Port Assignment for Dual and Triple Port Memories Using a Genetic Approach

C.A.Mandal[1] *P.P.Chakrabarti*[2] *S.Ghose*[2]

[1] Dept. of Computer Sci. & Engg.
Jadavpur University
Calcutta 700032, INDIA

[2] Dept. of Computer Sci. & Engg.
Indian Institute of Technology
Kharagpur 721302, INDIA

Abstract

When multi port memories are used in the design of digital systems it becomes necessary to assign the accesses to the memory to its ports. This problem is referred to as the port assignment (PA) problem. Here we present techniques for the PA of memories with two and three ports. PA occurs in conjunction with memory formation where it desirable to know the cost of PA for a packing of variables to a particular memory without the actual assignment. We present an estimator for the cost of PA of dual port memories.

1 Introduction

Memories constitute an important part of digital systems. Memories support indexed addressing which is a common programming practice. Individual variables used in a program which might otherwise have been implemented in individual registers may be packed into memories. The latter is an important source of interconnect optimization because a large number of individual registers in the design would lead to the use of more interconnection hardware. For the design of application specific integrated circuits (ASIC) we consider the use of a number of small memories (consisting of 8 to 16 cells) in the design as against the use of large ($\approx$ 1Kb) memories which are much slower. When a large number of memories are used the problem of mapping variables in these memories need to be solved for which a number of techniques exist [1, 3]. In this paper we consider a different problem arising out of the use of multiport memories in ASIC design, which we explain below.

Accesses to memories and their constituent cells have to be specified directly or indirectly by the designer at some level of abstraction. In a behavioural specification we specify variables, which are packed into memories and registers in subsequent phases of design. The variables are accessed by reading or writing from the memories. At a lower level such accesses are specified as part of transfers to and from memories. Nowadays multi port memories, mostly restricted to two and three ports, are increasingly finding use in digital systems. With multi port memories the designer either has to specify the port where the access is made and directly specify it in the HDL, or has to have an algorithm which can automatically decide on which port the access should be made. A judicious assignment of the points accessing the memory to its ports can reduce the number of interconnection elements required to achieve the connections. In the second situation the designer has the freedom to construct a behavioural specification without going into the details of the physical implementation of the reference mechanism of the variables. This enables the designer to prepare a specification for a target system using a HDL at a slightly higher level. This is the origin of the PA problem.

In the rest of the paper we first present the formulation for dual port memory PA which is followed by the GA for this problem. We also give a formula to estimate the cost of PA for this case without actually running the GA. We then present the formualation and the GA for triple port memory PA. We finally present some experimental results and close the paper with our conclusions.

2 Dual Port Memory PA Formulation

A dual port memory is equipped with two sets of address and data access ports. We assume that some variables have been packed into a dual port memory and a set of points in the circuit require data transfers with the memory. We would like to have a port assignment which will lead to the least number of points in the circuit being connected to both the ports. Since only two ports are involved, a point accessing the memory is connected to either one of the ports or to both the ports.

Example 1 Let r_1, r_2, r_3 and r_4 be registers packed into a dual port memory. Let a, b, c and d be some points in the circuit. Let us consider the following pairs of concurrent transfers of five time steps.

$$a \leftarrow r_1,\, b \leftarrow r_2; \quad b \leftarrow r_3,\, c \leftarrow r_4; \quad c \leftarrow r_2,\, d \leftarrow r_4; \quad a \leftarrow r_2,\, d \leftarrow r_3; \quad a \leftarrow r_3,\, c \leftarrow r_1;$$

If a is connected to both the ports, b and d are connected to the first port (say) and c is connected to the second port, then it may be verified that all the transfers may be satisfied. For such a connection scheme five switches would be required. The transfers will also be satisfied if b and d are connected to both the ports and a and c are connected to separate ports. Six switches are required in this scheme of connections. □

An inspection of the accesses of the memory by the circuit points in example 1 will reveal that it is not possible to satisfy the accesses, unless one or more of the points are connected to both the ports. We would, therefore, like to do the port assignment in such a way that a minimum of these points are connected to both the ports. This will lead to the minimization of the interconnect cost arising out of the port assignment. In this section we have formulated the problem of port assignment as a graph theoretic problem.

Let u and z be points accessing the memory in the same time step. Clearly u and z cannot access the memory through the same memory port in this time step. Such two points are said to have an *access conflict*. Two points having an access conflict must have connections to separate ports of the memory. Now suppose that z is connected to both the ports. Let u be connected to any one of the ports. It is easy to see that now the access conflict of u and z can always be resolved. This is because, no matter from which port u accesses M, z can always access M through the other port. On the basis of this argument, we also note that the access conflict of z with any other point accessing the memory will also be resolved if z is connected to both ports. This observation is fundamental to the formulation of the problem. Our formulation is as follows.

A conflict graph of points accessing the memory is defined to be a graph where each circuit point corresponds to a vertex and an edge is present between a pair of vertices if their corresponding points are in conflict. An attempt is made to colour this graph with just two colours, $\{0,1\}$, say, with the constraint that no two vertices connected by an edge (that are in conflict) have the same colour. This will be referred to as the two colouring of the graph. Two Colourability can be easily tested by checking up whether there are any *odd cycles* in the graph. If the colouring is successful then vertices coloured as 0 are connected to one port, and the vertices coloured as 1 to the other. Such an assignment of ports will clearly satisfy all the data transfers to and from the memory. In general the graph will not be two-colourable and it will be necessary to delete sufficient number of vertices to make it 2-colourable. A vertex that has been deleted will be connected to both the ports of the memory. The access conflicts of such a point with any other point can always be resolved.

This permits a straight forward formulation of the problem. In order to make the graph 2-colourable at minimum cost, the least number of vertices should be deleted from the conflict graph, which is the Minimum Node Deletion problem (MND).

3 GA for the Minimum Node Deletion Problem

We now present the genetic algorithm for node deletion which we shall refer to as GA2. We also present an estimator to estimate the cost of node deletion without actually running the GA.

3.1 Algorithm for MND

Solution representation Convention bit string representations sometimes mask the structure inherent in the solution. For MND we have found it convenient to represent the solution directly as three sets. The first two sets contain the vertices corresponding to each of the two colours. The third set contains vertices that could not be two coloured and which are chosen for deletion.

Fitness function The fitness function is defined as

$$g = |set \;\; of \;\; deleted \;\; vertices| \qquad (1)$$

Minimization of g is the objective.

Initial population generation Each member of the initial population is a randomly generated valid solution. While generation a solution, each vertex of the graph is tested for possible membership in one of the two partially constructed colour classes. In case of a failure in inclusion to one colour class, the membership for the other colour class is checked. In case of a repeated failure the vertex is marked for deletion. The sequence in which the vertices are visited while constructing a solution is also random.

Reproductive plan The parent selection policy, crossover and mutation operations constitute a reproductive plan. For a particular generation we have selected two parent solutions for crossover to generate each child solution, randomly without replacement from the current population. Only one offspring has been generated as a result of a single reproduction. The number of reproductions performed in one generation is determined by the crossover rate.

The crossover is performed as follows. The offspring solution first inherits a subset of a colour class from one of the parent solutions. During crossover, larger colour classes are chosen for inheritance with a higher probability, while smaller ones are selected with a lower probability. The inherited class is now augmented with uncoloured vertices according to the algorithm in figure 1. The augmentation is based on a graph colouring algorithm presented in [2]. The algorithm is successively applied to the two inherited colour classes. The augmentation algorithm is as follows. Let V be the set of vertices of the graph. Let the initial inherited colour class on which the algorithm is applied be X. $\Gamma(X)$ is defined as the subset of $V - X$, such that for each element of the subset there is an element in X to which it is connected by an edge. The effect of the steps (1) and (5) of the augmentation algorithm is to remove from Y all those vertices which have an edge with at least one vertex of X. The newly defined Y has the property that any of its vertices can be augmented to X. The process of augmentation continues till the set Y becomes empty. The vertex in Y that is to be selected is determined by the simple heuristic of step (3) of the algorithm.

After the first colour class of the offspring is formed, the second colour class is also formed similarly. The set from which the second colour class is inherited is as follows. Let P_1 and P_2 be the two parents. Let S_{11} and S_{12} be the two sets in P_1. Similarly, let S_{21} and S_{22} be the two sets in P_2. Suppose that the first colour class had been formed from S_{11} of P_1. Let

$$c_0 = \frac{|S_{11} \cap S_{21}|}{|S_{11} \cup S_{21}|} \quad and \qquad c_1 = \frac{|S_{11} \cap S_{22}|}{|S_{11} \cup S_{22}|}.$$

The values c_0 and c_1, $0 \leq c_0, c_1 \leq 1$ represent the affinity of S_{11} with S_{21} and S_{22} of P_2, respectively. Let $S = S_{2(i+1)}$, such that $c_i \leq c_{1-i}, i \in \{0,1\}$. S is the colour class of P_2 which is less affine to S_{11}. Normally the second colour class of the offspring is inherited from S, otherwise inheritance is from S_{12}. The vertices that could not be included in the two colour classes of the offspring are placed in the third set for deletion. This method of construction ensures that each solution constructed is a valid solution.

During crossover, inheritance of only a part of the colour class may be considered equivalent to the process of mutation. The *mutation rate* instead of being kept fixed, is varied with the standard deviation of the fitness value of the candidate solutions. Should the fitness function values tend to become uniform the mutation rate goes up.

Replacement policy In our implementation all the offspring generated in the current generation replace the maximum cost solutions in the current population.

We state without proof that with the above crossover mechanism the GA finds an optimum solution *with high probability (whp)*, as formalized in [2].

1. $Y = V - \Gamma(X)$
2. while $(Y \neq \emptyset)$
3. $\{$ among all $y \in Y$ let x have the minimum degree in Y
4. $X = X \bigcup \{x\}$
5. $Y = Y - (\{x\} \bigcup \Gamma(x))$
6. $\}$

Figure 1: The augmentation algorithm

3.2 Estimation of Minimum Number of Nodes Deleted

We only present the empirical formula used to estimate the number of nodes that need to be deleted to make a random graph bipartite. Let $G_{n,p}$ be a random graph, with n nodes and each of the $\frac{n(n-1)}{2}$ edges being present with probability p. Let $q = (1-p)$ be the probability that the edge is absent. We compute two quantities S_1 and S_2 such that:

$S_1 = \sum_{k=1}^{\lfloor \frac{m}{2} \rfloor - 1} \binom{m}{k} q^{\frac{k(k-1)}{2} + \frac{(m-k)(m-k-1)}{2}} (1 - q^k)(1 - q^{(m-k)})^d$. S_1 represents the expected number of bipartite partitions of m nodes when k are in one partition and $m - k$ are in the other, $k = 1..\lfloor \frac{m}{2} \rfloor - 1$,

$S_2 = \binom{m}{l} q^{\frac{l(l-1)}{2} + \frac{(m-l)(m-l-1)}{2}} (1 - q^l)(1 - q^{(m-l)})^d$

S_2 represents the expected number of bipartite partitions of m nodes when l are in one partition and $m - l$ are in the other. S_2 is computed separately to avoid double counting when m is even. The expected number of bipartite partitions of $G_{m,p}$ is $N_{m,p} = \left\{ \begin{array}{ll} S_1 + S_2 & \text{if m is odd} \\ S_1 + \frac{S_2}{2} & \text{if m is even} \end{array} \right\}$. For $G_{n,p}$ the expected number of bipartite partitions if d nodes are deleted at random is $M_{n,p,d} = \binom{n}{d} N_{n-d,p}$.

Let $B_1 = M_{n,p,d}$, for the minimum value of d such that $B_1 \geq 1$. Let $B_2 = M_{n,p,d}$, for the value of d which maximizes B_2. Let $B_P = \sqrt{B_1 B_2}$. Find the maximum value of $d = d_P$ for which $M_{n,p,d} \leq B_P$. This has been used as an estimate of $E(d)$. It has been experimentally observed that this estimate closely matches the cost of the actual solutions returned by the GA.

4 Formulation for Triple Port Memory PA

A triple port memory has three sets of address and data access ports. As usual we would like to perform the assignment so that the total number of links of the circuit points to the ports of the memory is minimized. Here too, we develop a graph theoretic formulation. But unlike the dual port memory PA case we are not able to map it onto a conventional graph problem. For a three port memory either one or two or three points of the circuit can access the memory in one time step. The access of the memory by a single point of the circuit in one time step does not require any special consideration. Access of the memory by two or three points, in a time step, gives rise to two different kinds of access conflicts. In particular, when three points access the memory in the same time step, it is necessary that our connection scheme of these three points to the ports should permit us to satisfy these access through distinct ports.

We construct a hyper graph to reflect the constraints imposed by these accesses. Let there be n points accessing the triple port memory. Let V be the set of vertices of of the conflict hyper graph H, $|V| = n$. An edge of H may include two vertices for access by two or three points in a particular time step. Such hyper edges will be called 2-edge and 3-edge, respectively. Thus if points p_1 and p_2 access the memory in a time step then H will have an 2-edge $< p_1, p_2 >$. Similarly, for access by points p_1, p_2 and p_3 the 3-edge $< p_1, p_2, p_3 >$ will be present in H. A 3-edge implies each of its two edges and these need not be considered. We shall refer to CHGs where such edges are absent as *reduced*. Each port is represented by a colour. For the three ports we arbitrarily designate three colours R, G and B. The vertices in H must, therefore, be assigned colours. To model the connection of a point to multiple ports we permit a vertex to by assigned any non-empty combination of the three designated colours. For a vertex $v_i \in V$, let s_i be the set of colours assigned to it.

Definition 1 *A set of colours $s \subseteq \{R, G, B\}$ will be called a 2-colour if $|s| = 2$.*

The colours assigned to the vertices should satisfy the following properties so that the accesses to the memory will be satisfied.

1. If vertices v_1 and v_2, coloured with s_1 and s_2, such that $|s_1| = |s_2| = 1$ and $s_1 = s_2$, then v_1 and v_2, together, should not be part of any edge in H. This is to ensure that two points connected to the same and only port of the memory are free of all access conflicts.

2. If v_1 and v_2, coloured s_1 and s_2 are connected by a 2-edge, then there should be colours c_1 and c_2 such that $c_1, c_2 \in \{R, G, B\}, c_1 \neq c_2$, and $c_1 \in s_1$ and $c_2 \in s_2$. This ensures that two vertices that access the memory simultaneously are connected to separate ports so that their access can be satisfied.

3. Similarly, if v_1, v_2 and v_3, coloured s_1, s_2 and s_3 are connected by a 3-edge, then there should be colours c_1, c_2 and c_3 such that $c_1, c_2, c_3 \in \{R, G, B\}, c_1 \neq c_2 \neq c_3$, and $c_1 \in s_1, c_2 \in s_2$ and $c_3 \in s_3$.

If a point is connected to all the three ports then its access conflict with any other point accessing the memory can always be resolved and the vertex for such a point in the CHG will be considered deleted. However, if a point is connected to only two ports of a triple port memory then some access conflicts may still remain to be resolved. The colouring process is now explained.

Since the colouring of the CHG is relatively more involved, first a check is made to see which vertices may be assigned single colours using graph colouring algorithms for simple graphs. For this purpose a projection of the CHG H to a simple graph G is defined as follows. G has the same (isomorphic) set of vertices as H. Two vertices in G are connected by an edge if they are connected by a 2-edge or a 3-edge in H. We attempt to colour the simple graph G with the three colours R, G and B. If the colouring is successful then the port assignment may be readily performed with each point being connected to just one port, of the corresponding colour. If the colouring is unsuccessful, then some vertices still remain uncoloured. The corresponding circuit points will have to be connected to two or three ports, i.e. they will have to take two or three colours. The single colours being R, G and B, the two colours will be $\{R, G\}, \{G, B\}$ and $\{R, B\}$, to be referred to as RG, GB and RB, respectively. RGB will be used to denote the 3-colour $\{R, G, B\}$.

Definition 2 *Given a 2-colour $s, s \subset \{R, G, B\}$, the colour c, such that $c = \{R, G, B\} - s$, will be called the resolving colour of s.*

Definition 3 *Suppose that a vertex has been assigned a 2-colour s_1. Vertices v_1 and v_2 will be said to constitute a c-conflict if s_2, coloured v_2 satisfies $s_2 \subseteq s_1$ and v_1, v_2 and some other vertex v form a 3-edge.*

Definition 4 *Let V be the set of vertices of a CHG and $< v_1, v_2 >, v_1, v_2 \in V$. The set of vertices $O = \{v | v \in V, < v_1, v_2, v >$ is 3-edge and v_1, v_2 make a c-conflict $\}$, will be called the orbit of $< v_1, v_2 >$.*

If a vertex v is assigned a 2-colour then, the vertices of the orbit of each c-conflict of which v is a member must possess the resolving colour. Unless this condition is met some of the accesses to the memory will not be satisfied.

We can conceive of the colouring of the CHG as follows. First we form the projection of the CHG and try to colour it using three colours. If it is successful then we are done. Otherwise, some of the vertices will have to be assigned 2-colours. In this process it might be desirable to re-colour some of the vertices that were coloured earlier. The vertices that could not be assigned even a 2-colour are now assigned 3-colours.

4.1 GA for Triple Port Memory PA

The basic scheme for this GA is similar to the one for dual port memory PA. We, therefore, present only the fitness function and the reproductive plan. We shall refer to this GA as GA3.

Fitness function Let V be the set of vertices of the CHG s_i be the set of colours assigned to vertex v_i. The fitness function is defined as

$$g = \sum_{i|v_i \in V} |s_i| - |V| \tag{2}$$

This cost function models the additional cost incurred due to the connection of a point to more than a single port. Minimization of g is the objective.

Reproductive plan The parent selection strategy is similar to the GA for MND. The crossover has been tailored to suit this problem.

The first thing to note is that the CHG is a labeled graph and each colouring of the CHG may be associated with two other isomorphic colourings which result from a renaming of the single colours. The 2-colours, however, cannot be renamed independently of the single colours, in general. The required renaming of the 2-colours is implied by the renaming of the single colours. It is desirable to take this non-trivial isomorphism into account, as far as possible, while performing crossover. This is necessary to avoid a sub-standard crossover between two parent solutions which are isomorphically proximal. Thus the first step in performing crossover is to derive a one-one mapping between the single colour classes of the two solutions. The crossover proceeds with the formation of the singly coloured sets of vertices, followed by the sets of vertices coloured with two colours. The remaining vertices are placed in the 3-colour set. The formation of the singly coloured set of vertices is similar to the way the colour classes are formed for the GA for MND. Thus a subset of a set of vertices is first inherited. This set is then augmented with with vertices, if possible. While augmenting a set of singly coloured vertices it is only necessary to check that the new vertex does not have an edge (2-edge) with any of the existing members of that set. The choice of sets for inheritance is made as follows. Vertices are first inherited from the largest set of singly coloured vertices of the better fit parent. At the same time the set in the second solution on which this set had been mapped is marked off. The larger of the remaining two sets in the second solution is used to form the next set in the child solution. At the same time the corresponding set in the first solution is marked off. The remaining set of singly coloured vertices in the first solution is used to construct the third set of singly coloured vertices in the child solution.

The sets of 2-coloured vertices are now formed in the child solution. The membership of vertices in a 2-colour set is governed by the vertices which constitute the c-conflicts and their associated orbits. The vertices of the orbits will certainly belong to some other set. In the child solution these vertices, in general, will not belong to similar sets of the parent solution. Therefore, an arbitrary subset of a secondary colour class may not be feasible. Thus each vertex v that is introduced to a set of doubly coloured vertices has to be checked for membership in that set. As noted in section 4 the membership criterion will be satisfied if each vertex of the orbit of a c-conflict of which v is a member has the resolving colour. The method of choosing the sets from which the inheritance is to be done is similar to that used for forming the sets of singly coloured vertices. Finally, the vertices which could not be included in any of these six sets are placed in the set of vertices which will be given the 3-colour (RGB). In most cases the colouring of offspring will have some differences with respect to both the parents. In view of this we do not keep a separate provision for mutation, as further forced changes might weaken the crossover significantly.

5 Experimentation

The algorithms have been tested extensively, on cases where the number of accessing points is small, medium and large. For dual port memories the results that we have obtained are excellent, while those for triple port memories are also promising. The testing scheme is summarized as follows:

- For sixteen accessing points GA2 has been compared against the optimal solutions obtained exhaustively. In all cases there was exact match in the cost of the solution.
- We have exercised both GA2 and GA3 on large number of problems for which a tight upper bound on the solution cost is known. The problems have been classified according to their graph parameters.
- The estimator for dual port memory has been tested on a large number of random graphs and the estimate has been compared with the actual results obtained by the algorithm.

Experimental results have been tabulated for the techniques developed for dual port memories. In the tables *nodes* refer to the points accessing the memory. The parameter *edge prob.* is the probability of the presence of an edge in the graph which models a specific instance of the problem. The cost estimates produced by the estimator match extremely well with the cost of the GA solution. Results for the estimator have been presented in table 1. Table 2 highlights the use of the testing technique that we have developed.

References

[1] D. K. Banerjee, J. C. Majithia, T. C. Wilson, A. Basu, S. Sutarwala, and A. K. Majumdar. High-level synthesis of data-paths from a behavioural description. *International Journal of Computer Aided VLSI Design*, 3:367–391, 1991.

[2] A. M. Frieze. *Probabilistic Analysis of Graph Algorithms.* Springer-Verlag, 1989.

[3] C. A. Mandal, P. P. Chakrabarti, and S. Ghose. Interconnect optimization techniques in data path synthesis. In *Proceedings of VLSI Design '92, Bangalore*, pages 85–90, 1991.

no. of. nodes	edge prob.	prag. est.	deletion by GA2 [a]
20	0.154	2	2.709
20	0.250	5	5.516
20	0.400	8	8.870
20	0.500	10	10.161
20	0.650	12	12.226
30	0.154	7	6.935
30	0.250	11	11.129
30	0.400	16	16.032
30	0.500	18	18.064
30	0.650	21	20.935
40	0.154	12	12.064
40	0.250	18	18.322
40	0.400	24	24.000
40	0.500	27	26.741
40	0.650	30	29.712
60	0.154	25	25.129
60	0.250	34	33.419
60	0.400	42	41.193
60	0.500	45	44.870
60	0.650	49	48.667

[a] Average on 30 individual random graphs.

Table 1: Comparison of the estimator against the number of nodes deleted by GA2.

no. of. nodes	edge prob.	deletion [a] by GA2	cost u.b.
20	0.250	3.483	4
20	0.400	6.903	7
20	0.500	8.903	9
20	0.650	10.70	11
30	0.250	8.677	9
30	0.400	13.61	14
30	0.500	16.80	17
30	0.650	19.70	20
40	0.250	15.35	16
40	0.400	22.29	23
40	0.500	25.48	26
40	0.650	28.77	29
60	0.082	7.451	8
60	0.107	12.38	13
60	0.154	20.38	21

[a] Average on 30 individual random graphs.

Table 2: Performance of GA2 on random graphs where an upper bound on the number of nodes to be deleted is known