North-Holland
Microprocessing and Microprogramming 30 (1990) 117-124

Stepwise Decomposition in Controlpath Synthesis

A.J.W.M. ten Berg

Department of Computer Science, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands

A method is presented for the synthesis of the microarchitecture of controlpaths. This method is called stepwise decomposition. It focuses primarily on controlpaths of instruction set processors, however it is also applicable for more general Finite State Machine synthesis. Many of the current controlpath synthesis algorithms are based on a fixed microarchitecture, and an optimization of that microarchitecture. This stepwise decomposition method is able to synthesize microarchitectures in a range from a single PLA to multiple PLA/ROM configurations and optionally further down to hardwired, which makes it more flexible and better suited to a wider range of controlpaths than current synthesis methods. A sequence of decomposition steps, from coarse to detailed, is performed on the design to move it to the area of the design space where all constraints on space, floorplan and delay are satisfied. The method is currently implemented in APL.

1. Introduction

Many of todays high level controlpath synthesis systems are restricted to a fixed target microarchitecture, mostly a PLA configuration or a microprogrammed ROM. Furthermore, in most systems only the size parameter of the design is minimized, as in logic synthesis [1] and in state-assignment algorithms [2,3]. Comparisons based on benchmarks stress often only the number of literals or productterms. Therefore the generated designs will not always satisfy all the constraints which do exist in a 'real' design environment. Especially the floorplan related design parameters and delay are not taken into account in those synthesis systems. Controlpath synthesis by stepwise decomposition is developed for application in a 'real' design environment, with floorplan and delay design constraints. It is directed specifically to the design of controlpaths for instruction set processors. In that application the complexity of control spans a wide range, from very limited in RISC, up to very complicated for CISC processors. For this wide range current synthesis methods are of limited use. The stepwise decomposition method however is able to generate different microarchitectures to cover this wide range of controlpath complexities. This is realized by a sequence of decomposition steps, together with floorplan directed synthesis. The microarchitecture is determined by the design constraints. Furthermore this method assures inherently the highest possible design regularity, which design parameter is recognized important [4] for the design quality. First, the limits adopted are discussed in section 2, then the method itself is considered in section 3, followed by a description of the most important decomposition steps currently implemented in section 4. The last sections contain the results and conclusions.

2. Design model

First, a framework of constraints and assumptions within which the synthesis performs its task is to be defined. This framework is called a design model. This design model limits the large number of freedom degrees in the design while retaining a high design flexibility. Thus, the computational complexity is kept manageable. The general part is discussed in this section, parts specific for one decomposition step are found throughout the other sections.
First some definitions. A microword is the word available at one address in the microprogrammed ROM. This microword is divided into control fields. Each control field corresponds with exactly one functional unit in the datapath as shown in figure 1. A control field contains microoperations for its particular datapath functional unit only.

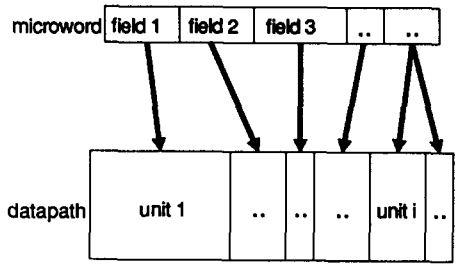

Figure 1. Control fields related to datapath units.

The wiring area in the controlpath is kept minimal by ordering the control fields in the microword equal to the ordering of the datapath functional units in the layout. Especially for complex datapaths with a large number of control lines, this saves much wiring area. This ordering

can also be applied for the individual control lines within each control field. However, in case of encoded control fields the ordering of control variables within a control field is hidden by the decoders and thus irrelevant. But, because our synthesizer is able to generate horizontal as well as encoded implementations, the ordering of all control variables in the microword is kept equal to the layout ordering of control terminals.

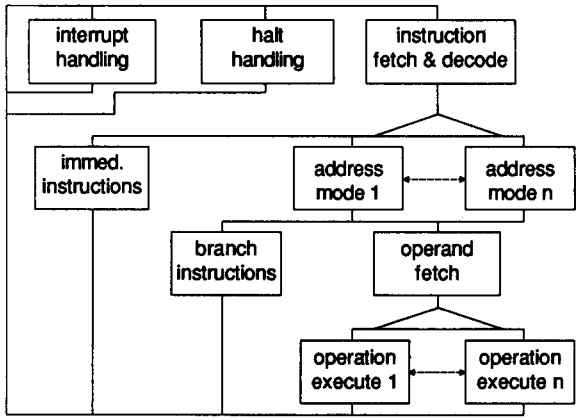

Figure 2. Typical state-graph for instruction set processor.

Furthermore a general structure is assumed for the state-graph of control function which is shown in figure 2. This structure is typical for instruction set processors [5]. The figure shows that state sequences are shared among instructions in principle [5,6]. For example state sequences needed for operand addressing are shared among instructions. Another typical structure in the state-graph are the few large 'fork' constructions, where selections are made between the different operations or address modes. The implications of this transition structure for the microarchitecture are made clear in the following sections.

3. Design refinement by stepwise decomposition

The principle of design refinement by a sequence of decomposition steps is feasible from several points of view. In contrast with previous decomposition methods it incorporates several decomposition levels or decision levels as shown in figure 3. This prevents that all design parameters have to be considered together at one decision level. By stepwise decomposition each design parameter is considered at the decomposition or decision level were it has most impact. A selection between alternatives in each decomposition step makes it possible to generate different microarchitectures to adhere the design to its constraints. This in contrast with other synthesis methods, which are mostly bound to one fixed microarchitecture. Such methods are not very flexible towards for example the floorplan area constraints. The decision whether or not to perform a next decomposition step is made before and independent of the comparison of the alternatives in the decomposition step (figure 3). This separation of design analysis phases and decomposition steps contributes to a structured and comprehensive design method.

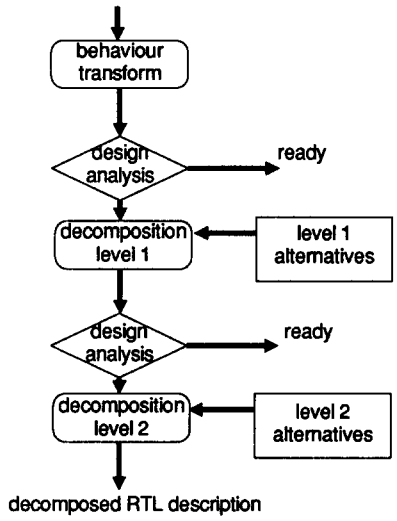

Figure 3. Decomposition decision flow.

Another advantage of stepwise decomposition is the inherent high regularity of the generated designs. When a design fits its constraints it is not decomposed needlessly into smaller components. The importance of a high design regularity was recognized already in [4]. The fewer different designed components and the larger the repetition of each of those, the more regular a design is called. A high design regularity improves the predictability of the final layout parameters in the synthesis phase. At last, by stepwise decomposition the computational complexity of the synthesis is kept limited, for each individual decomposition step does not have to consider the complete bulk of design details.

3.1. Behavioural description transformation

The first action of the synthesizer is to compile a behavioural description of the controlpath into a transition table. This behaviour can be specified in either a gateprogram format [7] or in a PLA format. The behaviour compiler replaces the control constructs in the gateprogram, typical for sequential programming, by completely specified conditional transitions. Initially a Moore machine type transition table is constructed. But the Mealy representation of the controlpath Finite State Machine is also generated and used together with the Moore representation throughout the decomposition process. The Moore state codes are applied as unique labels for the Mealy machine's transitions, which are required for the decomposition of Mealy machines.

4. Decomposition decision sequence

In this section the individual decomposition steps are discussed together with the algorithms involved. First we give a short overview of the decomposition steps incorporated in the current implementation of the method. Then, in the next sub-sections the individual steps are explained into detail. The stepwise decomposition principle implies a simple root configuration. The simplest configuration possible is a PLA with state feedback register. This configuration is then analysed with regard to the design constraints. If these constraints are not satisfied the root configuration is decomposed into a two PLA configuration to which a counter is added if useful. One PLA implements the Finite State Machine's (FSM) transition function and the other PLA implements the FSM's output function. Again the configuration is analysed on fulfilling the design constraints. If not, then floorplan driven decomposition steps are applied in which both PLA's of the configuration are independently decomposed into smaller PLA's and/or ROM's.

4.1. Root implementation

The least complex implementation contains one PLA with a state feedback register (figure 4). The alternatives at this level are a Mealy or Moore type FSM implementation. For several reasons the Mealy implementation is selected. In the first place, it is due to the large 'join'/'fork' constructs found in direct sequence in the control flow graphs typical for instruction set processors (figure 2). Therefore, the number of Mealy transitions, and thus the number of PLA product-terms, is much smaller than the number of Moore transitions. Secondly, the conceptual input-to-output delay is twice as large for a Moore single PLA as for the Mealy single PLA. By this, the Mealy implementation is favoured.

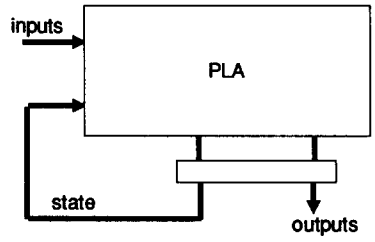

Figure 4. The root implementation.

This implementation is then checked with the floorplan, in aspect ratio as well as total size. Also the estimated PLA delay is compared with the desired delay. In both comparisons an estimated minimization impact of state-assignment optimization is taken into account. If one of these checks fail, the synthesizer decides to decompose. Otherwise state-assignment [3] is performed followed by logic minimization which completes the design.

4.2. Function separation

In case the single PLA implementation does not fit the design constraints, the first analysis phase decides to perform the first decomposition step. At this level we can apply several types of decomposition. The selection is based on estimations of the alternatives. Two alternatives are possible, in the first place a functional decomposition in which the two FSM functions are separated and implemented in separate PLA's as found in for example [2,8] (figure 5). The second alternative is a decomposition in parallel [9] or cascade connected FSM's (figure 6). The parallel decomposition or segmentation is based on separating the output variables into groups in such a way that each group has fewer product-term as the original PLA. However this will only provide optimization in case several independent groups of output variables (for example caused by different datapaths) exist. The decomposition into cascade FSM's is shown to be difficult [10] and furthermore we view this stage in the synthesis not as the appropriate one to perform this type of decomposition. Instead the FSM hierarchy should already be apparent in the behavioural description, which is the correct place for it. This because information is lost when a hierarchy is flattened and thus it is difficult and from a methodical point of view incorrect to try to recognize and build the FSM hierarchy up out of a flat description.

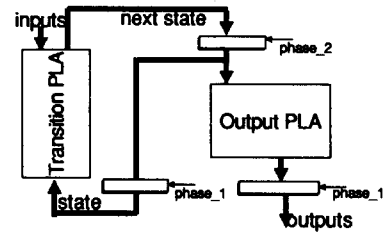

Figure 5. Functional decomposed Moore machine.

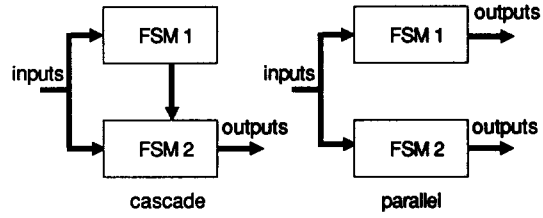

Figure 6. Parallel and cascade decomposition.

The functional decomposition on the other hand, provides a separation into PLA's which have a different structure. For example the output function has a much larger number of output variables than the transition function, and the reverse is true for the number of input variables. Therefore, by implementing both FSM functions in separate PLA's, each individual PLA can be minimized with its own strategy. This provides a larger optimization

potential than the previous decomposition alternatives at this decision level. However, functional decomposition applied without additional optimization does not always minimize the configuration. That is caused by the area of the input decoder plane of the output PLA. However, due to optimization methods discussed in following sections, the functional decomposition will nearly always be smaller as the single PLA configuration.

The selection of segmentation or functional decomposition is now based on the aspect ratio of the floorplan compared to that of the single PLA. If it differs, and the floorplan area is larger than the single PLA, segmentation [9] is performed. Otherwise the functional decomposition is performed. For instruction set processor control, segmentation often occurs in case of RISC machines. For those machines a single PLA implementation is in general acceptable, from a size point of view.

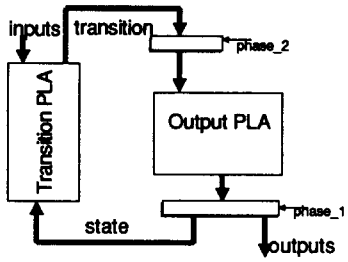

Figure 7. Functional decomposed Mealy machine.

Figure 7. shows the Mealy functional decomposition. The left PLA takes the inputs and the Mealy state codes and generates (Moore) transition codes which are input for the right PLA which generates the appropriate output and the Mealy feedback state code. Note that the transition codes are equal to Moore state codes. Another functional decomposition of the Mealy FSM is possible, which is equal to the Moore decomposition of figure 5, extended with the inputs also supplied to the output PLA. This alternative is much larger due to the presence of an extra input decoder in the output PLA. For that reason it is not considered. The next decision to be made is again a Moore or Mealy decision. Did the input to output delay differ for the single PLA, the functional decomposed machines have equal input-to-output delays in terms of clockphases (figures 5,7). But the Mealy machine has the smaller number of transitions which result in the smaller transition PLA. Therefore the Mealy machine prevails again.

The decomposed configuration is again analysed on its suitability to implement the control function. If the design constraints, e.g. floorplan and delay, are not fulfilled then floorplan driven decomposition steps are applied on both individual PLA's of this configuration.

4.3. Floorplan driven decomposition steps

This section discusses the decomposition steps applied on a functional decomposed FSM. In this stage it becomes necessary, not only to check the configuration against the floorplan, but also to let the floorplan drive the decomposition. Floorplanning, formerly mostly treated as a part of placement and routing, is becoming more and more incorporated in synthesis. Floorplanning within synthesis reduces the placement and routing problem significantly. The transition and output function PLA's are decomposed separately with different algorithms because the functions they implement have different parameters.

The decomposition is based on a few standard floorplans, which are designed such that they contain minimal wiring area. The basic floorplan area is divided into two sections as shown in figure 8. The output function PLA/ROM is always positioned vertically, so that the often very wide control line bus is kept as short as possible. This contributes to a minimal area consumption for wiring.

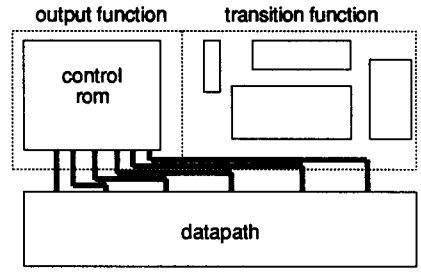

Figure 8. Global floorplan.

The last question is whether to decompose the output function or the transition function first. This is decided by checking which of both PLA's is the largest. The function belonging to that PLA is decomposed first. This is mostly the output function. After this decomposition it is checked if a decomposition of the other FSM function is still needed.

4.4. Output function decomposition

A number of design alternatives are possible for the output function implementation. Among them are ROM or PLA structures to implement the output function in. Also a decision must be taken on encoding alternatives for the output variables (datapath control variables). The decision whether to apply a ROM or PLA structure depends on the parameters of the output function of the FSM. The most important difference between a ROM and a PLA is the complete decoder found in the ROM. This implies that the state code must be regular in case of a ROM. That reduces the freedom for state-assignment methods. Due to this regularity column multiplexers can be applied in a ROM structure, while the number of columns in a PLA is bound to one.

Therefore the decision to apply a PLA or ROM structure is strictly based on the floorplan dimensions. In case the number of Mealy transitions (is Moore states) causes the

output PLA not to fit in the floorplan in the defined vertical way, the ROM structure is selected, for more than one column is needed to fit the structure into the floorplan. Otherwise a PLA structure is selected to keep a larger freedom for state-assignment algorithms. On the other hand also segmentation [9] of the output function may result in a PLA configuration which fits the floorplan. But output function segmentation is not considered to be more effective than segmentation of the single PLA (section 4.2) and therefore not incorporated at this design stage.

4.4.1. Output encoding

The encoding problem is to find an optimum encoding level between a horizontal and a completely vertical encoding. A systematic way to find this optimum, based on the increase of delays and decrease of size is incorporated [11]. Currently this method is open towards state-assignments. Therefore only ROM column counts are allowed which are a power of 2. Other numbers of ROM columns would cause missing state codes in the address range and thus restrict the state encoding. Thus, the aspect ratio of the ROM is variable with a stepsize of 4. For most technologies optimal aspect-ratio's for size and delay are found for aspect ratio's near one. Here the optimal aspect ratio of the ROM is defined as the one which fits best to the floorplan area reserved for the output function. This fit is defined in terms of the absolute measures (1).

(1) $\text{Floorplan-fit} = |Y_{rom} - Y_{flp}| + |X_{rom} - X_{flp}|$

Another aspect concerns the different floorplan variants for the ROM/PLA with output decoders. This because not every feasible floorplan leads to a near minimal wiring area. The floorplans incorporated are shown in figure 9.

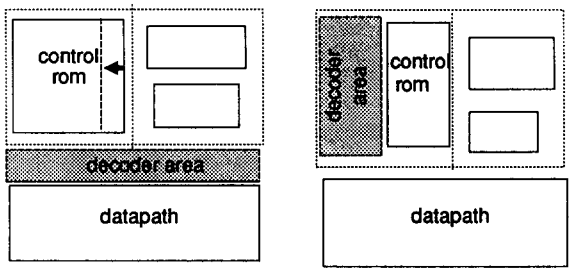

Figure 9. Floorplans for large and small decoders.

These two floorplans are in principle sufficient to allow a complete encoding range from horizontal to vertical. For low encoding levels small decoders are generated, which can be positioned in the area left free between the ROM and the datapath. Higher levels of encoding prohibit this positioning, because the decoders are large. For vertical encoding, the size of the decoder (nano-ROM) is in the order of magnitude as the micro-ROM itself. Therefore the second configuration in figure 9. is introduced. In this configuration the full height of the controlpath floorplan is available for the decoder(s). Different complexities of decoders are possible within both configurations. Due to the regularity of the codes used for encoding, the output decoding can also be performed by small ROM's with multiple columns instead of PLA's. That makes it possible to fit decoder ROM's of different word counts within one Y-dimension value.

The algorithm [11] generates an encoding level for each of the decoder size levels. The task of the algorithm is to generate encoding levels, which fit to the floorplan with a minimum of area loss and a maximum of size minimization. The encoding levels are created by a heuristic search for optimal combinations or clusters of control field decoders. The first encoding level above the horizontal encoding is the control field level. First the design parameters are computed for each control-field decoder. Then all other encoding levels are generated. A cluster is accepted by the algorithm if the size gain of the cluster decoder is larger than the sum of the size gains of the individual field decoders contained in the cluster. The optimal encoding level is then determined by the delay constraint.

4.5. Transition function decomposition

In case the transition PLA, after minimization, does not fit in the area reserved for it in the basic floorplan, it is decomposed. This section describes the decomposition of the transition function PLA. But first a minimization is discussed based on countable transitions.

4.5.1. Countable transitions

The application of a counter in functional decomposed FSM is studied extensively in [2]. However, Amann's study is based largely on Moore type FSM's. For Mealy type FSM's he applied a local Mealy/Moore transformation. Thus, each Mealy state is expanded with local bits to address the output productterms in the output PLA from the different transitions. This transformation, however, is inefficient with respect to the number of countable transitions. That is due to the state locality of the Mealy/Moore transformation performed, which keeps the number of countable codes limited to countable Mealy state codes. Due to the complete Mealy/Moore transformation performed in our system, we are not limited to countable Mealy state codes, but have countable Mealy transition codes which are equivalent to Moore state codes. Thus, the number of countable transitions is as high as the number of countable state codes in the equivalent Moore machine, while the number of transitions is significantly smaller.

The minimization occurs because the countable transitions are removed from the transition PLA at the cost of one extra state-code to activate the counter. This code is generated by the PLA as default, in case no productterms are matched. The detection of the optimal collection of countable transitions is performed differently for both types of behavioural input. With the gateprogram type of

input selected, no special algorithm is needed, because the gateprogram is imperative and thus, when concise programmed without redundant branches, it specifies countable chains of Moore states. This makes detection of countable transition chains trivial. For the PLA type of input however, the numbering of states is undefined and therefore must be assumed 'random-like' and thus not optimal. State-chain detection algorithms as developed in [8] are needed in this case. Currently a simple state-chain detection algorithm is implemented for this type of input.

4.5.2. State distribution

The need for decomposition of the transition function is also derived from the floorplan. First, a check is performed whether the transition PLA, already optimized for countable transitions, fits into the area reserved for the transition function in the floorplan. If it does, no decomposition is performed, and the final synthesis steps as state-assignment followed by logic minimization conclude the synthesis process. Otherwise, the transition PLA is split up into smaller PLA's, which are multiplexed to the next-address register (figure 10). Therefore the Mealy state code feedback from the output ROM/PLA is recoded by splitting it up in two separate codes as is shown in figure 10. The first feedback controls the transition PLA multiplexer and the second feedback is used for transition selection within the PLA's. Together both codes represent the Mealy state code. The advantage of this is twofold, the delay of the transition function decreases and a better fit to the floorplan is derived.

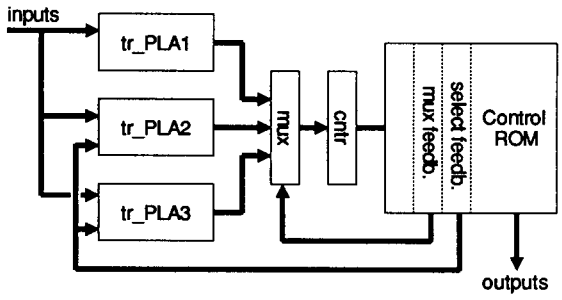

Figure 10. Transition PLA decomposition.

By widening the multiplexer more smaller transition PLA's occur, and in this way the transition PLA configuration is adapted to the floorplan area aspect-ratio. A constraint is that all transitions starting from a state must be kept together in one of the PLA's. This limits the widening of the multiplexer. In practice, however, this limit has no impact for the floorplan.

Furthermore, this decomposition saves space in case that not all input variables are relevant to the transitions captured in a transition PLA. For instruction set processors this is the case if, for example, the address modes and the opcodes are coded in separate fields of the instruction format. Then for decoding the operation from the opcode (in case no address dependency exists) no other input variables are relevant. Thus, when the start transitions of the operations are collected in one PLA, without other transitions (a transition PLA with one Mealy state), other input variables can be removed from this transition PLA. This reduces the total AND-plane area.

The actual decomposition is performed by a state distribution algorithm. This algorithm is incorporated to keep the numbers of productterms in the diverse transition PLA's about equal. This, because the numbers of transitions can differ largely between states. For example the state in which the operation selection or address mode selection is performed will be quite large (figure 2). It is evident that when a transition PLA contains transitions from just one state, it does not need feedback information to select the correct transition. The selection function is then performed completely by the multiplexer steered by the output PLA/ROM. Such transition PLA's without feedback inputs are called decoders.

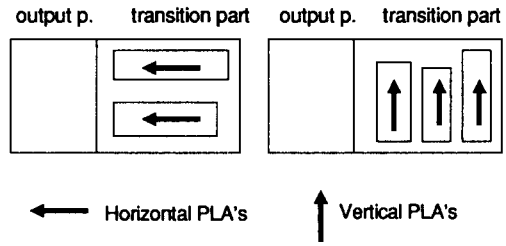

Figure 11. Transition function floorplans.

This algorithm performs the state distribution for each of both floorplans shown in figure 11, these differ in the direction of the PLA's. The distribution which has the best fit to its floorplan area is selected. First the algorithm estimates the number of transition PLA's based on the best fit in both dimensions. This estimation is based on the average number of productterms that the transition PLA's will contain. Hereafter the algorithm distributes the states over the transition PLA's in which it tries to keep the lengths of the PLA's about equal while it minimizes the AND-planes, by optimizing to the total area involved. The algorithm applies a greedy method which sorts the states according decreasing numbers of productterms. States which have a number of productterms nearly equal to that of the computed average number of productterms in one transition PLA are put aside. These states are implemented in separate transition PLA's. As mentioned 'single state' PLA's do not require feedback input which also saves area. Furthermore they are also open for pipelining, which can be realized by addition of a register between the PLA and the multiplexer. After the state distribution is made for both floorplans, the configuration which fits its floorplan the best is selected and actually generated.

4.6. Preview

A preview utility is incorporated to trace and estimate possible floorplan aspect ratio's with the amounts of empty space contained in them. The goal of this preview is

to supply the user with information of what he can expect from the first two decomposition phases in terms of aspect ratio's and sizes. This is useful because these first decomposition phases have the largest impact on the design. The user can detect if these steps are sufficient to fulfil the design constraints. Then he can choose the solution closest to the design constraints and put the decomposition to work. The decomposition result may, of corse, differ from the preview. The preview utility takes a range of feasible ROM column numbers into account and computes also a feasible range of transition PLA numbers. The transition PLA range is limited by the state with the largest number of transitions leaving it. The area savings due to input saving on transition PLA's are not estimated. Also the impact of output encoding levels is not included. Currently the preview utility is being expanded with the estimation of delay's involved in both clock phases. The decision whether the floorplan aspect-ratio or the delay is most important for the design is left to the user.

5. Results

In this section the results are given for some example machines implemented with this method. Table 1 shows the parameters of the three example machines. Machine 1 is a RISC-like instruction set processor with 5 address modes and 14 operations. Table 1 shows that the difference between the Mealy and Moore transition count is large, due to the 'fork' constructs. Machines 2 and 3 are more general FSM's. At this moment the delay parameter is implemented in the output encoding only and therefore left out of this comparison. The optimization is performed towards the floorplan and size. First a preview was performed. The preview data are found in table 2. These data are based on a given floorplan aspect ratio range between 2.0 and 0.2, with a maximum empty space of 10% for machines 2 and 3 and 20% for machine 1. Based on this data the designer took a choice for the floorplan. The aspect ratio in the second column of each machine in table 2 was chosen as target for the synthesizer. The parameters of the single PLA configuration are included in table 2 for comparison.

	Moore trs.	Mealy trs.	Moore st.	Mealy st.	ins	outs
exam1	281	91	67	50	12	25
exam2	226	115	96	48	7	19
exam3	185	166	139	121	27	56

Table 1. Parameters of example machines.

	exam1			exam2			exam3			
	1	2	s	1	2	s	1	2	3	s
X	115	66	65	128	76	51	155	292	144	131
Y	41	82	91	55	99	115	139	72	152	161
as,r	.36	1.2	1.4	.43	1.3	2.3	.90	.25	1.1	1.2
emp	183	880	-	161	645	-	684	163	1027	-
col	2	1	-	2	1	-	1	2	1	-
tr	1	2	-	2	3	-	1	2	2	-
dir	V	H	-	V	H	-	V	V	H	-

X = x-dimension of floorplan in PLA cells
Y = y-dimension of floorplan in PLA cells
as.r = aspect ratio of floorplan
emp = empty space in floorplan
col = columns of ROM
tr = number of transition PLA's
dir = direction of transition PLA's H(orizontal) or V(ertical)

Table 2. Preview results and single PLA data.

	exam1	exam2	exam3
X	70	79	264
Y	70	96	72
as.r	1.0	1.2	.27
emp	316	161	784
col	1	1	2
tr	2	3	2
dir	H	H	V
ins	8/8	6/7/6	18/16

ins = number of input variables for each transition PLA

Table 3. Synthesis results.

Table 3 shows the actual implementation data, all machines were functionally decomposed and a counter was included. These results are without logic minimization. Also the number of inputs is shown for each transition PLA. For all three machines the synthesis generated the number of transition PLA's as estimated by the preview utility. In all cases the synthesizer did manage to reduce the amount of empty space within the floorplan. This is caused mainly by the reduction of the inputs for each of the separate transition PLA's. Especially for machine 3, the result of this input separation over the transition PLA's is large, from the total 27 inputs just 16 inputs for PLA1 and 18 for PLA2 are needed. Also for machine 1 only 8 out of the 11 inputs are needed for each separate PLA.

6. Comparison to other systems

Many controlpath synthesis methods are found in literature, but they are all restricted to the size minimization problem. The first methods are the microprogram development methods [12,13,14]. Those methods are based on a fixed the microarchitecture and focus completely on the problem of generating an optimal microprogram, which problem is quite different from the problem of generating the optimal microarchitecture. Comparison of different microarchitectures is not incorporated in those methods. The comparison of microarchitectures was done in [4] and proven to be useful, but no synthesis method was presented.

A number of systems are based on PLA synthesis, with minimization techniques varying from state-assignment, PLA folding and PLA segmentation [2,3,8,9]. Floorplan or delay considerations are again not included. In [8] a system is described in which the state-assignment of [2] is combined with a method for minimizing the number of states by identifying state-sequences of which the differences in outputs can be decoded straight from the instruction code. In this system, however, the flexibility of the PLA configuration is determined completely by this size minimization and not on floorplan considerations.

The current synthesis methods do perform useful optimizations, but are not suited to generate the wide range of microarchitectures of todays microprocessors. Its ability to generate a wide range of microarchitectures together with the integration of floorplanning into the synthesis make stepwise decomposition suited to controlpath design. But stepwise decomposition inherits current optimization algorithms at those stages in the design were they do contribute to the design quality.

7. Conclusions

The stepwise decomposition method is more flexible towards floorplan and delay constraints than current controlpath synthesis methods. It is able to generate a range of microarchitectures determined by floorplan and delay constraints, beside the size minimization. The user is able to preview implementation estimations for the first decomposition steps. This supplies information on the design space available for the controlpath under synthesis. The method is currently able to synthesize PLA or ROM based output functions, combined with an optimal encoding in the range of horizontal to vertical encoding [11]. Also the transition function is decomposed to fit the floorplan and reduce delay. At this moment not any desired floorplan aspect ratio can be generated. Also the forward estimation of minimization results, of minimizations performed after decomposition of the components, needs more attention. Therefore research is planned on enhancement of the decomposition steps. Also methods for state-minimization by static instruction decoding as described in [4,8] are being developed. These developments will enhance the flexibility of the synthesis and enlarge the design space for the processed controlpaths.

References

[1] Brayton, R.K., Algorithms for multi-level logic synthesis and optimization, in: DeMicheli, G. and Sangiovanni-Vincentelli, A. (Eds.), Design Systems for VLSI circuits (Nyhoff, 1987) pp. 197-248.

[2] Amann, R., Algorithmische entwurfsverfahren fuer kombinierte pla/rom-steuerwerke unter verwendung van zaehlern, Dissertation, (VDI Verlag, Duesseldorf, 1987).

[3] DeMicheli, G., Optimal State Assignment for Finite State Machines, IEEE Trans. Comp. Aided Design, vol. CAD-4, (1985) pp. 269-284.

[4] Obrebska, M., Efficiency and Performance Comparison of Different Design Methodologies for Control Parts of Microprocessors, Microprocessing and Microprogramming 10, (1982) pp.163-178.

[5] Leveugle, R. and Soueidan, M., Design of an Appplication specific Microprocessor, in: Saucier, G. and McLellan, P.M. (Eds.), Logic and Architecture Synthesis for Silicon Compilers (North-Holland, Amsterdam, 1989), pp.255-268.

[6] Stritter, S. and Tredennick, N., Microprogrammed Implementation of a Single Chip Microprocessor, Proceedings of the 11th Annual Microprogramming Workshop., (1978), pp. 8-16.

[7] Blaauw, G.A., Digital System Implementation, (Prentice-Hall, 1976).

[8] Mahler, H. et. al., Processor Control Part Synthesis Using Effective Partitioning Algorithms, Microprocessing and Microprogramming 23 (1988) pp. 33-36.

[9] Grass, W., A Synthesis System for PLA-Based Programmable Hardware, Microprocessing and Microprogramming 12 (1983) pp. 15-31.

[10] Hartmanis, J. and Stearns, R.E., Algebraic Structure Theory of Seqential Machines, (Prentice Hall, 1966).

[11] Berg, A.J.W.M. ten, Floorplan driven Controlpath Synthesis, in: Veen, J.P. (ed.), Proceedings of the second Symposium on Design Methodology, (ProRisc (STW), Dalfsen, april 1990), pp. 67-70.

[12] Gessner, J. et. al., Synthesis of Control Units in a Design Environment for Chip Architecture, Microprocessing and Microprogramming 27 (1989) pp. 465-472.

[13] Sheraga, R.J. and Gieser, J.L., Experiments in Automatic Microcode Generation, IEEE Trans. on Computers (vol c-32, no.6, 1983) pp. 557-569.

[14] Davidson, S. et. al., Some Experiments in Local Microcode Compaction for Horizontal Machines, IEEE Trans. on Computers (vol. c-30 no.7 1981) pp. 460-477.