLB_R10C3.Y : 6.0ns ( 78.1ns)
Thru: Net a6r8                to CLB_R10C3.F2: 2.9ns ( 81.0ns)
Thru: Blk a6r9                to CLB_R10C3.X : 6.0ns ( 87.0ns)
Thru: Net a6r9                to CLB_R12C3.G4: 3.0ns ( 90.0ns)
Thru: Blk a6r11               to CLB_R12C3.Y : 6.0ns ( 96.0ns)
Thru: Net a6r10               to CLB_R12C3.F2: 3.0ns ( 99.0ns)
Thru: Blk a6r11               to CLB_R12C3.X : 6.0ns (105.0ns)
Thru: Net a6r11               to CLB_R13C3.G2: 3.0ns (108.0ns)
Thru: Blk <11-6>              to CLB_R13C3.Y : 6.0ns (114.0ns)
Thru: Net p11r                to CLB_R14C3.F2: 3.1ns (117.1ns)
Thru: Blk p11r2/a6r12         to CLB_R14C3.X : 6.0ns (123.1ns)
Thru: Net p11                 to P37.O       : 2.1ns (125.2ns)
  To: FF Setup (D), Blk c11             : 8.0ns (133.2ns)
```

Table 1: Components of the slower (left column) and faster (right column) paths.

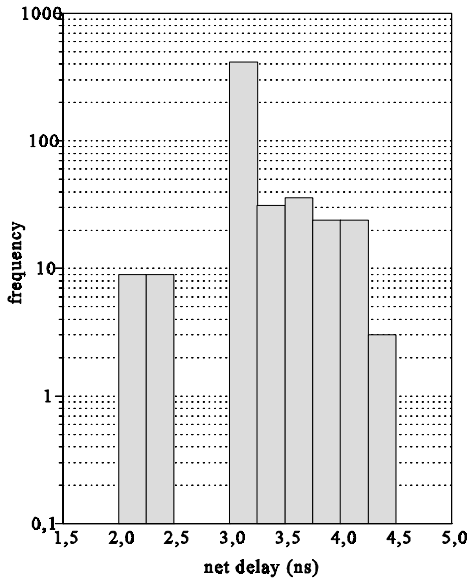

Fig.4: Net delay histogram.

The prototype has been tested using 2^{16} random vectors as well as a set of 16 operand that produce the toggle of almost all the output pins (both sequences were produced using another FPGA, an XC3120-3, that allowed a low-cost pattern generator to be obtained). The second sequence of numbers facilitated the detection of phenomena like double-clocking and zero-clocking, that are common on single-phase clock pipelines.

The circuit operated as fast as fine grain pipelines, but using 28 registers instead of 278 ones. The highest operational frequency band (measured) was between 83 Mhz to 85 Mhz, with 8 waves running inside it, a frequency that is close to the limit of the selected chip. For this example, the circuit ran nearly 10 times faster than the value predicted by the Xdelay tool, meanwhile the factor between simulation results and the real frequency of operation has been measured as 1.3 for non-equalized pipeline circuits (Fig.5).

Running at maximum speed, the circuit is equivalent to a pipeline of 8 stages; then, the latency of the prototype is 9 clock cycles including the I/O registers. Considering that, for n=7 bits, the Guild topology has 13 cells in the longest path, and each of them requires one CLB, the classical pipeline must have 13 stages in order to reach a throughput of 84 Mhz. Even if it were possible to obtain that speed by means of a careful layout, the latency would be 14 clock cycles. Pipelining every two cells could reduce the latency to 8 clock cycles but would imply an even less realistic timing budget for the chip selected: 11.9 ns for 2 LUT, register and routing delays.

The equalization produces an area overhead; the prototype makes use of 182 CLBs, while a hand-made classical pipeline design could be fitted in less CLBs (a 13-stage, n=7 bit, array just requires 278 registers). However, the final number of CLBs in the wave version is

a consequence of two choices: regularity and categorical matching. Both strategies simplify the equalization task but require extra logic. In principle, the use of extra routing to balance LUT delays could reduce the number of CLBs occupied even less than quantity needed by the equivalent classical pipelines. But preliminary evaluations of this possibility suggest that the gap between simulation and actual delays is different for LUTs and routing. Then, the non-categorical equalization option would require to be performed a previous characterization of all chip elements.

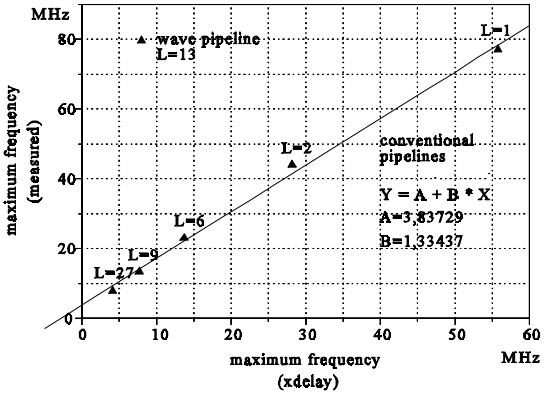

Fig.5: Simulated vs. measured frequency operation.
Conventional and wave pipelines (L=logic depth in LUTs)

In terms of power consumption, the wave pipeline technique diminishes synchronization power as well as datapath spurious activity (due to path equalization). For example, a non-equalized combinational

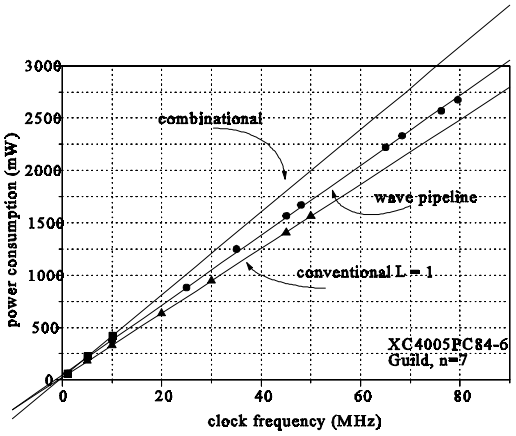

Fig.7 :Power consumption vs. frequency.
Conventional and wave pipelines

array outputs a maximum of 40 intermediate values between two consecutive results: ten times the number measured for the wave array (with output registers removed). Thus, both conventional and wave pipelining, allow to increase the speed and reduce power simultaneously [Boe95]. In Fig.6 the mW/CLB-Mhz figure for several arrays with different logic depth has been plotted. Both fine-grain pipelines and the wave pipeline prototype consume less power than the combinational version in the common range of operation, in spite of the hardware overhead. Nevertheless, the wave prototype consumed a little more than fine-grain pipeline arrays (Fig.7).

Finally, the main disadvantage of wave pipelines is its strong dependence on the power supply voltage. For example, at 84 Mhz (the middle of the last frequency band), changes in power supply below 4.88 V or above 5.13 V produced erroneous outputs, while in the middle of the six-wave mode band, 62.5 Mhz, variations between 4.75 V and 5.21 V can be tolerated.

4 Conclusions and future work

The feasibility of constructing wave pipelines using LUTs has been demonstrated. Even using commercial chips and tools, a low latency and high throughput figure has been obtained. Nevertheless, the efficiency of wave pipelining on FPGAs would be improved if extra buffers with the same delay of LUTs were included in future chips. In terms of power consumption, no advantage with respect to conventional pipelined arrays has been obtained; however, the power reduction could be significant for FPGA architectures in which synchronization fraction dominated the power consumption.

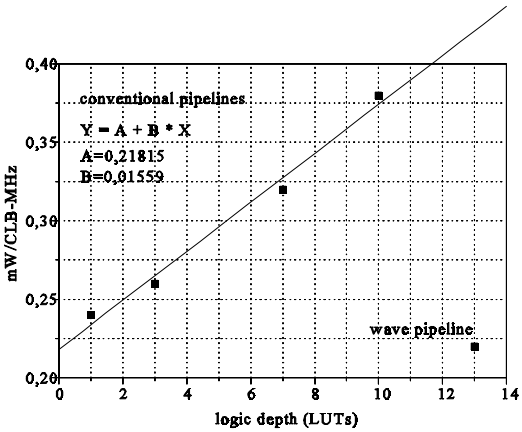

Fig.6: mW per CLB-MHz figure vs. logic depth.
Conventional and wave pipelines.

Current research includes the development of a clock-skewed [Gra93] wave pipeline version. In this case, the clock edges are delayed in the same way as the data. It eliminates the dark bands of the circuit, but also allows resynchronization of the data using intermediate registers. It leads to classical pipelines in which each stage run in the wave fashion. Preliminary evaluation suggests that, for the same chip utilized in this experiment, a speed up to 100 Mhz can be obtained.

Acknowledges

This work has been supported by the CICYT of Spain under contract TIC92-0083. The authors wish to thank Seamus McQuaid, and anonymous referees for their constructive comments.

References

[Boe95] E. Boemo, G. González de Rivera, S. López-Buedo and J. Meneses, "Some Notes on Power Management on FPGAs", *Proc. Fifth Int. Workshop on Field Programmable Logic and Applications*, pp.149-157, Oxford (U.K.), Eds: W. Moore & W.Luk. Berlin: Springer-Verlag 1995.

[Bur94] W. Burleson, c. Lee and E. Tan, "A 150 MHz Wave Pipelined Adaptative Digital Filter in 2µm CMOS", *VLSI Signal Proc. VII*, Ed. J. Rabaey, pp.296-305. New York: IEEE Press, 1994.

[Cot65] L. Cotten, "Circuit Implementation of High-Speed Pipeline Systems", *Proc. Fall Joint Computer Conference (AFIPS)*, 1965.

[Fly95] M. Flynn, K. Nowka, G. Bewick, E. Schwarz y N. Quach, "The SNAP Project: Towards Sub-Nanosecond Arithmetic", *Proceeding 12th Symposium on Computer Arithmetic*, pp.75-82. IEEE Press, 1995.

[Gho95] D. Ghosh y S. Nandy, "Design and Realization of High-Performance Wave-Pipelined 8x8 Multiplier in CMOS Technology", *IEEE Trans. On VLSI Systems*, Vol.3, nº1, pp.36-48, March 1995.

[Gra93] C. Gray, W. Liu and R. Cavin, *"Wave Pipelining: Theory and Implementation"*, Kluwer Academic Publishers, 1993.

[Gui69] H Guild, "Fully Iterative Fast Array for Binary Multiplication and Addition", *Electronic Letters*, pp.263, Vol.5, Nº12, June 1969.

[Kla94] F. Klass and M. Flynn, "A 16x16-bit Static CMOS Wave-Pipelined Multiplier", *Proc. ISCAS 94 Conf.*, pp.143-146. IEEE Press 1994.

[Liu94] W. Liu, T. Gray, D. Fan, W. Farlow and T. Hughes, "A 250-MHz Wave Pipelined Adder in 2-µm CMOS", *IEEE Journal of Solid-State Circuits*, Vol.29, N°9, pp.1117-1127. Sept. 1994.

[Now94] K. Nowka y M. Flynn, *"Wave Pipelining of High Performance CMOS Static RAM"*, Tech.Rep. nº CLS-TR-94-615. Stanford U. January 1994.

[Now94] K. Nowka and M. Flynn, *"Environmental Limits on the Performance of CMOS Wave Pipelined Circuits"*, Tech.Rep. nº CLS-TR-94-600. Stanford University. January 1994.

[Xil94] "The Tilde De-Mystified", *The Programmable Logic Array Data Book*, Second Edition, pp. 9-3. Xilinx Inc. 1994.

Notes:

The new address of Eduardo Boemo is:

School of Computer Engineering
Office B-430
Universidad Autónoma de Madrid
Ctra. de Colmenar Km. 15
28049 Madrid - Spain

e-mail: Eduardo.Boemo@ii.uam.es
http://www.ii.uam.es/~ivan/
Telephone: +34 91 348 2261