High level synthesis: a data path partitioning method dedicated to speed enhancement

Fabrice Monteiro, Bruno Rouzeyre, Georges Sagnes

HIGH LEVEL SYNTHESIS: A DATA PATH PARTITIONING METHOD DEDICATED TO SPEED ENHANCEMENT

F.MONTEIRO B.ROUZEYRE G.SAGNES

Laboratoire d'Automatique et de Microélectronique de Montpellier
UA CNRS D03710, Université Montpellier II
Place E. Bataillon, 34095 Montpellier Cedex 5, FRANCE

Abstract

In the field of high level synthesis, a speed improvement of structural designs can be obtained by partitioning the physical data path of the behavioral compilers outcome. This speed improvement is achieved by increasing the number of operations treated simultaneously without appreciable overhead in the silicon area.

In this paper, we present a partitioning method based on bus splitting. This method makes use of hierarchical clustering and a description of all the measures needed for partitioning is given.

I-Introduction

High level synthesis consists in generating an RTL structural description of a circuit from a behavioral specification, usually given as an algorithm. The design research space has two dimensions:

– the first one is the time needed for completion of the algorithm on the generated design. Time is usually expressed as the number of control steps.

– the second one is the silicon area used by the design.

Time and area evolve in opposite directions. In fact, the number of operations that can be simultaneously performed directly depends on the number of available hardware resources (function modules, memories, interconnections, etc...). Usually, in order to explore the design space, a one dimensional constraint is set up and then a design minimizing the other dimension is sought.

Synthesis consists of three main tasks:

– the scheduling of the behavioral description, i.e. the assignation of operations to control steps.

– the allocation of hardware components. This consists in mapping the operations (e.g. additions) onto function modules (e.g. adders, ALUs), and the variables onto memory modules. The allocation aim is to obtain a minimal number of design resources and consequently to minimize area cost.

– the generation of interconnections (buses, Mux/Dmux, etc...) in order to link the previously defined structural entities.

The constraints taken into account during the scheduling task are either time or hardware constraints. In the first case (e.g.[1],[2]), the behavioral description is split up into steps so that the total running time (or the latency for pipeline structures) is below a given boundary and the amount of hardware required (or an estimation of that amount) is minimized. In the other case, the type and the amount of hardware are limited and the scheduling splits the algorithm up so as to minimize running time (e.g.[2],[3],[4],[5],[6]). Allocation and scheduling are interrelated tasks: the constraints related to the sharing of hardware are linked to the distribution of operations between the control steps. For instance, an operator can not be used more than once in a step and a register can not be shared by variables with overlapping life times.

Ideally, scheduling and allocations should be simultaneously processed. In order to find global optimal solutions, some systems adopt this approach (e.g.[3],[7],[8],[9],[10]). However, as problem complexity is important, the two tasks are often performed in turn: scheduling before allocation (e.g.[4],[11]) or conversely (e.g.[12]). In order to obtain good solutions, some estimates of the second task objectives are used during the first one. This method can be repeatedly applied until a satisfactory solution is reached.

The last task in the synthesis of the data path is the generation of a minimal set of connections binding the hardware entities. Two models of connectivity are used:

– a point to point model using Mux/Dmux devices and wired broadcast nets.

– a bus based model. In this case, the resources are only connected to and from buses. Buses are the generalization of Mux/Dmux (see [13] for a complete presentation). As place and route information is not available at this point in the synthesis, the buses are assumed to run all over the design. Thus, only the number of buses and of connections to and from buses are subject to minimization.

In both design styles, estimates of connectivity cost can be used to drive the allocation (e.g.[8]) and/or the scheduling tasks (e.g.[5],[6]). For instance, in a bus based connectivity model, parallelism and time (i.e. scheduling) are linked to the number B of buses by:

$$(1)\quad B(s) \geq Tr(s),\ \forall\, s \in S \quad \text{and} \quad B = \text{Max}\ \{B(s)\}$$

where Tr(s) denotes the number of data exchanges between different resources during step s and S is the set of steps issued from the scheduling. This implies that any boundary on the number of buses entails an increase in the number of steps, i.e. a loss of speed of the circuit.

Most of the synthesis tools do not explore the possibility of automatically partitioning the data path and so tend to produce designs which are as compact as possible. In particular, they are unable to find the natural and functional splitting of the data path. The data and address computation parts of a microprocessor are a classic example of this. Some tools [14], BUD[15], APPARTY[16], partition the early behavioral description in order to reduce synthesis complexity. The partitioning is performed before allocation and influences the overall design.

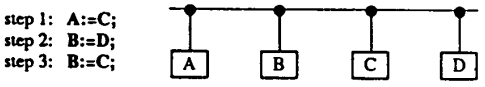
In this paper we propose a method for partitionning a pre-synthesized design by splitting the buses in order to improve parallelism without enlarging the area. This research is initiated by the method used in the APPOLON[17] data path synthesizer. Parallelism was made possible in the 2-bus fixed target architecture by means of such splittings.

II-Method principle

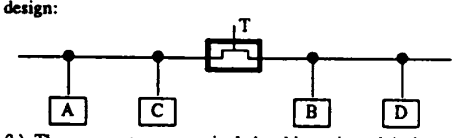
II.1-Principle

The basic idea is illustrated in Fig.1. A transmission gate allows the temporary isolation of two parts of a bus. Thus, one actual bus can be split into two virtual buses. When the transmission gate is OFF, two data transfers can be simultaneously performed, one on each side of the split, instead of only one when the gate is ON. So, by means of the splits and of a rearrangement of the resources on both sides of the bus (e.g. A,C and B,D on Fig.1b), it is possible to reduce the number of steps without any change in the number of actual buses.

The method proposed below is the generalization of this idea for a design with B buses. It consists in reorganizing the connectivity binding of a circuit so as to speed it up. The N resources are to be apportioned to P sub-data-paths (clusters), each cluster having no more than B buses. More accurately, the resources must be topologically organized so that bus splitting positions can be determined. These splits allow an increase in parallelism i.e. more

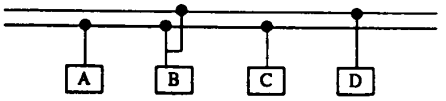


(a)-Extract from an algorithm on a single bus non-partitioned design:



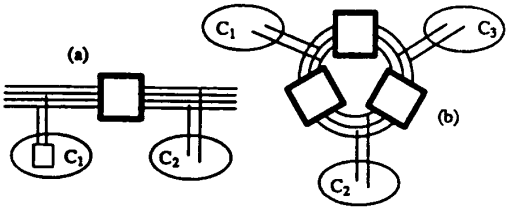
(b)-The same extract on a single bus bi-partitoned design:

step 1: A:=C || B:=D; (gate OFF)
step 2: B:=C; (gate ON)



(c)-For the same scheduling as in (b) on a non-partitionned design, two buses are necessary. Notice that the number of connections is increased

Fig.1: Example of partitioning



The figures (a) and (b) illustrate the organization for two and three clusters. For more clusters, the problem of their ordering around the bus ring arises: communications between non adjacent clusters reduces the amount of available buses for intermediate clusters. Only the bi- and tri-partitioned designs are considered.

Fig.2: Architectures of partitioned data-path

operations being executed simultaneously without an increase in area. From a structural point of view, the P clusters are linked by Mux/Dmux devices. These are composed of transmission gates binding the buses of the different clusters together to form B global buses (cf.Fig.2). At a given time, these devices allow any pair of local buses to be connected while keeping the other local buses isolated. This method is accurate for a bi- or a tri-partition (cf.Fig.2).

SPEED ENHANCEMENT: At a given time, if local buses are joined by closed transmission gates, they form what we call a virtual bus. If Bv(s) is the number of virtual buses at step s, equation (1) still holds for Bv(s). Since parallelism and connectic are linked by relation (1) and $B \leq Bv(s) \leq P \times B$, the shortening of the schedule is much more important if Bv(s) can be kept close to P×B, and this for as many steps s as possible. In order to handle this, the clusters have to be isolated for as long as possible. This means that resources must be placed in such a manner that as few data exchanges as possible have to pass through the splits.

CONNECTIC: Besides this time aspect, the partitioning of a design doesn't enlarge the connectic cost and most of the time it reduces it.

- Each resource's port has at most the same number of connections to the buses as in a monolythic design with B buses. In a partitioned design, resources may have less connections since the Mux/Dmux devices allow other communications than wired ones.
- In architectures with more than B buses, the resources' ports may at worst be connected to each bus (and from our experience the number of connections is about the same as with B buses). So in this case, not only is the number of actual buses higher but the number of connections is also potentially higher (cf. B in Fig.1c).

In any case, the only loss is the Mux/Dmux device. Furthermore, the topological arrangement of the ressources into clusters is valuable information for future place/route tasks and a priori better positionning will result.

II.2-Position in the design flowchart

Let T_1 be the number of steps resulting from the initial scheduling on a non-partitioned design using B buses and a set $E=\{R_1,..,R_N\}$ of hardware resources. Let T_0 be the number of steps obtained when the bus constraint is relaxed. The number B_{max} of buses needed in this case is extracted from equation (1).

A partition in P (P=2 or 3) operative "sub-data-paths" of the design (E,B) is sought so as the number of steps T_P is less than T_1. Fig.3 positions this partitioning method in the synthesis flowchart. It is unhelpful to apply the method when the constraint B on the number of buses is not lower than B_{max} because $T_0 \leq T_P \leq T_1$ always holds and $T_1=T_0$ every time $B \geq B_{max}$.

As any scheduling tool which allows connectivity constraints to be taken into account can be used to compute T_0, T_1 and T_P, this paper only focuses on the partitioning method.

III-Method presentation

First, we deal with circuits for which the behavioral specification (built in algorithm) is composed of only one sequence of statements. The most wellknown examples of such circuits are digital signal processing circuits (DSP). The generalization for ordinary algorithms (with loop constructs, branching, conditions, etc...) will be presented in paragraph VI. So, we consider a sequential specification mapped onto a set E of N resources and scheduled in T_0 steps by relaxing the bus constraint.

The problem consists in apportioning the N resources to P sets (clusters) in order to speed up the algorithm. Thus, it is a

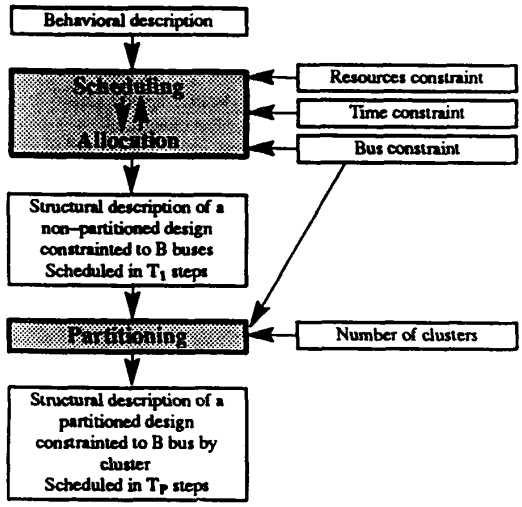


Fig.3: Position of partitioning in data path synthesis.

classification problem. As mentionned earlier, the partitioning will be all the better as less data transfers between clusters will occur.

If we define for each pair of resources an affinity measure of the interest of grouping them, the grouping into clusters will be all the better as the measure will be important for resources in a given cluster (and low in different clusters).

If the N resources are represented by points in an N-dimensional space and if the distance between the points denotes the advantage of grouping them, then, this problem is equivalent to the partitioning of the overall set of points into P clusters so that the dispersion inside clusters is minimal.

Finding such a partitioning is an NP-complete combinatory problem. A suboptimal solution is obtained by using a hierarchical clustering. It consists in repeatedly joining the clusters that involve the lowest increase of intracluster inertia or equivalently, the lowest loss of intercluster inertia (Ward criteria) [18]. It is well known that, for a cluster of points, the dispersion of the points matches with the internal inertia of the cluster (e.g. [19]).

IV-Method description

IV.1-Representative points definition

(1)-A proximity measure $m(R_i,R_j)$ of a purely local nature is first established for each pair of resources. The greater the number of data exchanges occurring between these resources, the more important is the measure. The measure is defined as follows:

- let $E=\{R_1,...,R_N\}$ be the set of the resources.
- let $S=\{S_1,...,S_{T0}\}$ be the set of the steps that make up the sequence. The data flow of the step S_k is known as a subset $DFG(S_k)$ of the data flow graph DFG(S) of the sequence S.
- let $A(S_k,R_i,R_j)$ be defined as (cf.Fig.4b):
  - $A(S_k,R_i,R_j)=B-1$ (recall that B is the maximum number of buses allowed)
  - $A(S_k,R_i,R_j)=A(S_k,R_j,R_i)=B-L(S_k,R_i,R_j)$ when $i\neq j$ and a directed path between R_i and R_j (or R_j and R_i) in $DFG(S_k)$ shorter than B exists, $L(S_k,R_i,R_j)$ being the length of the shortest one (for instance in Fig.4b, $A(S_4,I,\delta)=5-1=4$).
  - $A(S_k,R_i,R_j)=0$ otherwise.

$A(S_k,R_i,R_j)$ is an overvaluation of *the number of buses that remain available in* S_k to achieve the data transfers other than those between R_i and R_j. This number is strongly related to the potential shortening of the final sequence scheduling (as the number of data transferts that can be moved into S_k.

- Finally, $m(R_i,R_j)=\sum_{k=1}^{T_0} A(S_k,R_i,R_j)$

For instance, in the example of Fig.4b, $m(\delta,H)=4+4+4+0$. Notice that the influence of the other resources is ignored in $m(R_i,R_j)$.

(2)-The resources are placed in the N-dimensional space where the j^{th} coordinate of the resource R_i is the value $m(R_i,R_j)$. Placing the representative points in the N-dimentionnal space looks like the generation of a correlation matrix. It crosses the local information between them and supplies information of a global nature.

The affinity measure is taken as the euclidian distance $d(R_i,R_j)$ between R_i and R_j in this space. It is a global measure between these two points as it takes into account not only the physical link of the pair but also their respective environments, i.e. the other resources they are linked to. In this way, it corrects the drawback of $m(R_i,R_j)$.

IV.2-Measure between sets of resources and partitioning

From the definition of the measure between points, it is necessary to derive a distance for sets. The pertinence of regrouping two clusters is relevant to the intercluster inertia reduction when joining the two clusters (or equivalently, to the intracluster inertia increase, the total inertia being the sum of the inter and intracluster inertias). We will call the loss of intercluster inertia when merging two clusters "agglutination cost".

The partitioning is done by ascendent classification: at the beginning, each resource is considered as a cluster. At each step of the clustering, the two clusters with the minimal agglutination cost are joined together. The agglutination costs between this new cluster and the others are updated. The process is repeated until only one cluster remains. P clusters are obtained by cutting the hierarchy tree at a cost level so that there are P subtrees under the cut (cf.Fig.4d).

Intercluster inertia is the sum of the inertias of the gravity centers G_i of every cluster C_i related to the total gravity center G. When Q cluster are present, if $W(C_i)$ is the weight (mass) of cluster C_i, interclass inertia I_{Inter} is:

$$I_{Inter}=\sum_{i=1}^{N} W(C_i).d^2(G_i,G) \qquad I_{Total}=\sum_{i=1}^{N} W(R_i).d^2(R_i,G)$$

The total inertia of the set of points I_{Total} is equal to the sum of inter and intracluster inertias (cf. Huyghens inertia equation).

When two clusters a and b merge into one, the contribution $W_a.d^2(G_a,G)+W_b.d^2(G_b,G)$ of these clusters a and b to the intercluster inertia is replaced by $(W_a+W_b).d^2(G_{ab},G)$ where G_{ab} is the gravity center of the new cluster and (W_a+W_b) its weight.

Since: $G_{ab}=\dfrac{W_a.G_a+W_b.G_b}{W_a+W_b}$ and

$$(W_a+W_b).d^2(G_{ab})=W_a.d^2(G_a,G)+W_b.d^2(G_b)-\frac{W_aW_b}{W_a+W_b}.d^2(G_{ab},G)$$

the inertia loss $\delta I(a,b)$ is:

$$\delta I(a,b)=W_a.d^2(G_a,G)+W_b.d^2(G_b,G)-(W_a+W_b).d^2(G_{ab},G) \quad (2)$$
$$=W_aW_b/(W_a+W_b).d^2(G_{ab},G)$$

In the beginning, the first task is to build the matrix Δ of the "inertial distances δI" of every pair of resources ($\{R_i\},\{R_j\}$) using the relation (2), since each resource defines a cluster. The above–mentioned repetitive process is then applied.

When two clusters a and b merged into the cluster aUb, the matrix of the inertial distances is updated using the following formula to compute the $\delta I(aUb,c)$ of the new cluster aUb for all the other clusters c ($c\neq a$ and $c\neq b$):

$$\delta I(aUb,c)=\frac{(W_a+W_c).\delta I(a,c)+(W_b+W_c).\delta I(b,c)-W_c.\delta I(a,b)}{W_a+W_b+W_c}$$

The final result can be presented as a binary tree (cf.Fig.4d) where the internal nodes are represented at levels proportional to the loss of inertia for the corresponding mergers.

IV.3–Remarks

(1)–In order not ro break common sub–expressions structures into several clusters, an extra term ζ is added to A which reinforces the link between such resources.

- $\zeta(S_k,R_i,R_j)=(B-\text{Min}\{L(S_k,R_i,R_u)+L(S_k,R_j,R_u)\})/2$ when R_u common descendant of R_i and R_j exists in DFG(S_k) and $\text{Min}\{...\}<B$ (e.g. in Fig.4b, $\zeta(S_4,F,I)=(5-(1+1))/2=1.5$).
- $\zeta(S_k,R_i,R_j)=0$ otherwise.

(2)–The same weight can be set to each resource. But it can be interesting to give different weights to the different resources. In fact, clusters having lower weights are inclined to merge before the others since for the same distance the variation of inertia will be lower. Due to the greedy aspect of the clustering algorithm, this fact leads us to assign lower weights to the resources that are to be merged first.

This is the case of situations in which some resources often occurs while some others rarely do. As it is interesting to deal with the first ones as soon as possible, low weights are assigned to them. One possible weighting, in inverse ratio to the number of steps in which resource R_i occurs is $W(R_i)=1/m(R_i,R_i)$.

The opposite situation occurs when the sequence involves a great number of resources each one being seldom used. In this case, it is better to group first the resources with few partners. As these resources, in this situation, are also the least used ones, the weighting factor is proportional to the number of steps in which the resources occurs. $W(R_i)$ is simply $m(R_i,R_i)$.

V–Example

Fig.4a is a didactic example of a data flow graph constrained by 5 buses. Fig.4b is the scheduling with the bus constraint relaxed. Fig.4c is the coordinates matrix from which the global distances and the inertial matrix are built. The hierarchical clustering tree is given in Fig.4d. The final structure and the scheduling for the two–partition design obtained are given in Fig.4e and Fig.4f respectively. On this example, the partitioning method leads to a speed enhancement of 3 steps which is the maximum for this example.

VI–Generalization

Except for digital signal processing circuits, behavioral descriptions are usually made up of several sequences (also called basic blocks) linked together by branch and test actions, synchronization points, etc... Thus the internal representation of behavioral specification appears as several DFGs.

Several problems appear when the partitioning is to be done on an RTL description carrying a multi–sequence algorithm:

a)–Partitioning must be applied to the information contained in all the DFGs (resources involved in the sequences, dependencies between these resources, etc...). It is not possible to merge the results of partitioning local to each sequence since two sequences which are independently partitioned can lead to conflicting results: e.g. partitioning the three resources R_1,R_2,R_3 can result in $\{R_1,R_2\}$, $\{R_3\}$ for a sequence S_i and $\{R_1\}$, $\{R_2,R_3\}$ for a sequence S_j.

b)–The sequences do not run the same number of times. It is the same for data exchanges in those sequences. The affinity coefficients must take into account these numbers of times.

The method that we propose consists in computing an "average partitioning" global to all sequences, controlled by the best local results. This method is the following:

(1)–For every individual sequence S_k:

- compute from DFG(S_k) the matrix Λ_{S_k} of the coordinates of all resources involved in sequence S_k ($M_{Sk}(R_i,R_j)=m(R_i,R_j)$).
- generate the partitioning P_k of sequence S_k with the above method.
- extract the gain $G(S_k,P_k)$ in running time.
- let $N(S_k)$ be the number of times the sequence S_k runs. Then, $G_T(S_k,P_k)=N(S_k)*G(S_k,P_k)$ is the contribution to the whole algoritm total gain if P_k is applied independently on S_k.

(2)–Compute the global matrix M of coordinates as:

$$M(R_i,R_j)=\left[1+\frac{\sum_{S_k\notin S(R_i)} G_T(S_k,P_k)}{Card(S(R_i))}\right].\left[\sum_{S_k\in S(R_i)}\frac{G_T(S_k,P_k).M_{S_k}(R_i,R_j)}{T_0(S_k)}\right]$$

where $S(R_i)$ denotes the set of sequences in which the resources occurs.

(3)–Apply the previous partitioning method to M the global matrix.

Remarks:

- The $N(S_k)$ coefficients can be obtained by simulation or at parse time.
- Generally the total gain will be less than $\Sigma N(S_k).G(S_k,P_k)$.

VII–Benchmarks

The algorithm has been applied to the differential equation presented in [4], using one adder, one subtractor, one comparator and one multiplier with a 3 bus constraint. The results are summarized in the following values:

- T_1=11 steps. This is the time used for the non–partitioned design.
- T_0=7 steps. This is the time spent when relaxing the bus constraint. With regard to allocations, it is the shortest time.
- T_2=8 steps. This is the time used on a bi–partitioned design.
- T_3=7 steps. On the tri–partitioned design, the maximum parallelism has been reached.

A second example is the filter presented in [5] synthesized using one multiplier and one adder. With this example, the results are: T_1=34 steps, T_0=26 steps, T_2=28 steps.

For these examples the partitioning obtained by our method is the optimal one as far as speed enhancement is concerned.

VIII–Conclusion

In this paper, a method of partitioning data path has been presented. It produces, under a given set of constraints (resources are already matched and the maximum number of buses defined) a partitioned design composed of P subparts with B buses at most. The resources are distributed between P subparts so as to minimize

the running time by maximizing the number of operations simultaneously executed at every control step.

The method is composed of three main tasks:

– Firstly, a local measure l of the importance of the physical link between the ressources is evaluated.

– Next, the N resources are placed in an euclidean N–dimensional metric space using l(ri,rj) as the jth coordinate of the resource Ri. The global measure between the resources is defined as the euclidean distance between the poins in this space.

– Lastly, the resources are grouped according to the minimal distance between resources sets (clusters). The distance between two clusters is defined as the loss of intercluster inertia when the two clusters merge. Beginning with all resources as individual clusters, the regrouping stops when only P clusters remain.

For most designs, this method allows the designer to improve circuits speed without a considerable overhead in silicon area.

This algorithm has been implemented in the SCOOP behavioral compiler developed at the LAMM [20] and has been successfully applied to explore design space.

ACKNOWLEDMENTS: The authors are grateful to anonymous referees for their valuable remarks.

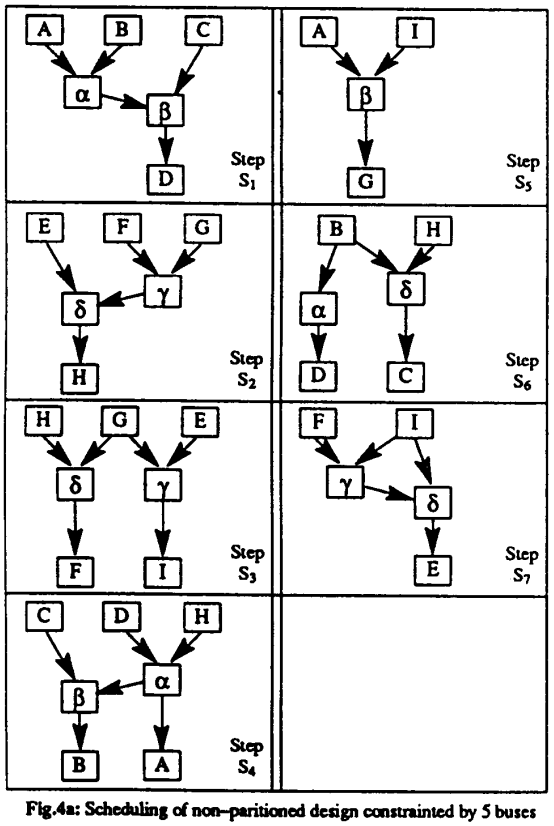


Fig.4a: Scheduling of non–paritioned design constrainted by 5 buses

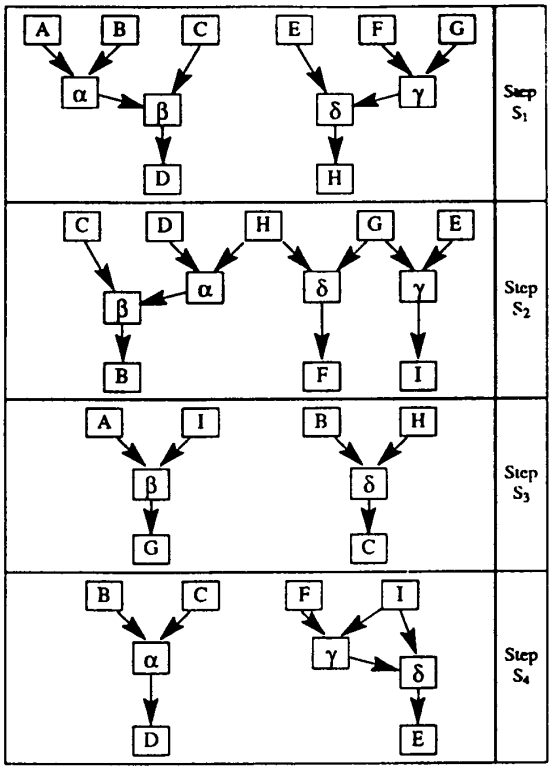


Fig.4b: Scheduling with bus constraint relaxed

	A	B	C	D	E	F	G	H	I	α	β	γ	δ
A	12												
B	0	16											
C	0	6	16										
D	5	7	6	12									
E	0	0	0	0	12								
F	0	0	0	0	2	12							
G	3	0	0	0	0	3	12						
H	3	2	3	0	3	5	2	12					
I	0	0	0	0	6	0	6	0	12				
α	8	11	4	11	0	0	0	4	8	12			
β	6	7	8	7	0	0	4	3	0	0	12		
γ	0	0	0	0	7	8	8	3	8	0	0	12	
δ	0	4	4	0	8	10	7	12	4	0	0	8	16

Fig.4c: Matrix of coordinates

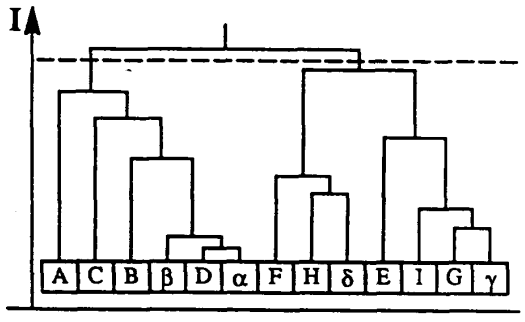


Fig.4d: Hierarchical clustering tree

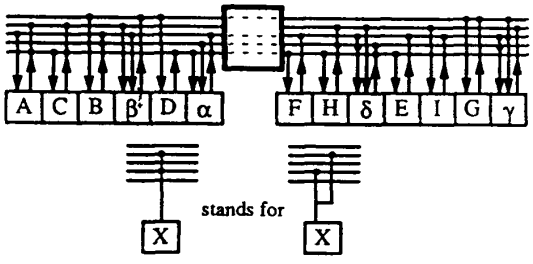


Fig.4e: Structure of partitioned design : 4 gates are necessary.

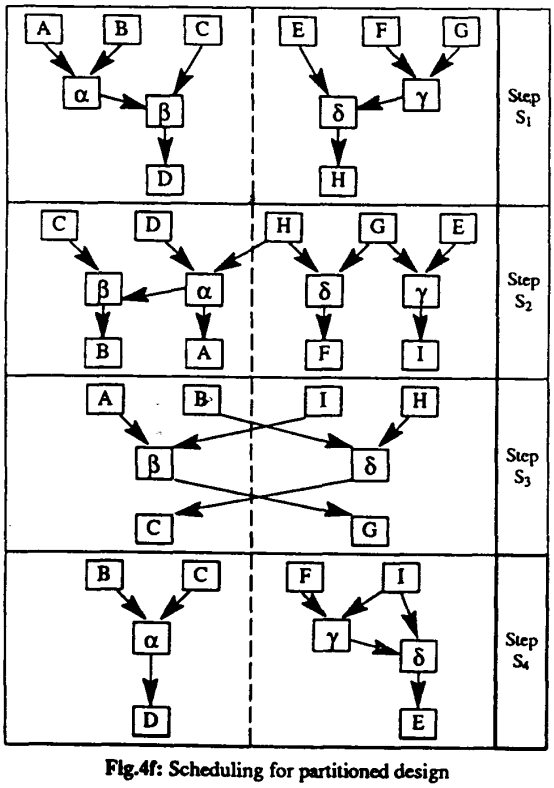


Fig.4f: Scheduling for partitioned design

REFERENCES

[1] P.G.Paulin and J.P.Knight: "Force-directed scheduling in automatic data path synthesis". Proc. of the 24th DAC, June 1987, pp: 195–202.

[2] N.Park and A.C.Parker: "SEHWA: a software package for synthesis of pipelines from behavioral specifications". IEEE Transactions on Computer Aided Design, vol 7, n° 3, March 1988, pp: 356–370.

[3] A.C.Parker, J.T.Pizarro and M.Mlinar: "MAHA: a program for datapath synthesis". Proc. of the 23rd DAC, June 1986, pp: 462–466.

[4] B.M.Pangrle and D.D.Gajski: "SLICER: a state synthesizer for intelligent silicon compilation". Proc. of the ICCD'87, Rye Brook (USA), 05/10/87, pp: 42–45.

[5] P.G.Paulin and J.P.Knight: "Force-directed scheduling for the behavioral synthesis of ASIC's". IEEE Transactions on Computer Aided Design, vol 8, n° 6, June 89, pp: 661–679

[6] B.Rouzeyre and G.Sagnes: "High level synthesis: scheduling under hardware constraints". L.A.M.M. Internal Report n° 90040, 1990.

[7] P.G.Paulin, J.P.Knight and E.F.Girzyc: "HAL: a multi-paradigm approach to automatic data path synthesis". Proc. of the 23rd DAC, June 1986, pp: 263–270.

[8] M.Balakrishnan and P.Marwedel: "Integrated scheduling and binding: a synthesis approach for design space exploration". Proc. of the 26th DAC, June 1989, pp: 68–74.

[9] R.K.Brayton, R.Camposano, G.DeMicheli, R.H.J.M.Otten and J. VanEijindhoven: "The Yorktown Silicon Compiler". In Silicon Compilation, D.D.Gajski, Ed. Reading. MA: Addison-Wesley 1988, pp: 204–311.

[10] S.Devadas, R.Newton: "Algorithms for hardware in data path synthesis". IEEE Trans. on CAD, vol 8, n° 7, July 1989.

[11] B.M.Pangrle: "SPLICER: A heuristic approach to connectivity binding". Proc. of the 25th DAC, July 88, pp: 536–541.

[12] G.DeMicheli and D.C.Ku: "HERCULES: a system for high-level synthesis". Proc. of the 25th DAC, July 1988, pp: 483–488.

[13] C.H. Tseng and D.P.Siewiorek: "The modeling and synthesis of bus systems". Proc. of the 18th DAC, June 1981, pp: 471–478.

[14] M.C.McFarland: "Computer-aided partitioning of behavioral hardware descriptions". Proc. of the 20th DAC, June 83, pp: 472–478.

[15] M.C.McFarland: "Using bottom-up design techniques in the synthesis of digital hardware from abstract behavioral descriptions". Proc. of the 23rd DAC, June 1986, pp: 474–480

[16] E.D.Lagnese and D.E.Thomas: "Architectural partitioning for system level design". Proc. of the 26th DAC, June 1989, pp: 62–67.

[17] R.Jamier and A.Jerraya: "APPOLON: a datapath compiler". Proc. of the ICCD'85.

[18] J.M.Bouroche and G.Saporta: "L'analyse des données". P.U.F. Coll."Que sais-je?",n° 1854, pp: 54–62

[19] M.R. Anderberg: "Cluster analysis for application". Academic Press. 1973.

[20] B.Rouzeyre, G.Sagnes and T.Ezzedine: "CAD for ASIC architecture". Proc. of the ISSSE89 Conf., Sept.1989, pp: 621–624.