High Performance Data-Path Synthesis via Communication Metrics

Andrew Seawright and Forrest Brewer

ECE Dept. University of California, Santa Barbara, CA 93106

Abstract

We present a novel decomposition of the high-level data-path synthesis problem taking as input scheduled procedural behavior and outputting geometrically placed and optimized linear data-paths. The technique uses communication based analysis to allow simultaneous synthesis of the placement, bindings, and RTL topology for high performance linear data-paths. In effect, the procedure creates an architecture and a layout with specified performance and cost constraints.

1.0 Introduction

System level computer-aided-design tools for the most part synthesize all levels of the architecture before layout is generated. In particular, the circuit topology is fixed as a netlist before geometric placement and routing. This is especially true for ASIC's built using standard cells or gate arrays. For regular structures such as computer data paths, substantial performance and area efficiency is achieved with the use of regular layout styles. Bit-sliced linear data-paths are important commonly used topologically "simple" implementations and many layout assemblers exist [Mar86] [CoSh91]. These typically require the user to either pre-specify or interactively generate: the interconnection topology, bus driver sizing, and possibly, the explicit layout ordering.

If, instead, the design is specified as a behavior at a higher level of abstraction, it might be possible to determine a circuit topology which has the property that it is efficient (in both area and time) when mapped to a particular chosen physical topology. With few exceptions [McFa86] [BrPS90] [WePa91], current efforts in high level synthesis, however, aim at producing solutions which are "minimal" relative to abstract costs. Unless these costs are correlated to the desired physical implementation, the resulting designs will not be optimized in the physical implementation [BPLS91]. This is particularly important in high speed designs, where due to scaling, distributed RC effects can cause severe performance degradation in long busses [Soji88] [Bako90].

We propose a strategy to enhance the performance of the design of linear data paths by de-coupling the high-level synthesis problem in a novel way. First, analyze the communications required from the atomic data-flow operations to construct a globally efficient linear organization, while simultaneously performing the bindings of operands to registers, and operations to specific function units. Second, use this output placement and bindings to construct an optimized interconnection binding for the data path. This decomposition has several advantages:

1. In the first step, the placement of the data path components and the bindings of both operands to registers and functions to function units are simultaneously determined. This allows the direct optimization of the circuit topology to optimize the speed and size of the resulting physical structure. Since the placement is also determined in this step, the models used to estimate the timing and spacial constraints can be far more accurate than without this placement information.

2. In the second step, the interconnection binding (the mapping of communications onto busses, multiplexors, and other communications components) is determined using the placement and component mappings of the previous step to achieve an efficient physical interconnection network. Knowing this geometric information allows accurate grading of potential trade-offs during the interconnect generation. This also allows the fine-grain modeling of the interconnection as well. For example, bus repeaters can be automatically added to time-critical busses.

0-8186-2610-0/92 $3.00 © 1992 IEEE

3. Unlike timing driven placement techniques [MMS92], at the gate and netlist level, the new technique performs the placement from the behavioral viewpoint. When evaluating the performance of the design, the timing is induced directly from the data-flow required by the behavior, and measured with respect to the binding and placement of the potential design. In this way, the RTL topology of the circuit is dynamically optimized as well as the placement. This binding freedom extends to the allocation of the function units as well as register and interconnection mapping. False timing paths which are often a problem in netlist level timing analysis are not manifest in this model since only the valid data communication flows are analyzed[1].

4. Several unique optimizations are possible in this scheme due to the ability to measure their direct impact on area and performance using the geometric information. For example, operands may be transferred between registers in such a fashion as to ensure that they are close to specific function units in a critical future cycle. This type of trade-off would be virtually impossible to evaluate without the geometric information.

This work focuses on the first step of the decomposition and is encompassed in the timing driven communication placement tool called TDCP. Figure 1 below shows the relationship between TDCP and its environment. The timing driven communication interconnection tool, TDCI, performs the second part of our decomposition. TDCP's output is TDCI's input. TDCI's output is assembled to obtain the final layout of the data path

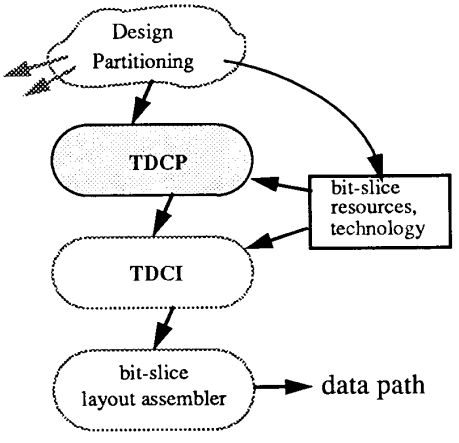

Figure 1: System

1. We assume the control transfer graph specifies only the valid transitions.

2.0 Related Work

The binding of operations to hardware resources focusing on determining an efficient data path allocation was done by Devadas and Newton [DeNe89]. In this work, simulated annealing was applied to perform bindings, however geometric constraints and timing performance were not modeled. Data transfer communication through interconnect was approached in a novel way with the use of transport productions in [LyElG90]. This model for communication is quite general, again however, performance with respect to geometry was not possible. Data path geometry has been modeled in many "Silicon Compilers", particularly for DSP applications. Examples include [Raba87] and [JaYa90]. These systems presuppose a fixed architecture or a specific application. Fasolt [Kn90] optimizes RTL architectures with respect to a floorplan layout. Feedback-driven transformations are applied to the netlist and a new floorplan generated. Mentioned earlier, a large body of work has been done with respect to timing driven layout at the gate and netlist level. An overview of this field is [MMS92].

3.0 Communication Model

Our optimization focuses on the communications between the data flow operations present in the scheduled behavior. For it is these communications that impose the most basic constraints on the timing and layout of the design. We shall further describe our model through example. Figure 2 illustrates an example control flow graph for a design. Each node represents the register transfer operations taking place in a particular clock cycle. The arcs between the nodes represents potential state transitions. Thus, it is actually a control flow transfer graph. Our model supports arbitrarily complex control flow.

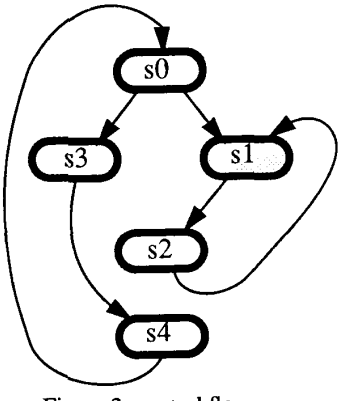

Figure 2: control flow

The data flow operations for a single node in this figure is examined in more depth in Figure 3. The nodes in the diagram represent data computations, such as arithmetic or logical operations. In the example there are two chained

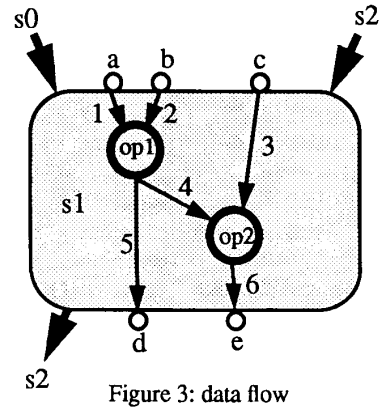

Figure 3: data flow

operations in cycle s1. The data flow interface of cycle s1 are the operands from predecessor and successor cycles, which are labeled (a) through (e) in the figure. The numbered data flow arcs represent the required virtual communications between data operands and operations. In order to execute the cycle s1 the operands must be bound to registers, the operations bound to functions units, and the virtual communications bound to physical interconnection resources.

The bindings of the operands and the operations are performed over all states in the control flow simultaneously with the linear organization of the data path in order to achieve globally efficient operand transfers over all potential state to state transitions. Figure 4. illustrates a particular linear arrangement of data path cells that lends to an efficient execution of s1. Note, the bindings of operand to registers must be consistent across control transitions. For example the operand binding mapping for the following transitions are the same: s0 to s1, s1 to s2, and s2 to s1. The notion that an operand may be bound to several registers at once is supported by the model.

The primary optimization goal is to minimize the critical cycle time of the design which is determined from the most constraining cycle. The most constraining cycle contains the critical path register transfer(s). The optimization metric is further enhanced by using a cost function that incorporates other attributes of the design such as aggregate wiring, tracks, and area. Several different cost functions may be specified depending on the optimization goal and layout model chosen. The types of cost functions include: A linear combination of specific costs, a prioritized cost scheme, and an ordered vector cost function containing the delay time of the n most critical paths through the design. Distributed RC bus models with and fan-in and fan-out loading are used in evaluating the critical paths. These models are a function of the linear organization, and the operand and operation bindings.

Recall, the key to this approach is the notion of an atomic communication graph. Each of the data flow communications can be represented by a series of arcs from the information sources to the information sinks. These arcs can be mapped directly onto the placement organization and the path density (representing minimal track allocations) length (representing minimal wire length allocations) and fan-out can be evaluated simply. This technique is enormously simpler than completing the interconnection analysis, yet allows determination of the minimal requirements for the physical implementation. However, in order to implement the above techniques a model for the geometric constraints of the data path is essential.

4.0 Data-Path Target Model

This section describes the target linear data path layout model used in both the TDCP and TDCI programs. We assume a linear bit-sliced style of layout where each slice represents one bit's functionality. For an n-bit wide data path n slices are stacked where each slice is typically the same or slightly different. The differences between slices arise for several reasons. Examples include: boundary conditions on the edges of the data path, the routing of signals from slice to slice, the routing signals out of the data

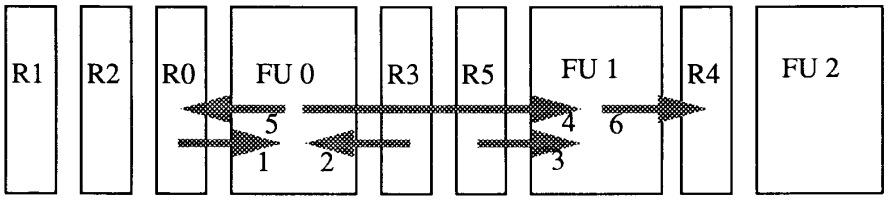

Figure 4: linear organization

path, and to accommodate selections on specific portions of signal busses. Since the n slices are stacked, we chose to model the ensemble data path by modeling one of the slices.

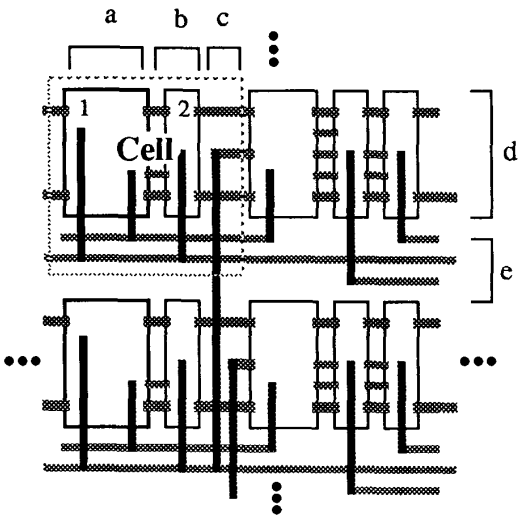

Figure 5: Bit-Sliced Layout

Figure 5, above illustrates the model by showing a portion of an example bit-slice data path. The regions denoted by (d) and (e) together represent the notion of a slice, thus the figure shows a region containing two stacked bit-slices. Each slice is composed of an abutment of cells horizontally, region (d), interconnected through the horizontal routing region, (e). An example cell is enclosed by the dotted box in Figure 5.

The physical hardware in each cell is divided into two portions. The first portion is the basic hardware that performs the cell's function and is assumed to be fixed in size. An example is an arithmetic or logic function component. The operations in the procedural data flow are bound to these cells. The second portion models an allocation of interconnect resources allowing for complex interface of the cell to the horizontal busses in the routing channel below. Specifically these interconnection resources are tri-state buffers. This second region is parametrized as function of the interconnection requirements of the cell. For example, if an output of the cell connects to a heavily used bus, a tri-state driver is required. On the other hand, if the output is a dedicated connection to another cell, no driver is needed. These two regions are illustrated above by (a) and (b) respectively. In the example, the (b) region contains one tri-state driver in the designated cell. Slice to slice interconnection and the routing of control signals is done in vertical channels neighboring the cells. For example the region marked (c). The physical area constraints of each cell can be abstracted as shown in Figure 6.

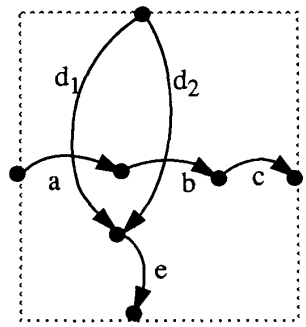

Figure 6: Bit-Sliced Cell Area Constraints

The area of the data path is simply the total length of the bit-slices multiplied by the width of the data-path. The width of the data path is sum of the widths of the slices. Thus routing channel (e) has a significant impact of the area efficiency of the data path. The number of tracks required by the busses packed into this channel is a metric of this efficiency.

The primary objective, however, is the timing performance of the data path. We use a RC timing model for the timing of data transfers through the busses. This model is based on the models in [Saku83] and [bako90] and is used to determine communication transfer time:

$$T_{transfer} = (R_d \cdot C_d + R_d + R \cdot l) \times (C \cdot l + C_L)$$

In the equation above, R_d and C_d are the resistances and capacitances of the active driver. The transmission length is l, and the distributed resistances and capacitances per unit length are R and C. The capacitive loading on the bus due to distributed receivers and inactive drivers is reflected in loading capacitance C_L. This capacitance represents the effective capacitance of the distributed loads and modeled at the far end of the transfer.

In the placement and binding process, the *explicit* interconnection is not generated and the intricate wiring is not known, however, bounds for routing and communication transfer are used. This is powerful enough create efficient binding and placement that can be exploited by the TDCI

interconnect generation program. In TDCI the timings of communications are re-evaluated in the process of constructing the interconnect. The model for the bit-sliced data path model presented above is abstracted in the following ways in the TDCP program:

1. Cells are modeled as fixed in size based on a nominal interconnect allotment.
2. Operations bound to the cells are modeled as occurring at the centers of the cells.

5.0 Implementation

The TDCP tool has been implemented utilizing the above techniques to optimize linear data paths of arbitrary complexity for high performance synthesis. The code is approximately 6000 lines of C++. The optimization algorithm is a probabilistic hill climbing algorithm very similar to simulated annealing [KiGV83]. To facilitate this algorithm a move-set was developed and tuned. The moves are divided into several classes: placement adjustment moves, and operation and operand binding moves. The moves in each class are sub-divided further. Sparse bit matrices are used to implement maps containing the binding information. The sparse bit matrices allow rapid evaluation of the design during the evaluation process. Care was used in designing the data structures such that he complexity growth in the data path evaluation routine as a function of design size would be linear.

The control flow of the design is represented internally as a directed graph, as in Figure 2. Each node of this graph points to the data structures used to represent the data flow in each cycle. These data structures consist of an acyclic directed graph representing the data flow and pointers to three binding map objects. The binding maps represent: the bindings of the operations in the cycle to bit-sliced function units, the binding of the input operands to registers, and the binding of the output operands to registers. The operand binding maps are consistent across state to state transitions. For example the pointer to the output binding map for a state is the same as the input operand map pointer for its successor states, as well as for all the successor states predecessors' output operand maps. Another map relates the bit-sliced function unit instances in the design to a linear data path ordering. In the evaluation process of the annealing algorithm, the design is timed with respect to the binding and placement mappings. During this process the binding and placement information is used to evaluate the times for data transfers through busses and for estimating the track density along the data path. It is possible, during the course of optimization, that a function unit or register not be mapped to any operation or operand in the entire schedule. In this case this bit-sliced cell "disappears" and is not present on the data path and does not influence the evaluation of the design.

The input file for TDCP consists of blocks code representing the register transfers for each cycle, as well as the control flow from block to block. Each block contains a definition of the operand parameters passed on input and output of the block. Next is specified the operations to be performed in the cycle with respect to the defined operands. The syntax of the block is a subset of VHDL. Files containing technology data, and the bit-sliced function unit resources are read in during the initialization process as well.

The output of TDCP is a description of the ordering of the linear data path cells and specification of the sets of disjoint communications between the bound cells on the data path. In this respect this file can be viewed as a pseudo-netlist where the communications are the nets. This output file is read in by the TDCI program.

6.0 Limitations of TDCP

There are several limitations to be stressed regarding the present version of TDCP. First the current version of TDCP has an abstract model of the controller. Only a fixed set-up delay time, at the start of each cycle, can be modeled presently. For data dominated examples this may not significant, however, for control dominated designs the current models are lacking. Related to this point, is the fact that controller complexity is a function of the bindings. This issue is not presently modeled but should be addressed. With respect to the timing of the data flows along the data path, we do not model the fact that the timing of the different bit-slices differ in general (bit level timing). In TDCP all of the bit-slices are modeled as one abstract slice. For example, this means the delay values for function units must be overly conservative. Finally appropriate models for power consumption need to be incorporated into the evaluation of data path.

7.0 Example

An example design was studied to test the initial version of the TDCP program. This example is a particular scheduled implementation of the HAL behavior [PaKnG86]. Since our target implementation is a bit-sliced data path we chose to use realistic bit-slice resources. The multiplications in our implementation of the example use bit-sliced pipeline stages. In addition each of these pipeline stages are able to implement different steps of the multiplication. The pipeline stages consist of cascaded

carry save adders plus bit selection and AND functions. Each stage can perform one of the first three portions of the four step multiplication (operands are 12 bits wide). This pipeline stage can be thought of as a micro ALU. A final propagate adder is required in a fourth step in order to complete the multiply. These final adder resources in the multiplier pipelines are shared among the rest of the additions and subtracters in the schedule. Table 1 lists the resources allocated for this example, and their characteristics. Table 2 contains the technology data used by TDCP for this example

Table 1: Bit-slice FU resources

unit	no.	length (μm)	time (nS)
adder / subtracter ALU functions: add, sub	2	50	5.1
three stage mul ALU functions: mul parts 1, 2, 3	4	220	2.5
comparator function: <	1	30	4.8
register functions: read, write	14	40	1.0

Table 2: Technology Data

parameter	value	units
R, bus	500	Ω / cm
C, bus	2000	pF / cm
R_d, driver	2000	Ω
C_d, driver	100	fF
C_L, median distributed tap C	300	fF / bus

This example, CSAHAL, contains a rich binding problem with respect to the four CSA ALUs, the two adder / subtracters, and the registers. For instance, any of the different multiply sub-functions can each be mapped to any of the four available multiply ALUs. Taking this viewpoint, TDCP optimization will assemble an efficient pipeline with respect to the entire HAL behavior, not just for each multiplication. All registers in the CSAHAL designs are explicit, there is no distinction between registers containing operands, pipeline registers, and registers holding temporary values.

TDCP's results for the CSAHAL example are tabulated in Table 3. The first row reflects the initial design with "sensible" register assignments and bindings but without any re-placement or re-binding. In the second row the optimization results with only placement moves enabled are given. This optimization is analogous to timing driven placement, but at the behavioral / communication level. The third row contains the data for TDCP with all binding and placement moves enabled. The critical cycle time column contains the data for the estimated cycle time for the designs. These times are the times of the estimated critical path(s) in the critical cycle(s). The track estimate is an *upper* bound for the number of tracks needed in the cycle(s) with the most congested transfers. Given this upper bound on the tracks in the congested cycles(s) this number is a *lower* bound on the tracks needed over all cycles. Formulating the track estimate this way improved the evaluation of the data path. The aggregate track estimate in the fourth column is the sum of all the tracks estimates of the individual cycles. This is another evaluation metric. An estimate for aggregate wirelength is tabulated as well. This cost value is calculated similar to the track estimates. Finally, the estimated length of the data path is in the last column. Recall, this number does not reflect the contribution due to the final interconnect generated by the TDCI program. The run time for the TDCP program for this example was less than 10 minutes on a Sun 4/110 workstation.

The results show the benefit of the simultaneous placement and binding optimizations in TDCP. These optimizations improved *both* the area and timing efficiency of the design example. The track and wiring improvements over the initial design were substantial, 47% and 63% respectively. Although the estimated time improvement was only on the order of half a nanosecond, we believe that after the interconnect generation this difference will be substantially greater. This is due to several reasons: First, the main objective of the TDCP program is to generate, for the TDCI program, placement and binding information that can be exploited in constructing high performance interconnect. Without optimized placement and binding information TDCI is constrained. Second, the performance of this kind of data path is also a function of its area efficiency and this effect is not modeled. Finally, the estimates for the RC delay of data transfers assume favorable circumstances. Thus these estimates are more accurate for the performance of a "good" design than for an un-optimized design. Another point concerning performance of the example, is the fact that a 1.57 mm data path is small enough that the quadratic RC delay is not a dominating factor.

A position-time plot generated by TDCP reflecting the behavior of the optimized CSAHAL example is illustrated

in Figure 7. This figure shows the activity of the data path over the entire schedule. The two quantities position and time, with respect to the data path, are orthogonally measured: Time vertically (increasing downward along the page) and position along the data path measured horizontally. The vertical divisions denote the lengths and locations of the registers and function units in the placement, and the horizontal divisions denote the clock cycles in the schedule. In this example the cycles proceed down the diagram in the order of loop execution.

8.0 Conclusion

This work has demonstrated the benefit, to high performance data path synthesis, of considering the geometric implications of placement and bindings before the interconnect generation. This decomposition of the high level synthesis problem is a novel one. We believe this technique to be valuable in the design of regular, high performance data path structures where the knowledge of the geometry and timing "opens the door" to the potential of new kinds of optimizations. On-going research in this area involves improving the algorithms and models used in TDCP, and the integration of TDCP with other tools - notably TDCI. The authors would like to acknowledge Chuck Monahan for his work on the TDCI tool, a parallel research project underway at UCSB.

9.0 References

[BrPS90] F. Brewer, B. Pangrle, and A. Seawright, "Interconnection Synthesis with Geometric Constraints," IEEE Micro'23 Workshop, November 1990.

[PBLS91] B. Pangrle, F. Brewer, D. Lobo, A. Seawright, "Relevant Issues in High-Level Connectivity Synthesis," Proceedings of the 28th Design Automation Conference, June 1991.

[Kn90] D. W. Knapp, "Feedback-Driven Datapath Optimization in Fasolt," IEEE International Conference on Computer-Aided Design, November 1990.

[LyElG90] T. Ly, W. L. Elwood, and E. F. Girczyc, "A Generalized Interconnect Model for Data-Path Synthesis", Proceedings of the 27th Design Automation Conference, June 1990.

[McFa86] M. J. McFarland, "Using Bottom-up design techniques in the synthesis of digital hardware from abstract descriptions", Proceedings of the 23rd Design Automation Conference, July 1986.

[Shoj88] M. Shoji, CMOS Digital Circuit Technology, Prentice Hall, Englewood Cliffs, NJ, copyright 1988.

[Bako90] H. B. Bakoglu, Circuits, Interconnections, and Packaging for VLSI, Addison Wesley, Reading, Mass 1990.

[Saku83] T. Sakurai, "Approximations of Wiring Delay in MOSFET LSI," IEEE Journal of Solid-State Circuits, Vol SC-18, No. 4, August 1983.

[KiGV83] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science, vol 220, no. 4598, May 1983.

[MMS91] M. Marek-Sadowska, "Issues in Timing Driven Layout," special volume: *Algorithmic Aspects of VLSI Layout*, Lecture Notes Series on Computing, Springer Verlog, to appear in 1992.

[Mar86] T. Marshburn et. al., "DATAPATH: A CMOS Data Path Silicon Assembler," Proceedings of the 23rd Design Automation Conference, July 1986.

[CoSh91] A. B. Cohen, M. Shechory, "Track Assignment in the Pathway Datapath Layout Assembler," Proceedings of the International Conference on Computer-Aided Design, November 1991

[DeNe89] S. Devadas, A. R. Newton, "Algorithms for Hardware Allocation in Data Path Synthesis," IEEE Transactions on CAD, Vol 8. No 7, July 1989.

[TsSi86] C. J. Tseng, D. P. Siewiorek, "Automated synthesis of data-paths in digital systems", IEEE Transactions on CAD, v. CAD-5, July, 1986.

[WePa91] J-P Weng, A. C. Parker, "3D Scheduling: High-Level Synthesis with Floorplanning," Proceedings of the 29th Design Automation Conference, June 1991.

[GrDe90] D.M. Grant, P.B. Denyer, "Memory, Control and Communication Synthesis for Scheduled Algorithms", Proceedings of the 27th IEEE Design Automation Conference, June 1990.

[PaKnG86] P. G. Paulin, J. P. Knight, and E. F. Girczyc, "HAL: A Multi-paradigm Approach to Automatic Data Path Synthesis," Proceedings of the 23rd Design Automation Conference, July 1986.

[Raba87] J. Rabaey, "CATHEDRAL-II: Computer-aided synthesis of digital processing systems," presented at the IEEE Custom Integrated Circuits Conference, Portland, OR, May 1987.

[JaYa90] R. Jain, et al, "FIRGEN: a CAD System for Automatic Generation of High Performance FIR Filters", Proceedings of the CICC, 1990.

Table 3: CSAHAL Results

optimization	est. cycle time	est. tracks	ag. tracks	"wire"	est. total length
initial	9.36 nS	15	85	69045	1.57 mm
TDCP (bindings fixed)	9.33 nS	9	54	28920	1.57 mm
TDCP	8.92 nS	8	47	25560	1.57 mm

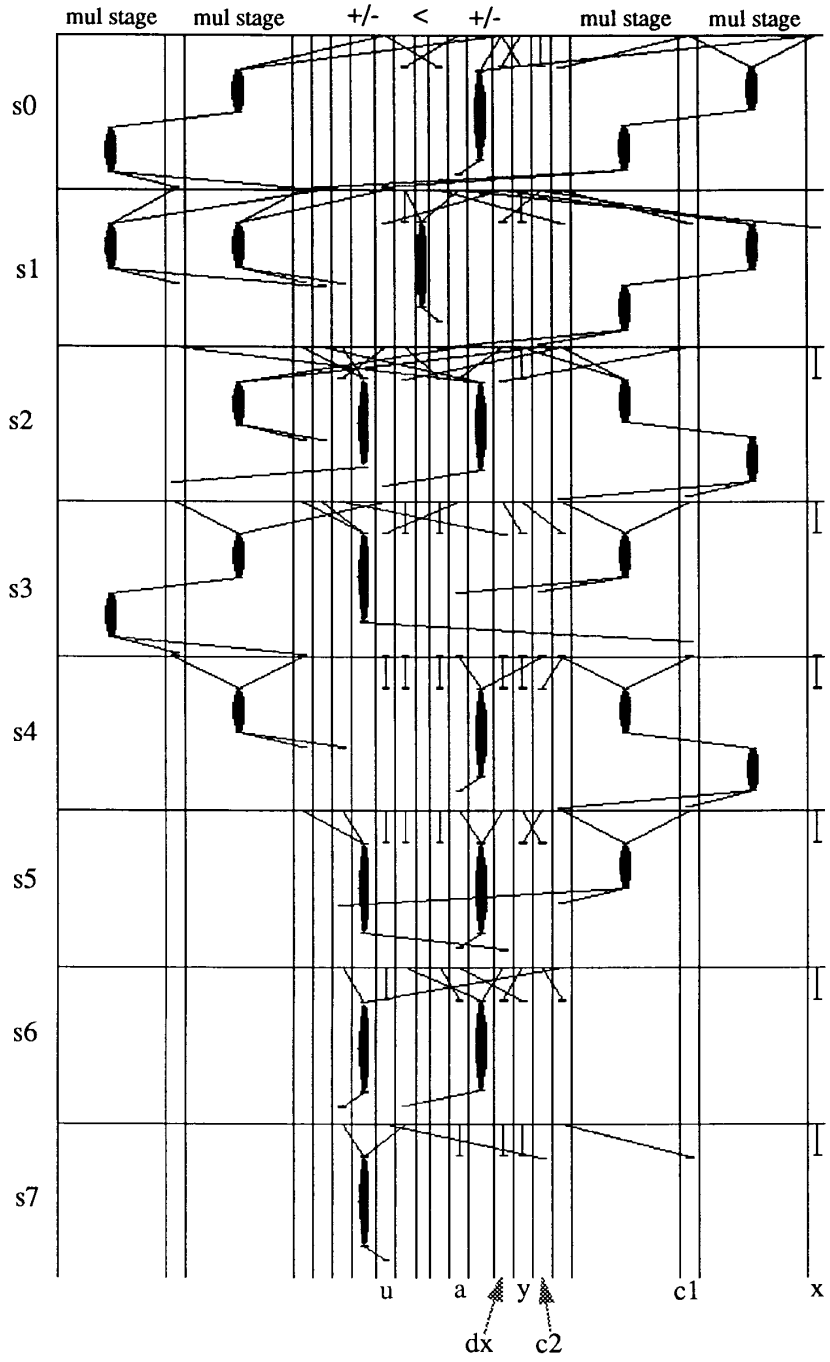

Figure 7: CSAHAL Position-Time Diagram