Programmable real-time FIR-filter logic device

E. Boemo, F. Barbero, J. Faura, J. Jáuregui and J. Meneses

E.T.S.I. Telecomunicación, Universidad Politécnica de Madrid
Ciudad Universitaria. 28040 Madrid, Spain

ABSTRACT

This paper resumes the development of an integrate tool for designing high-speed, real-time, FIR-filter circuits. The system is composed of programmable IC and an associate software for filter response analysis, synthesis of coefficients, and circuit programming. The architecture is highly regular, easily expandable and its control is distributed. The chip can be programmed by a PC or by using an EPROM. The prototypes have been fabricated using the CMOS 1.5µ Standard Cell of ES2. Moreover, some heuristics about multipliers upgrated to CMOS 1µ - Cadence DFWII are resumed.

Keywords: FIR Filtering, Programmable Logic, Pipelined Arrays, Systolic Architecture, Standard Cells, Multipliers.

1. INTRODUCTION

Recently, some module generator for FIR Filters have been published. Starting from a frequency response specification, these systems generate a netlist compatible with VLSI CAD tools and libraries. For example, a FIR filter compiler[1] compatible with *Mentor*, that synthesize different architectures depending on the speed required has been developed. The designer can obtain information about area, number of transistors, consumption and the maximum frequency operation of each architectural option. A similar tool[2] for *Mosis* and *Cadence*, that include directives for an efficient floorplanning and routing has been also published. These systems make the designer task easier by permitting the implementation of optimized circuits starting from a high level specification. Thus, the designer can focus his or her attention on system level problems, forgotten hardware details. However, there still exist several implementation barriers: the user must have a VLSI design tool and have microelectronics background, he or she must be able to afford prototype prices and the chips admit neither modifications nor reconfiguration.

In this paper is presented a system for fast prototyping of FIR filters based on the fundamental ideas of FPGAs: to use a standard blocks with programmable elements. Thus, the design cycle is shortened, the prototype cost is reduced, and the scope of users is not limited to only VLSI designers. The system is composed of a fine-grain-pipelined programmable integrated circuit, and two programs: Synthesis of Coefficients and Chip Configuration Control. The loading of the coefficients on the chip is done by a bitstream from a personal computer or by using an EPROM plus external hardware.

2. CHIP FEATURES

In order to implement a FIR filter it is necessary to execute the following computation over a data stream:

$$y(n)=\sum_{k=0}^{N-1} h_k \cdot x(n-k) \qquad (1)$$

This problem, analogous to matrix-vector multiplication, presents a set of characteristics suitable for pipelined array processing: a) it is limited by computation: the processor must perform N multiplications and acummulations between two consecutive data (this allows to overcome PCB or pads bandwidth bottleneck); b) it does not present feedbacks; c) the operations are fixed and data independent (this attribute simplifies the control of the process); and d) the long dataflow makes the initial latency of the array bearable.

The possibilities of mapping eq.(1) on hardware are multiple and vary from a one-multiplier processor to an N-multiplier array. In general, the choice of an N-multiplier processor is in accordance with a bit-serial approach, due to the inherent area efficiency of these kinds of circuits. From the point of view of functionality, the design of FIR filter chips can be oriented in two different directions: programmable or fixed-coefficients. The first category allows the scope of applications to be extended, as well as admits dynamic reconfiguration. The disadvantage is a hardware overhead. On the contrary, the second category has a limited application field but allows the designer to do exhaustive hardware optimization, by simplifying the logic circuit in accordance with the peculiarities of the set of coefficients.

In this work a bit-parallel approach has been selected, with a one-multiplier architecture and reprogrammability capacity. The architecture is similar to the proposed for L. Rabiner[3]. It has one multiplier accumulator and two circular buffers: one for data samples and the other for coefficients. In this architecture, the circuits must operate internally to a frequency N times bigger than that of the data sampling frequency. In order to obtain high throughput, the multiplier has been fine-grain pipelined and the output data skewing of this block has been coupled with the input data skewing of a pipelined ripple-carry adder. Thus, the extensive use of pipelining has allowed the maximization of the chip bandwidth. Other one-multiplier FIR filter example has been published by Roncella et al[4].

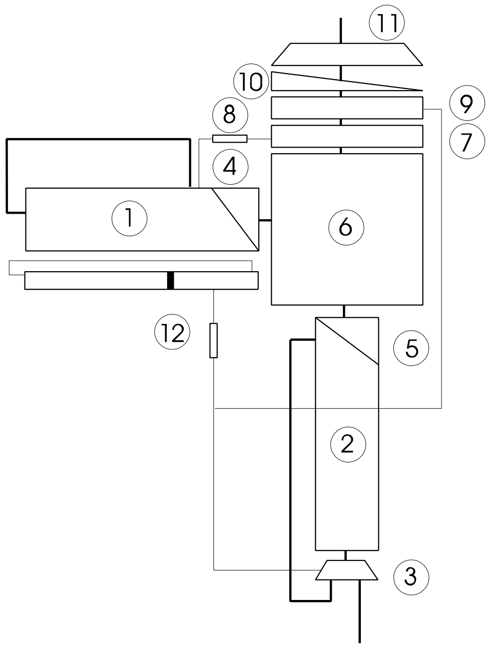


Fig.1: Block Diagram of the Filter.

Block diagram of the circuit is shown in Fig.1. In this prototype, the number of coefficients that can be selected are between 4, 8, 12 and 16. The implementation has been oriented to video applications where high speed and few number of coefficients is usual[5]. The principal modules of the chip are:

1. Coefficient Circular Buffer: It is composed of segments of FIFOs interconnected by a programmed multiplexer. Thus, it is possible to modify the buffer size to 4, 8, 12 and 16. The coefficients are represented in a 7-bit+sign format.

2. Data-Sample Circular Buffer: Similar to the buffer described above. The data samples are represented in an 8 bit unsigned format.

3. Input Data Multiplexer: This unit lets a new datum enter to the buffer at the end of every computation cycle.

4. Coefficient Skewing Registers: This register chain produces the necessary input skewing of each coefficient bit. This permits processing in parallel slices of different samples into the array.

5. Data Skewing Registers: Similar to the one described above, for input data samples.

6. Pipelined Multiplier Core: This module is based on McCanny architecture[6]. This module operates with unsigned data.

7. Two's Complement Converter: This module converts the multiplication result into 2's complement. It is controlled by the sign bit of each coefficient.

8. Sign-Bit Delayer: This chain of registers synchronizes at the input of the accumulator the sign bit of each coefficient with the corresponding multiplication result.

9. Pipelined Accumulator: It is a 20-bit long unit and operates in two's complement.

10. Deskewing Registers: Bank of registers in order to arrange the slices of results in bit-parallel.

11. Output Multiplexer: This block allows outputing and registering of 12 successive bits of the result, and thus it adjusts the precision in accordance with the particularities of each filter design.

12. Control of Processing: The control is distributed and straightforward extensible. It consists of an additional token bit that circulates in the coefficient buffer. Thus, the end of each processing cycle can simply be determined. This token is used to load a new datum, output the result, and clear the accumulator.

All the modules of the pipeline are balanced: the logic depth has been adjusted in order to maintain the maximum delay of every stage shorter than a multiplier cell delay. The design has been implemented on the Cadence 2030, using the 1.5µ Standard Cell of ES2 (*European Silicon Structures*). Implementation results are depicted in Table I. Placement and routing has been done automatically. Simulation results indicate a typical internal frequency operation over 100 MHz (the prototype's functionality has been verified to 50 MHz, the maximum frequency of our chip tester).

Nº Transistors:	35712
Nº Equivalent gates:	8928
Nº Standard Cells:	1849
Area (mm^2):	36.5
Nº Pads:	33
Encapsulate:	DIL 40

Table I: Implementation Results

The IC design includes some facilities for testing. If the circuit is configured in "Test Mode", both accumulator and output multiplexer blocks become transparent[A]. Simultaneously, both data and coefficient buffers are configured as FIFO (that is, they do not recirculate data). Hence, a sequence of test vectors can be used for test the multiplier. In this mode, a fault coverage over 93% has been easily obtained.

3. FIR FILTER SYNTHESIS TOOL

FIRFIL, the synthesis program, is based on Remez's algorithm[7]. The designer must enter the sample rate, the filters bands (value and weight), and the number of coefficients. The program outputs the coefficient values and ideal module/phase response. Then, specifying the numerical precision, the program generates the quantified coefficient and the real module/phase response for the number of bits selected. Each design can be store on disk, edited, printed, etc. All the features of the program are independent of the filter chip for the sake of enlarging the application scope. The particularities of the configuration has been grouped in *PROGFIR*, the other part of the software. The programs have a complete user interface and make use of all advantages of Windows 3.1. In Fig.2 some examples of the tool are shown.

[A] These improvements has been suggested by Prof. Carlos López-Barrio.

The filter data are entered into the chip via a bit-serial communication. The circuit has two dedicated inputs lines: *ConfigIn* and *ConfigClock*. The configuration load order is as follows: circular register length, output multiplexer setup, and finally coefficient values.

The configuration bits are loaded in a shift register, whose input is *ConfigIn* and the output is *ConfigOut*. Some bits are parts of coefficient values; meanwhile others control the configuration. In order to reduce the complexity of programming control, every configuration bitstream has a fixed length independent of any filter specification. The watchword during the design of the system has been to minimize hardware complexity.

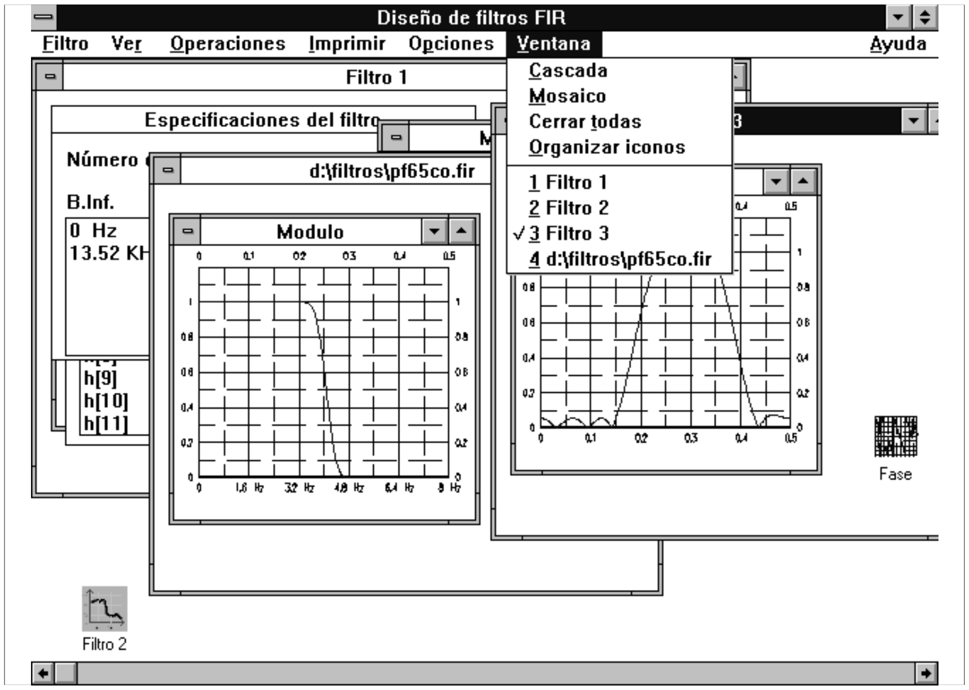


Fig.2: The Firfil Tool.

The configuration is done by connecting the outputs *D0, D1* and *D2* of the PC parallel port to the chip input *ConfigIn, ConfigClock* and *Config* respectively. The PC input port *ACK* is connected to *ConfigOut*. During programming, *PROGFIR* sends the configuration bitstream two times and collects the first configuration bit chain via *ACK*. Thus, the software can detect possible programming errors.

4. SIGNAL PROCESSING ASIC DESIGN HINTS

In order to extend the scope of this work, in this section, an exploration of pipelined array multiplier upgrated to CMOS 1µ ES2 Standard Cells - Cadence DFWII is resumed. In particular, the Guild[8], Hatamian[9] and McCanny[6] topologies have been analized, that allows common situations on bit-level paralellism to be reproduced. The results can be useful to FCCM researchers interested on including affordable ASIC in their machines.

It is well-known that, in principle, on breaking a combinational circuit into N balanced stages, a throughput N times bigger than the original circuit will be obtained. However, in practice, technological factors not only limit the improvements in performance to values more modest than N, but also increment the size (price) and complicate the synchronization. A primary parameter for the analysis of pipeline multipliers is the logic depth or granularity[10] β defined as the maximum number of elemental cells (an AND gate plus a two bit full-adder) between consecutive register lines.

The first point to analyze is the area-speed trade-off. Although ideal speedup of 16-bit β=1 pipelined arrays with respect to the combinational version should be over 31, actual maximum values obtained were between 8 and 13 times[B]. Register and net delays establish a lower bound for stage delay, limiting the throughput even for an infinite number of stage partitions. For this technology, using automatic placement and routing processing, the maximum rate (simulation results) for a β=1, 16-bit, pipelined multipliers is near 150 MHz. However, at this speed, off-chip load capacitance values over 100 pF became critical. Thus, before designing a high speed ASIC, the future load assigned to the chip output pads should be estimated.

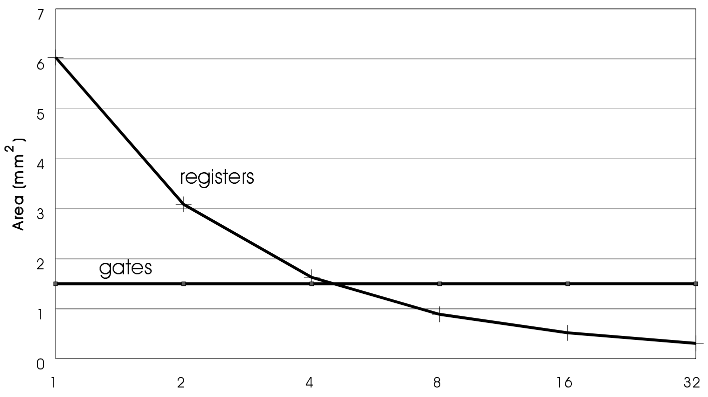


Fig.3: Register and gate area versus pipelining degree.

Moreover, the extra cost of pipelining is high. For example, Fig.3 shows area occupied by gates and registers versus the granularity β for a 16-bit Hatamian array (the number of registers depends on the degree of pipelining, meanwhile the number of gates is constant). Thus, the speedup of array multipliers follows a law of diminishing returns: when the pipeline grain is small (ie: the EP and register area-delay are close), an additional reduction of granularity increases the area but does not produce a corresponding increment in speedup. So, the obtainment of a high throughput leads to law of diminishing returns. For example, starting from the 16-bit β=16 version, it is possible (pipelining with β=8) to increase the speed by 88 % spending just 6 % of additional core area. A new partition (β=4) allows a speedup of 63 % with respect to β=8, but has 34 % extra area cost. For the β=2 step, the numbers are 63 % and 38 % respectively; and finally, for β=1, the transformation is expensive: a relatively modest 33 % additional speedup has a 68 % extra cost.

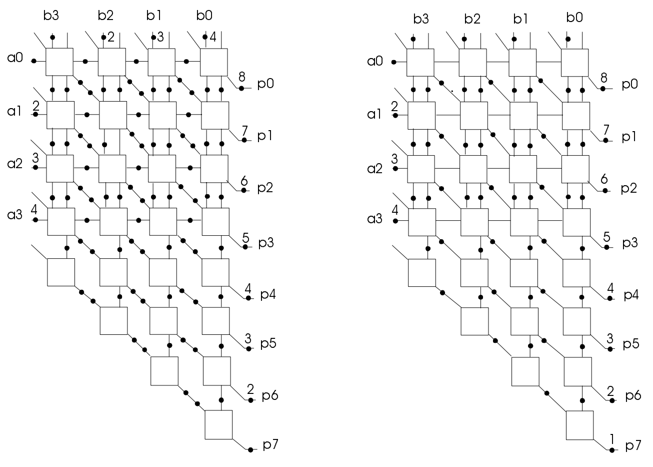
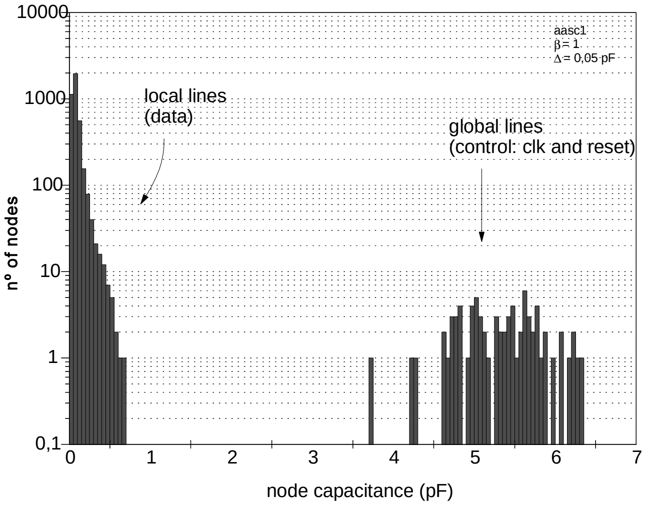


Fig.4: Global versus local communication between cells.

[B]. These bad results should not be discouraging: Henry Ford divided the Model-T motor assembly into 48 pipelined operations[11] and just obtained a speedup by a factor of 3.

Other important aspect to discuss is the utilization of local or global communication. In pipelined multipliers it is possible to broadcast an operand into the array multiplier (global communication) or to use just local communication between cells. The first option allows latency and size to be reduced, but also disminishes the throughput due to the increment in node capacity, while the reverse occurs for local communication. These differences can be quantified by comparing the arrays depicted, in Fig.4 (each register is represented with a dot, and k-registers in cascade are represented by a k over the ● symbol). Both circuits[6,9] have the same structure, but different pipeline strategies.



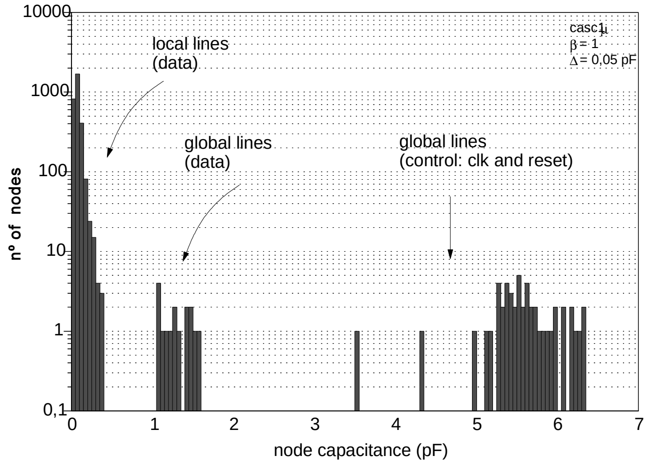


Fig.5: Node capacitance histogram. Local communication (above), global communication (below).

For example, for n=16 bits and β=1, the local communication array presents a throughput 25 % higher than the global one, but the latency increment is 15 additional clock cycles; the core area is 41 % higher; and the number of registers 43 % higher. The better speed of the local communication array is a result of the lower capacitance of their data nodes (Fig.5). For example, the worst node capacitance in this array is 0,58 pF versus 1,55 pF for the global one, a difference that produces an increment of 1 ns on the clock period, a significant value for frequencies over 100 MHz.

In the arrays described above, the simplest mechanism of synchronization is the use of a single-phase clock. However, the quantity of registers (for 16-bit fine-grain pipelined arrays is easy to reach more than 2500 registers) makes it necessary to pay attention to clock distribution. The effect of placement-routing process on clock distribution can be observed in Fig.6. It shows the final post-layout unbalance (in pF) on clock lines versus number of registers (NR) for different arrays. The starting point was a careful prelayout clock tree design with unbalances between branches below 2 registers and maintaining a load near 90 registers per buffer in any case. The unbalance (pF) vs NR can be transformed in skew (ns) versus NR by multiplying for the Δtp parameter of the selected clock buffer: the worst-case resultant value must be less than the minimum register propagation delay in order to prevent double-clocking in skewing/deskewing registers. Despite the warnings of self-timed synchronization designers, by using the above data, it can be demonstrated that single-phase clock is still a safe timing scheme for affordable chip sizes.

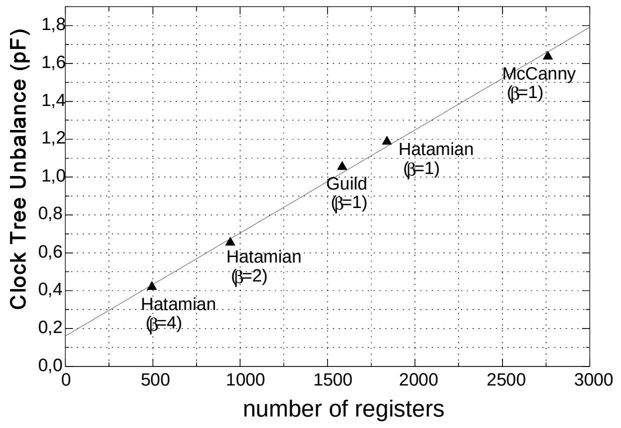


Fig.6: Post-layout unbalance (in pF) on clock lines versus the number of register, for different arrays.

Finally, in standard cells the designer can choice between pipelining or parallelism in order to speedup a circuit. For example, a β=1 pipelined 16-bit Guild multiplier has a typical speed near four times faster than the library (module generator) multiplier but the core area is 6.6 times larger. Then, an immediate question is: in order to obtain a high throughput, is it more efficient to use four library multipliers running in parallel or just one pipelined array multiplier? The key to the answer is the cost of the interconnections. The area-time overhead of multiplexing/demultiplexing four data buses must be estimated. Note that the library multiplier delay grows linearly with n while pipelined array delay is quite independent of n; however, pipelined array area is a stronger function of n, and also the circuit presents an excessive latency. For the 16-bit case, the core area results were: 9.7561 mm^2 for the pipelined array versus 12.1053 mm^2 for the "classical" parallel circuit. Additionally, pipelining also reduces power consumption[12] by avoiding the generation and propagation of glitches, that constitute a significant percentage of node activity in array circuits.

5. ACKNOWLEDGES

This work has been granted by the CICYT of Spain under contract TIC92-0083. CAD tools, silicon and platform have been provided by EUROCHIP. The authors wish to thank Seamus McQuaid for his constructive comments.

6. REFERENCES

1. E. Bidet, C. Joanblanq y P. Senn. "GENFIR: An Integrated VLSI FIR Filter Compiler". *Proc. European Conference on Design Automation*. pp.466-471. Paris 1993.
2. R. Jain, P.T. Yang y T. Yoshino. "FIRGEN: A Computer-Aided Design System for High Performance FIR Filter Integrated Circuits". *IEEE Trans. on Signal Processing.* Vol.39, Nº7, pp.1655-1668. July 1991.
3. L.R. Rabiner y B. Gold. "Theory and Application of Digital Signal Processing". Prentice-Hall. 1975.
4. R. Roncella, R. Saletti, P. Terreni y D. Piatelli. "Application of a Systolic Macrocell-Based VLSI Design Style to the Design of a Single-Chip High-Performance FIR Filter". *IEE Proceeding - G*, pp.17-21, vol.138, nº1. February 1991.
5. G. Privat y L. Paris. "Design of Digital Filters for Video Circuits". *IEEE J. of Solid-State Circuits*, pp.441-445, Vol. sc-21, Nº3. Jun 1986.
6. McCanny J.V et al. "Completely iterative, pipelined multiplier array suitable for VLSI". *IEE Proceedings*. pp.40-46. Vol.129,

Part G, Nº2. April 1982.
7 Mc Clellan, Parks and Rabiner. "A Computer Program for Designing Optimum FIR Linear Phase Digital Circuits". *IEEE Trans. on Audio and Electroacoustics*. December 1973.
8. .H. Guild, "Fully Iterative Fast Array for Binary Multplication and Addition", *Electronic Letters*, pp.263, Vol.5, Nº12, Jun. 1969.
9. Hatamian M. and G.L.Cash. "A 70-MHz 8-bit x 8 bit Parallel Pipelined Multiplier in 2.5-um CMOS". *IEEE Journal of Solid-State Circuits.* Aug. 1986.
10. C. Hauck, C. Bamji and J. Allen, "The Systematic Exploration of Pipelined Array Multiplier Performance", *Proc. ICASSP 85*, pp.1461-1464. New York: IEEE Press, 1985.
11. "*Crónica de la Técnica*". Plaza & Janes: Barcelona, 1989.
12. E.Boemo, G. Glez de Rivera, S. López-Buedo and J. Meneses, "Some Notes on Power Management on FPGA-based Systems", *Proc. Fifth Int. Workshop on Field Programmable Logic and Applications*, Oxford, U.K. Eds: W. Moore y W.Luk. Berlín: Springer-Verlang 1995 (in press).