2.1.3 Total Wire Length

Intimately tied to chip area is total wire length. If the routing solution is optimized such that overall chip area decreases, then total wire length usually drops as well. Shorter wires contribute significantly to the manufacturability of the circuits, since longer wires have a much higher probability of a defect. Additionally, from an electrical standpoint, the longer the wire the higher the capacitive load. For lines with RC charging characteristics, the increase in length limits the bandwidth of the connections. For these reasons total wire length are extremely important.

2.1.4 Number of Vias

The last of the top four characteristics associated with routing solution quality is vias count. Each via represents an electrical connection that carries with it the potential of failure. For critical nets, the propagation time can be adversely affected by the number of vias that must be traversed. In the case of wires designed as transmission lines, vias can introduce reflections, just as connection stubs.

Of the factors discussed, vias are the most dependent upon the routing algorithm. The majority of all current algorithms attempt to minimize total wire length and chip area, using vias as a tradeoff factor. Consequently, the degree to which vias are minimized is typically a direct result of the routing algorithm itself. For a "perfect" routing system, the ability to tailor the algorithm by net type would be a capability to consider incorporating.

Having addressed the four primary metrics used to assess the quality of a routing solution, a review of the significant algorithms will now be conducted.

2.2 MAZE RUNNING

One of the earliest automated routing algorithms was the maze running algorithm published by Lee[Lee61]. This approach begins by constructing a grid that represents the entire routing space. Given the two points to be connected by a net, wave fronts are expanded about one. This is usually simulated by inserting an integer number representing the wave propagation iteration in each of the nearest neighbor cells. This is repeated until the target point is encountered. Then, by moving from integer to integer in sequential fashion, the route can be constructed between the two.

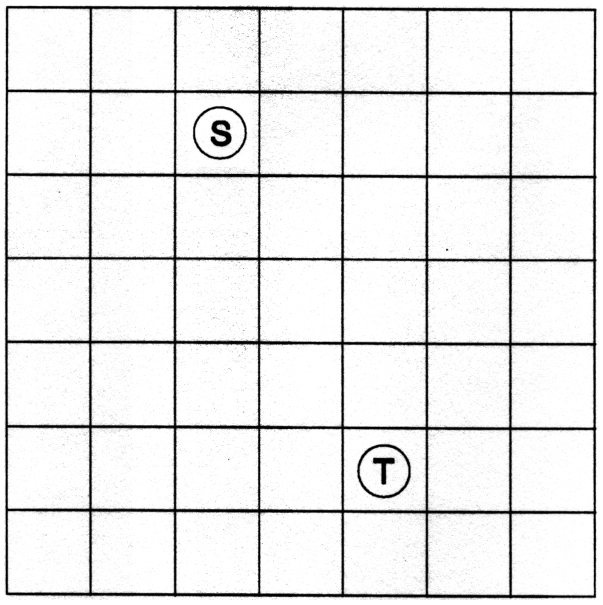

FIG. 2.1 START CONFIGURATION FOR LEE

The diagram shown in Fig. 2.1 presents the starting configuration for the maze running algorithm. The start point is identified with the *S* and the destination is identified by the *T*. The routing grid has been superimposed showing the available vertical and horizontal routing tracks. From this initial configuration, a wave front is propagated outward from the start point. At the first time step, a one is placed in each of the four nearest neighbor cells that are vacant. At each succeeding time step, an integer value corresponding to the time step is placed in the nearest neighbor cells of the most recently added locations. In this way, the effect of a spreading wave is simulated. Advancing two time steps produces the intermediate configuration shown in Fig. 2.2.

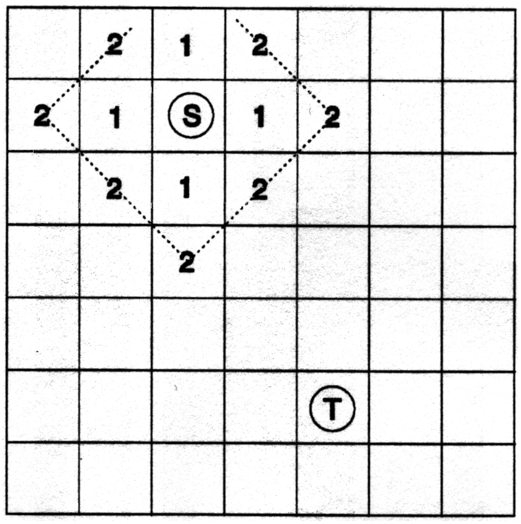

FIG. 2.2 TIME STEP 2 OF LEE ALGORITHM

Continuing to advance the time steps produces the final configuration shown in Fig. 2.3.

3	2	1	2	3	4	5
2	1	Ⓢ	1	2	3	4
3	2	1	2	3	4	5
4	3	2	3	4	5	6
5	4	3	4	5	6	
6	5	4	5	Ⓣ 6		
	6	5	6			

FIG. 2.3 FINAL LEE CONFIGURATION

Once the propagating wave encounters the target point, the spreading ceases. Then, by walking from the target point to the start point, following a decreasing integer sequence, the path between the two points can be traced. For our example, the final path is depicted in Fig. 2.4.

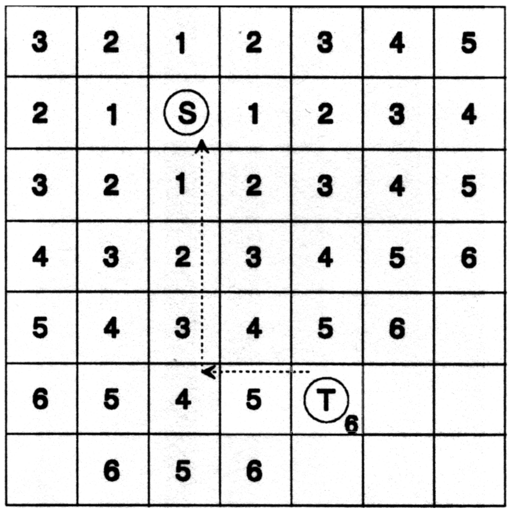

FIG. 2.4 PATH TRACE

An examination of the integer "walk" between the points reveals that there can be multiple paths between the two. Additionally, these paths may have different numbers of turns. The path with the fewest turns can be identified by adding the stipulation that direction of the walk is only altered if necessary.

The original Lee algorithm was defined for single layer routing, but was easily adapted to the two layer problem space[Heis68]. The major advantage of the maze running class of algorithms, is that if a connection can be constructed between the two points of interest, the algorithm will discover it. However, the drawbacks of the approach are the run time and amount of storage space required. For large chips with extremely fine wiring pitch, the wave storage grid can become prohibitive. Alternatives have been proposed to pare this space, such as limiting wave expansion to a single quadrant. The time involved to propagate the wave between widely separated points is significant, since it requires updating of all intermediate grid points at each time step as the wave spreads. These factors must all be considered when designing new and more capable routers.

2.3 LINE PROBE

In an effort to overcome the storage requirements associated with maze running algorithms, line probe routers were constructed[High80]. The additional advantage of the line probe routers were their dramatic improvement in speed. The Hightower router begins with a start point and a destination point. From each, rays of two lines are constructed. These emanate in both the horizontal and vertical direction. If the line from one point intersects the line emanating from another point then a route between the two has been discovered. An example how the line probe router operates is provided in the following paragraphs.

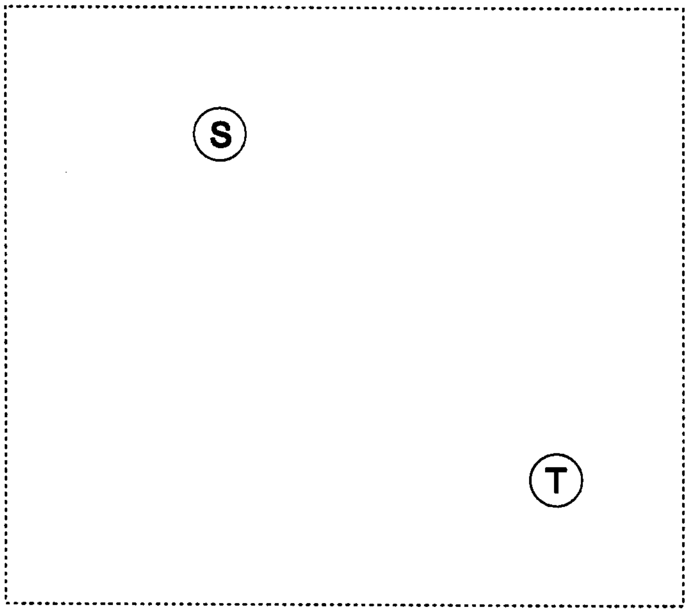

FIG. 2.5 START CONFIGURATION FOR HIGHTOWER

An initial configuration depicting two points that must be connected is shown in Fig. 2.5. Next, lines are generated that extend in both the horizontal and vertical direction from each of the points. This is portrayed in Fig. 2.6, where ray 1a intersects 2b, and 1b intersects 2a. The intersecting lines provide the immediate routing track solution. One solution is shown in Fig. 2.7.

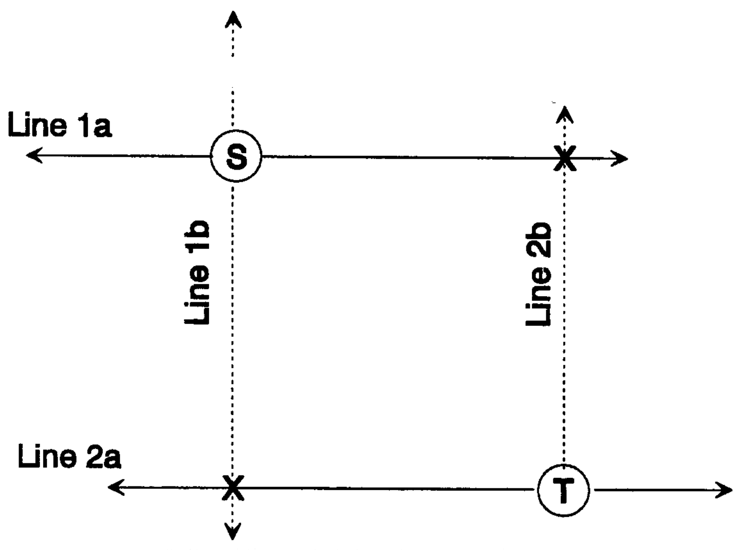

FIG. 2.6 LINE PROBES EXTENDED

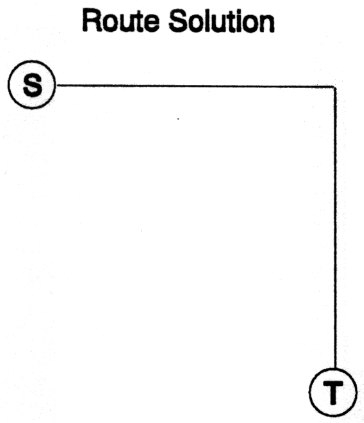

FIG. 2.7 ROUTING TRACK SOLUTION

If the initial probe lines fail to intersect, then further escape points can be selected along those original lines. Each of the escape points is then treated as an origin or target, and the ray projecting technique repeated.

The advantages provided by the line probe solution include a considerably reduced memory requirement, and an improvement in run time performance over the maze running algorithms. Line probes and potential intersections can be generated and quickly tested. The major shortcoming is their completion capability. Where the maze running approach guarantees to find a solution if one exists, the line probe technique does not. Therefore, as congestion levels increase, the likelihood of a routing failure also increases. Again this is an important tradeoff issue that must be considered in router design.

2.4 CONSTRAINED PROBLEMS

In addition to the two basic approaches, a third category of solution has emerged. It generally involves a more constrained routing space. Two examples of such regimes are the channel and the switch box routing problems. By introducing constraints on where nets travel along with the direction and layer type involved, graph theory can be brought to bear on the problem.

The vertical constraint graph is the one most commonly employed. The difficulty with its use is the computational time required to construct the graph itself. Also, should the net map into a constraint graph with a cycle, a solution cannot be derived until a partitioning step is applied to the net. After splitting, the components will then yield two acyclic constraint graphs from which a solution can be constructed[Wada81].

2.5 HEURISTICS & AI TECHNIQUES

As improvements were introduced to the basic algorithms, along with those for the more constrained environments, the use of heuristics and artificial intelligence techniques began to be employed. One of the first routers to utilize heuristics was the Greedy Channel router[Rive82]. By using a small number of rules the run time of the channel routing problem could be reduced considerably. However, by selecting nets in the "greedy" manner, local optimum were emphasized. This can have the impact of creating situations where the decisions with regards to routing one of the early nets can adversely affect the routing of later nets. It may prove that the channel is un-routable, or the width of the channel will be increased unnecessarily.

In addition to the normal heuristics, the routing problem appeared well suited for the application of knowledge based or "expert" systems. A thorough example of applying expert systems ideas to the routing problem is provided by Joobani in his doctoral thesis[Joob86]. His router, WEAVER, utilizes a number of modules, each of which is configured as an expert. There is a Congestion Expert, a Via Expert, a Steiner Tree Expert, and a Common Sense Expert to name just a few. Each is responsible for interacting with the chalkboard communication mechanism, and providing its input to the final solution. By capturing what he felt to be the essence of each expert in a rule based,

IF *condition* THEN *action* or *rule firing*

system, he was able to demonstrate objective improvements in router system performance over any of the basic algorithm techniques employed independently.

2.6 SUMMARY

Both knowingly and unknowingly, mankind has been solving different forms of the routing problem since the earliest times. Only recently, and most predominantly in the integrated circuit arena, has the cost of a less than optimal solution to the routing problem had such severe penalties.

For unconstrained problems, the maze running and line probe algorithms provide a solid basis on which to build. A system incorporating the ideas from both techniques coupled with proven heuristics and an expert system control structure seems the logical next step[Ohts86]. Line probe methods can route the early nets, and as congestion increases, region specific maze running algorithms can take over to increase the completion percentage. Knowledge based conditions can be tested periodically to see that the quality metrics remain within reasonable limits.

The literature coverage of these traditional problems is extensive. Yet apparently, due to the obvious cost of increased interconnect associated with differential signal routing, that problem has not yet been addressed. In developing my solution to the adjacent placement problem of differential signals, it is the solid foundation derived from a thorough review of the literature upon which I will build where possible.

CHAPTER 3

Managing Differential Signal Placement

NOVEL ROUTER PROCESS

3.0 General Thoughts

For several years, the RPI design team has envisioned a work around solution for accomplishing differential signal routing. In the absence of commercially available tools, it was believed that a conventional router could be coaxed into providing a routing solution using large geometry or "fat" wires. These wires could then be split, yielding an adjacent placement solution. Brian Donlan began the investigation when he developed his wafer scale routing system[Donl86]. It was followed by the work of Mark Russinovich[Russ90]. Each of these early efforts recognized that there were certain net topologies that could not be split without reverting to solving the underlying routing problem. As a result, they only achieved a partial solution, and due to the nature of the work, solving the wafer scale integration routing problem, were never in the position to construct working chips.

When the Advanced Research Projects Agency (ARPA), funded the 1ns RISC Processor research effort, the ability to perform differential routing migrated from an interesting area of research to that of a requirement. This chapter proposes a novel architecture for dealing with the adjacent placement problem for complimentary wire pairs.

3.1 THE TRADITIONAL APPROACH

Conventional scientific and engineering methodologies occasionally fall short when dealing with a new instance of a problem. Applying traditional wisdom, researchers attempt to extend current solutions incrementally. Differential routing is one problem instance where the conventional approach yields something less than the optimal solution.

If a conventional router is given the task of handling the differential routing problem, significant penalties accrue. First, the number of nets that must be routed, *N*, is immediately doubled over what would have to be routed in a single ended design of equivalent complexity. This factor alone acts to double the run time, assuming the underlying algorithm were linear to begin with. Unfortunately, even the basic Manhattan routing problem in standard cell regions has been shown to be NP-complete[Joy92][Ohts86]. As problem size grows, run time increases. Using heuristics to prune the search space, and thereby cut the run time, sacrifices a certain degree of optimality. This tradeoff is necessary in order to arrive at a solution in acceptable time. This approach does not guarantee adjacent placement of the differential pairs throughout their runs. Certainly, a moderate degree of adjacent tracking will take place. Inevitably, obstructions will occur that will force the wires to diverge from traveling in adjacent tracks.

The problem size increase could be dealt with by solving the routing problem for only one wire of every pair. The solution for laying the complement is to place it in the adjacent track along the run. This assumes there are no obstructions along the entire adjacent track. Obviously, the router's work in assuring a free track for the complimentary wire is not as computationally intensive as routing the other wire, but it is not trivial. Handling inversions complicate the problem still more.

The next significant obstacle a conventional router would encounter is the concept of signal inversions. When single ended routers use the net list abstraction, the concept of crossing wires does not exist. The router only needs to insure that the terminals of each net are connected properly. When the adjacent placement constraint is imposed on signal pairs, the router is forced to deal with the problem of wire inversions. This immediately complicates the approach of just laying the second wire of a pair in an adjacent track. Consideration must be given to providing the capability of handling signal inversions. To guarantee that a signal inversion can be achieved during the run of a signal pair, a block out region must be established to permit the interchange. One method of forming this region is to block adjacent tracks on both sides of the first signal of a pair to be routed, as depicted in Fig. 3.1.

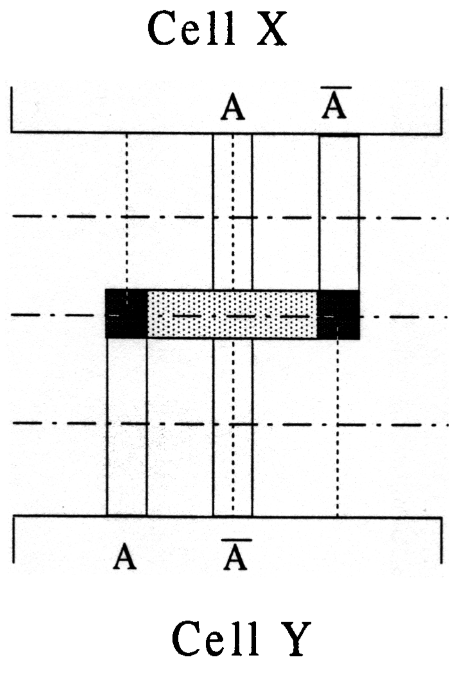

FIG. 3.1. THREE TRACK BLOCK OUT REGION

When the router establishes its grid, a three track zone is used. The center track is used for the primary wire solution. Then, when placing the compliment, a free track is guaranteed on either side of the primary throughout the run. In this way, regardless of whether or not an inversion occurs, the proper connection can be achieved. The example in Fig. 3.1 shows the three track block out zone from Cell X to Cell Y. In this figure, a wire inversion is required. The block out region allows the inversion by guaranteeing space to accomplish the cross over in the second metal layer. The advantage of this approach is that it effectively reduces the nets that must be routed. However, the maximum routing density, without compaction, is greatly reduced by either poorly or completely unused track segments.

It is possible to accomplish inversions by only allocating two tracks to the pair instead of three.

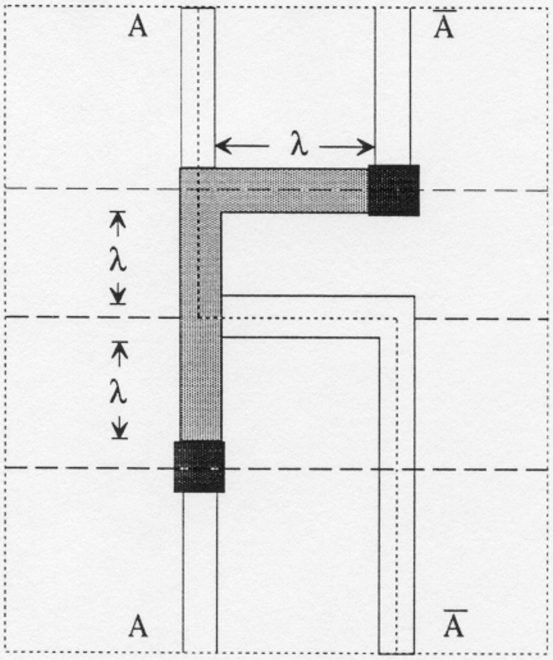

FIG. 3.2. TWO TRACK INVERSION TECHNIQUE.

In order to assure the block out region can actually handle the inversion, this solution must now obstruct the two original block out tracks, plus three orthogonal tracks. An example of how such an approach could be implemented is given in Fig. 3.2. The total block out area is a three track by two track box. Once this pair has been routed, the obstruction region used for the inversion must be added to the router database. When the router calculates the next net path, it must take into account the forbidden zone generated by the previous pair. Even though the second approach is an improvement, it too wastes considerable routing area.

3.2 BREAKING DOWN MENTAL WALLS

Some would argue that the evolutionary path of discovery I followed was straightforward, and that my solution is obvious. I contend this is not necessarily the case. From a router builders' perspective, the inertia contained in a functioning system somehow impels them to draw heavily from that design. Even if they were provided a clean sheet of paper and given full design freedom, their thinking and approach is naturally inhibited by what they have come to know and with which they are comfortable. Given this natural working environment, I strongly believe that one of the incremental improvement approaches would be the likely avenue of advance.

In a manner of speaking, I was operating in a restricted environment. Access to commercial tools was limited. Time was not available to construct a new router from scratch. But in this case, what appeared as an over constrained situation, actually served to produce an intellectual leap. Rather than construct a new router, a new router architecture was formulated.

As was mentioned at the outset of this chapter, attacking the differential routing problem head on requires the router to face twice the number of nets that would ordinarily have to be routed for a single ended design of equivalent complexity. Instead, an alternative approach was envisioned: one which made a conceptual leap. The hybrid architecture of Fig. 3.3 was developed. It operates by preprocessing net lists so that

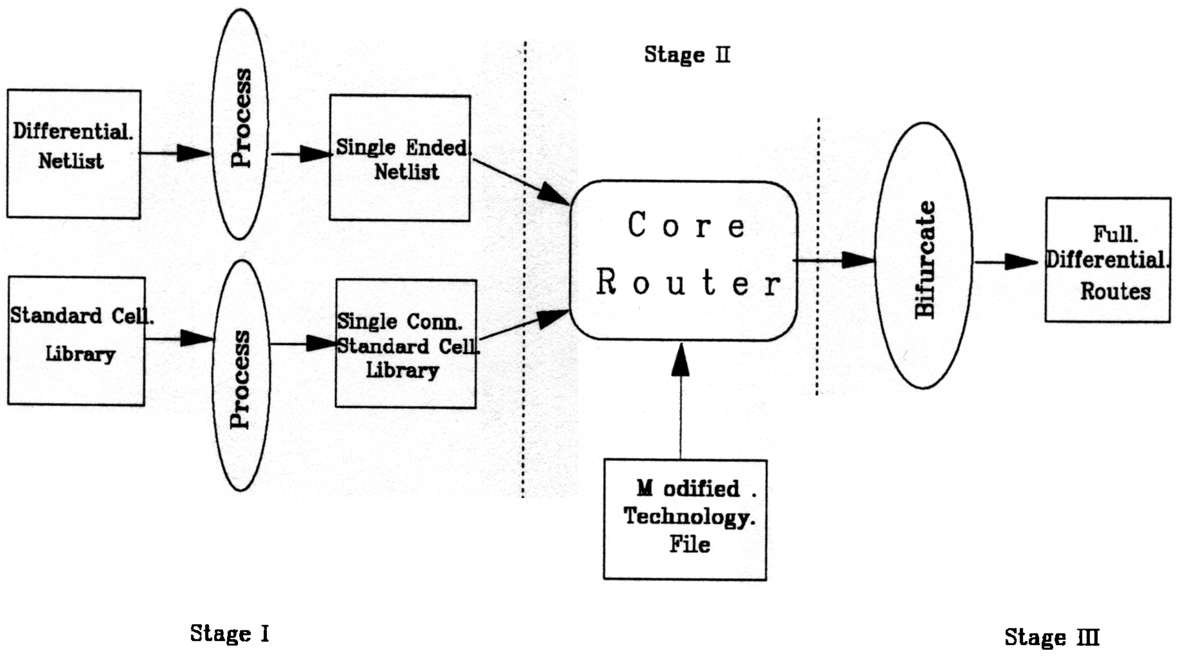

FIG. 3.3. NOVEL ROUTER ARCHITECTURE

differential pairs are rewritten as single logical nets, routing these single logical entities (SLE), and then bifurcating the resulting wires. This achieves adjacent placement of differential signals, while at the same time making use of the best, currently available commercial routing tools designed to handle single ended wiring.

3.2.1 COMPUTATIONAL COMPLEXITY ANALYSIS

The architecture is partitioned into three stages. Stage I is comprised of several preprocessing steps. An image of the standard cell library is generated. The cells in this image library have each differential port pair merged into a single logical instance, Fig. 3.4.

The original net list is collapsed, so that all differential entries are transformed into SLE nets, as depicted in Table 1. While merging the pairs inversions must be identified and extracted, since inversion information is not preserved in collapsed form. These steps can be completed in linear time through scanning and translation operations.

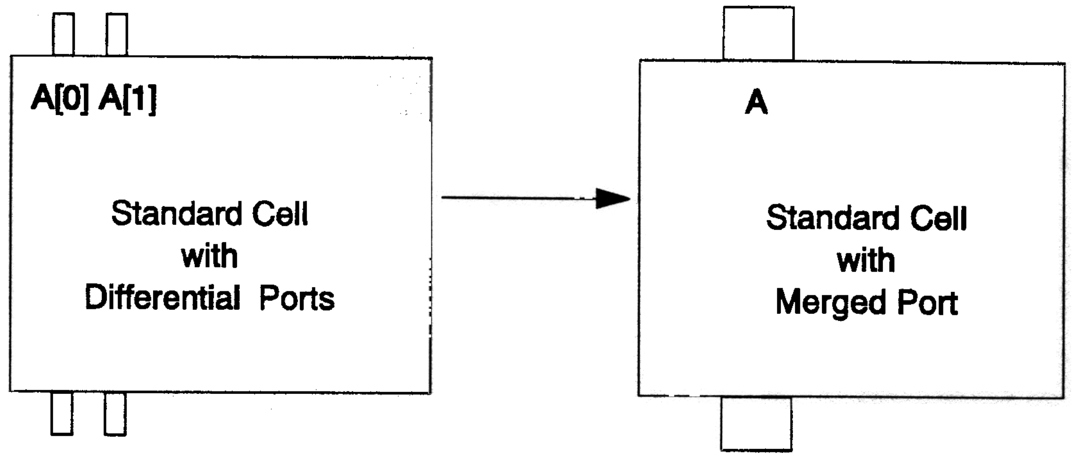
A[0] A[1]
Standard Cell with Differential Ports
A
Standard Cell with Merged Port

FIG. 3.4. MERGING OF STANDARD CELL PORTS

Stage II uses a commercial routing tool, designed for single ended logic, as the core router. A modified routing environment, consisting of standard cell images with single logical ports, a single logical entity net list, and a modified design rule file dictating

TABLE 3.1.

NET LIST COLLAPSING MAPPING TABLE

Pin	Signal Pair	Pin	Logical Net	Inversion
101 102	Interrupt[0] Interrupt[1]	101	Interrupt	No
105 106	Addr[1] Addr[0]	105	Addr	Yes

large dimension wires is constructed. The geometry of these wires encompasses two standard wires and the interpair spacing. Operating in this environment, the core router generates an output file of the chip routed in large geometry wires. Stage II of the architecture requires time equivalent to that necessary to route a single ended design of equivalent logic complexity.

Stage III is a post processing phase. The large geometry wires are bifurcated, the standard cell images are replaced by the real standard cells, and the inversion information is re-introduced. The result is a chip with adjacent track routing of all differential nets. The bifurcation and re-introduction of inversions, proved the true intellectual challenge. For a given net, the worst case time complexity of the generalized algorithm is exponential as a function of decision points, or vias for that net. To split all nets the upper bound is given by equation 3-1, where *n* represents the total number of nets, and *v* stands for the number of vias in net *i*. The results of my research have reduced this algorithm to a linear time one.

$$Bifurcation\_Size = \sum_{i=1}^{n} 2^{v_i} \qquad (3\text{-}1)$$

Each phase of the process will be examined in more detail, but a short example of bifurcation is appropriate at this time.

3.2.2 Intuition of the Bifurcation Operation

To fully appreciate the bifurcation operation, an example is necessary. But before proceeding with the example, a further look at splitting single logical entity (SLE) vias is in order. When an SLE via is partitioned, there are two possibilities. Fig. 3.5 provides a visual illustration of these two possibilities. Fig. 3.5(a) shows the initial SLE via. Fig. 3.5(b) shows the northwest-southeast (NW-SE) splitting, while Fig. 3.5(c) shows the northeast-southwest (NE-SW) splitting option.

From this example it is apparent that each SLE via actually introduces a degree of freedom into the net splitting operation. With this understanding it is now time to demonstrate a complete net bifurcation.

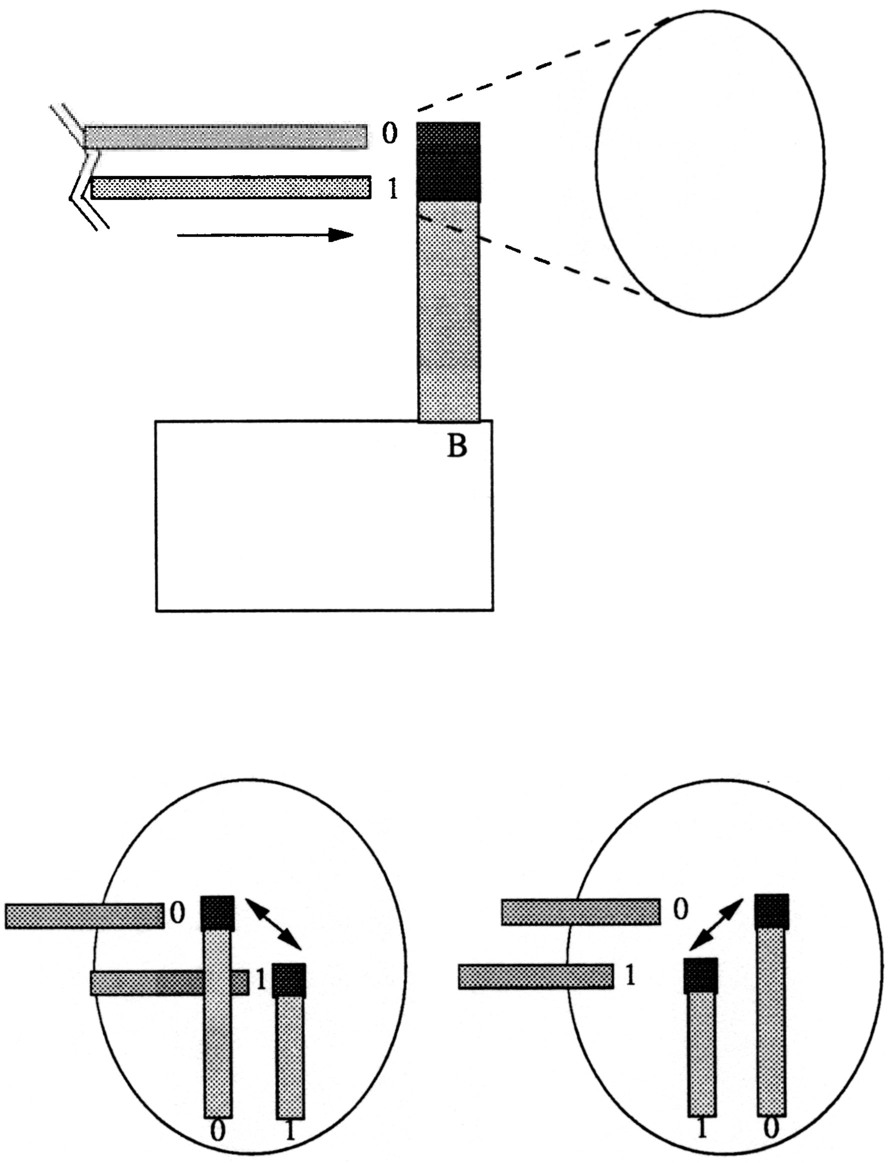

FIG. 3.5 SLE VIA SPLITTING OPTIONS

Fig. 3.6 provides a clear demonstration. Given the single logical entity net of Fig. 3.6(a), begin at the instance location on the left. Split the port into its original two components, split the oversized geometry wire into two standard sized wires, and propagate the respective instance polarities along the two segments, Fig. 3.6(b). At the first via junction, in the absence of any other constraints, either via split option shown in

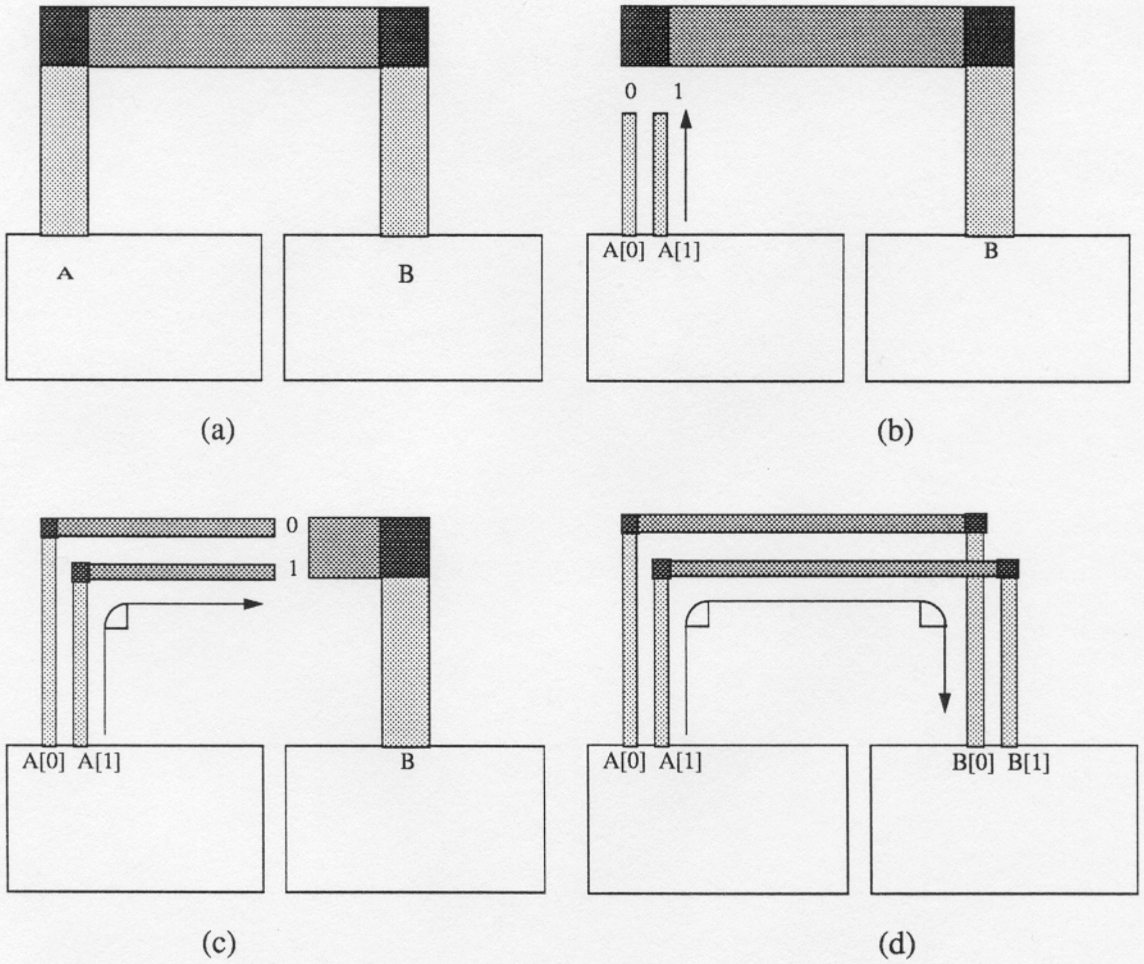

FIG. 3.6. SAMPLE SLE NET BIFURCATION

Fig. 3.5 is permissible. The NW-SE split is selected. The horizontal segment is now split and the polarities pushed forward, Fig. 3.6(c). At the next via junction, an incoming port segment provides a constraint. The combination of the polarities being pushed forward along the horizontal segment, and the constraint imposed by the polarities emanating from the port segment permit only one valid choice for the via split. The via split corresponding to Fig. 3.5(b) produces the final differential pair shown in Fig. 3.6(d). With this intuitive understanding of the bifurcation operation, each stage of the routing process will be examined in detail.

3.3 PROCESS ANALYSIS

The layout of the overall process was shown in Fig 3.3. Apart from pure complexity issues, each stage of the process presented unique problems and provided interesting options. As each stage is discussed, the challenges and the opportunities will be highlighted.

3.3.1 STAGE I (PREPROCESSING)

Three major functions occur during Stage I. First, the differential net list must be translated into an equivalent, single ended net list. Next, the standard cell library designed to support the differential technology required processing so as to produce a phantom version for each standard cell. These phantom cells had single logical ports rather than the actual differential pair. The third task was extracting virtual inversion information.

3.3.1.1 Net List Translation

Table 3.1 demonstrates the concept. The actual implementation is somewhat more complex. The complexity arises from the multiple signal formats involved in differential systems. Power, ground and other special signals, appear as single ended signals and must be treated in the traditional way. Then there are regular signals, which are single logical nets that must be handled as a differential pair. Finally, there are busses. They come in various sizes, and signal orderings, but must be collapsed in proper sequence.

Since the specific implementation details of this portion of the system are individualized for the COMPASS design tool suite, they are provided as a separate maintenance document. At this time only the major steps are reviewed. When scanning the net list, the signal type must be recognized. If the signal is a special one, or a power or ground line, then it is passed to the output file directly without modification. For signals that are logically single, the subscript notation used to identify the wires of the pair must be modified. In the example of Table 3.1, the subscripts for such wires are eliminated altogether when the output file is written. For busses, the level of sophistication in dealing with the subscripts increases. Some form of modified numbering scheme must be generated. As the lines of the bus are encountered, the subscript transform function is applied. This generates the signal/subscript combination that is passed to the output file.

Next, the net list typically provides some signal connection information with respect to a model of each type of standard cell used in the circuit. This information includes the labeling of the instances, and some physical ordering information. Usually, this is abstracted to a high level representation so that actual coordinate information is only of importance to the screen, and layout generation routines. The labels and orderings for the differential pairs must be collapsed in a manner that parallels that used for the signals. Maintaining strict consistency is very important.

3.3.1.2 Inversion Extraction

Although a separate task, inversion extraction will, in all likelihood, occur as part of the net list translation operation. During the net list scanning and translation process some database type cataloguing must take place. As signals appear for the first time, the ordering of the primary signal of the pair and its compliment is noted. Whenever another instance of the signal is encountered during net list processing, the ordering is checked against the database. If the orderings have switched, an inversion of the wire between the nodes has occurred. This information is appended to the database. When the template lines for the cells is encountered, the pair orderings are again assessed. The database information is compared with the physical cell instance orderings and a final decision on inversion is calculated. At the conclusion of the net list translation phase, the inversion list is output to a separate file. This file is critical input to Stage III. As each logical net undergoes bifurcation, the inversion file is checked, and the appropriate wire crossing accomplished at each port-via connection.

3.3.1.3 Standard Cell Modification

The third thing which must be accomplished during Stage I is the creation of pseudo cells, or phantoms. The differential logic standard cell library contains cells that have port pairs for each input and output signal. The phantoms are created by generating an image of each cell. These cell images consist of only the outlines, and have the port pairs collapsed into fat wire ports. The particulars of the task are intimately linked with the original geometry decisions for the technology. Once the technology is selected, designers must calculate the desired metalization geometry and the wiring pitch. For ordinary single ended designs, the geometry and wiring pitch are chosen based on the minimum feature size that can be fabricated. For differential routing, a more in depth analysis is necessary. Chapter nine of the dissertation deals with the electro-magnetic issues associated with determining individual wire and pair spacing so as to optimize the trade off of capacitive effects against chip area increases.

All of these decisions are important, because the choice of wire geometry and pitch affect the original standard cell design. Cell sizes grow in fixed amounts, analogous to quantum levels for electrons. If the devices and interconnect within a given cell cannot be contained within a given boundary, the cell size must increase in the horizontal direction by a multiple of the large geometry wiring pitch. This is necessary so that when port pair connectors are collapsed, their centerline falls on a large wire routing grid. This is explained in more detail in chapter nine, where the electro-magnetic effects of differential pair spacing is analyzed.

As a consequence, during the collapsing of the port pairs, the pseudo-port coordinates must agree with the large wire routing grid. This is a necessary condition for the router operation in Stage II. Additionally, as the collapsing process occurs, the ordering information for each pair is recorded.

For future differential projects, a standard ordering pattern on ports would simplify the final bifurcation operation. In an ideal world, each cell type would have a version with all possible pair orderings. But the time required to generate, and the space needed to hold this quantity of information, makes such an approach unreasonable. A solution which overcomes both the time and space constraints, and also insures all nets fall in the bifurcatable category will be presented in chapter eleven.

3.3.2 Stage II (Routing)

During Stage II the actual routing problem is solved. The position of core router, shown in Fig. 3.3, can be filled by any single ended router. Obviously, a well thought out choice will provide final differential results, whose optimality is equivalent to the core router operating on a single ended system. If one desires to minimize vias, then a core router that minimizes vias should be chosen. If total interconnect is to be minimized, then a core router with this characteristic can be selected.

The router operates in a scaffolded environment. The standard cell library that it sees is the image of the cell library constructed in Stage I. The net list, on which it is to operate, is the modified version that mimics a single ended one. Even its technology data base is camouflaged. Two parameters are adjusted to meet the overall design requirements. The wire geometry for each metalization layer is the first parameter selected. It is chosen, such that the fat wire width encompasses the width of the two small (normal size) differential wires, and the necessary inter-pair spacing. Fig. 3.7 illustrates this fact. The geometry of the large wire on the right matches the dimensions of the two normal wires plus their relative spacing. For the F-RISC/G platform, the standard wire geometry's were 2µm for metal-1 and 3µm for metal-2.

FIG. 3.7. FAT WIRE DIMENSIONS

The other parameter is the wiring pitch. Rather than use the normal pitch, a modified pitch is selected to properly space the large wires. The minimum spacing used was 3µm. Future projects using differential wires should use the methodology outlined in chapter nine to arrive at a proper spacing.

Operating with this scaffolding, the core router produces the fat wire solution. The run time consumed is equal to that used for a single ended problem of similar logic complexity. As an aside, it is important to note that had the logical net insight not been employed, the run time would have doubled at a minimum. The one restriction imposed on the router is to follow the Manhattan constraint. Horizontal segments must be routed in one layer of metalization and the vertical segments routed in another[Ohts86]. The rationale for imposing this constraint will be discussed later. At this point, if suffices to say that by adhering to the Manhattan constraint, sufficient degrees of freedom exist in the large wire solution to correctly bifurcate the nets in the taxonomy presented in chapter four.

3.3.3 STAGE III (BIFURCATION)

As was eluded to earlier, the bifurcation stage is the one that provided a real intellectual challenge. During this phase, three things had to be accomplished. The large wires had to be resolved into their respective pairs. Inversion information extracted during Stage I, had to be reviewed and re injected into the solution. And finally, at the block level of routing, the last available degree of freedom for polarity is at the pads. With both polarity versions available for each pad, the bifurcation function must select the appropriate pad to properly finish each net.

Equation 3-1 presented the computational complexity involved with bifurcation. Unless this can be reduced from exponential to something more tractable, the viability of the overall approach is in jeopardy.

3.4 SUMMARY

Other approaches to the problem certainly exist. It was not my intent to enumerate all of them, rather only to mention two that were representative of incremental thinking. Each is certainly a potential candidate. They attempt to reduce the number of nets that have to be managed. But to proceed along traditional lines, the solution generated will always be inferior to the one generated by the modified router architecture presented in this chapter. The next three chapters concentrate on a solution to the bifurcation problem. It evolves through a series of steps, which ultimately lead to a linear time solution.

Chapter four recounts the evolution of feature vectors, from a primitive recognition vehicle, to a powerful tool critical to the final solution. Chapter five presents a theoretical proof of correctness for the feature vectors, through the use of finite state machine theory. Chapter 6 develops the complete bifurcation problem, and presents the ideal, linear time solution that resulted from this research.

CHAPTER 4

Managing Differential Signal Placement

FEATURE VECTOR EVOLUTION & NET TAXONOMY

4.0 GENERAL THOUGHTS

Although the basic concept behind bifurcation is easy to comprehend and almost intuitive, the details of implementation prove quite complex. As is the case with most difficult problems, the solution to bifurcation was not immediately apparent. Instead, the solution to the bifurcation problem evolved over time. This evolution progressed through several stages. The first of these was the feature vector. As the feature vector concept began to solidify, a full net taxonomy started to take shape. This taxonomy provided the mechanism which triggered the development of the finite state machine theory of chapter five. It was the finite state machine theory that would ultimately provide the theoretical underpinnings of the research. Ultimately, the various components of the solution were shown to be linked through the use of regular expressions.

This chapter deals with the development of the feature vectors, from a tentative recognition mechanism, to their final form upon which the linear time complexity of the overall bifurcation algorithm is based. As the feature vectors began to evolve a taxonomy of nets possessing the characteristic of being properly bifurcated developed. This taxonomy helped demonstrate a completeness of the final solution, and a tool with which to explore the extendibility of the idea into n-layer metalization in chapter seven.

4.1 FEATURE VECTOR CONCEPT

The general idea of a feature vector has been around for many years. Its beginnings are rooted in the area of pattern recognition and machine vision[Scha89]. In this domain, the input information typically has a certain degree of uncertainty associated with it. Either the acquired image is not clear, the resolution does not provide sufficient detail, or natural/man-made obstructions exist that obscure portions of the object under investigation. In the presence of such uncertainty, probability theory has to be incorporated. Feature vectors provide a mechanism for organizing and applying the probability theory.

For each object that a system must recognize, a feature vector is defined. The vector is made up of components which represent certain features or characteristics of the object necessary for recognition. As image analysis algorithms process the input, these characteristics are identified to various degrees. The relative certainty of the feature under consideration is then quantified and assigned to each vector as a percentage.

After processing the input information, the set of feature vectors is examined to see which, if any, provides a match. Typically, with incomplete information, several of the vectors will provide partial matches. Weights can then be assigned to each vector component to indicate their relative importance to the final understanding of the object. It then becomes an exercise in probability theory to identify which among the competing vectors identifies the object with the highest degree of certainty. The theory is solid, but the combinatorics involved usually cause it to become a computationally intensive approach.

Bringing this theory to bear as a means of recognizing net topologies was not immediately apparent. The remainder of this chapter reveals how this particular idea evolved, and produced a taxonomy of readily bifurcatable nets.

4.2 MANIPULATING "CP" FILES

The viability of the entire architecture hinged on the key premise: "routing" of the complimentary signal of differential pair could be handled by splitting a fat wire. To accomplish this, the splitting operation relied on being able to apply the proper modifications to an arbitrary intermediate file format. For this investigation, the *Compose Editor* (cp) file for VLSI Tools was used. At the outset of the research, uncertainty existed about the possibility of being able to successfully manipulate this file.

4.2.1 THE SIMPLE CASE

When initially defined, the bifurcation problem seemed so large and complex, doubt existed about whether a solution could actually be constructed. Consequently, in developing the solution, divide and conquer was the obvious strategy of choice.

The first step was to confirm the feasibility of introducing modifications into the *cp* file, and having the design system understand them. It turns out that in its basic form, the cp file format is well suited for the intended operations. Fig. 4.1 provides an extract from a cp file. Table 4.1 focuses on the net components of a cp file and their explanation. To keep the modifications manageable a trivial net was identified for a sample splitting.

By analyzing the components, the topology of a net can be discerned. For the example in Table 4.1, the net defined is portrayed in Fig. 4.2. It has a a single backbone segment, two vias, and two metal-1 segments connecting those vias to standard cell ports.

Heading	V 5 VLSIcompose
	A -60 0 6900 4716
	B -61 -1 6901 4717
	F F
Connectors	C * 6900 1993 Metal2 30 1 * * E OUT[0]
	C * 6900 1933 Metal2 30 2 * * E OUT[1]
	C * -60 1993 Metal2 30 3 * * W OUT[2]
	C * -60 1933 Metal2 30 4 * * W OUT[3]
	C * -60 101 Metal2 30 5 * * W OUT[4]
	C * -60 41 Metal2 30 6 * * W OUT[5]
Instances	I U9$9 120 142 0 "ant21m" scp * F
	I U9$8 720 142 0 "ant21m" scp * F
	I U9$7 1320 142 0 "ant21m" scp * F
	I U9$5 1920 142 0 "ant21m" scp * F
	I U9$6 2520 142 0 "ant21m" scp * F
	I U9$2 3120 142 4 "ant21m" scp * F
Net Section	D 4470 2713 * CW1W2 0 20 20
	D 5250 2713 * CW1W2 0 20 20
	W 4455 1771 5265 2855 U8.A3
	S 20 V Metal P 7 I U15 14
	S 30 H Metal2 P 7 P 8
	S 20 V Metal I U8 10 P 8
	D 3390 2593 * CW1W2 0 20 20
	D 3750 2593 * CW1W2 0 20 20
	D 6210 2593 * CW1W2 0 20 20
	W 3375 2518 6225 2855 U1.A12
	S 20 V Metal P 64 I U12 14
	S 30 H Metal2 P 63 P 64
	S 20 V Metal P 63 I U13 7
	S 30 H Metal2 P 64 P 65
	S 20 V Metal P 65 I U1 7
Postscript	A H I g1 19 I U9$9 18 * *
	A H I g1 20 I U9$9 17 * *
	A H I U9$9 19 I U9$8 18 * *
End	E

FIG. 4.1 *CP* FILE EXTRACT

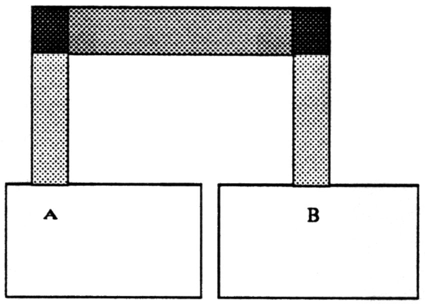

FIG. 4.2 NET DESCRIBED BY *CP* FILE

As a first attempt to split the net, a straight forward duplication of components was performed. All of the entries were duplicated and modifications applied. The wire segment and via dimensions were adjusted appropriately. To have the proper segments continue to connect with the vias, the via coordinates were offset to match the revised segment center lines. The modified segment section is shown in Table 4.2. With these changes in place, the system was asked to re-display the routing solution.

TABLE 4.1

CP FILE NET SECTION

Notes:	**CP File Example**
D lines indicate vias with coordinates and dimensions.	D 4140 2000 * CW1W2 0 80 80 D 3600 2000 * CW1W2 0 80 80
W line gives net bounding box.	W 3500 1900 4200 3000 Interrupt
S lines represent wire segments going from cell instances to pins (vias), and between pins.	S 80 V Metal I instance_name Pin_no P Pin_no S 80 V Metal I instance_name Pin_no P Pin_no S 90 H Metal P Pin_no P Pin_no

TABLE 4.2

NET CUT SAMPLE

Notes:	Cut CP File Example
Via coordinates adjusted to offset points.	D 4110 1970 * CW1W2 0 80 80
	D 3570 1970 * CW1W2 0 80 80
	W 3500 1900 4200 3000 Interrupt
Pin numbers for first wire of net remain constant.	S 80 V Metal I Interrupt 10 P 1
	S 80 V Metal I Interrupt 11 P 2
	S 90 H Metal P 1 P 2
Complimentary wire of net.	D 4170 2030 * CW1W2 0 80 80
	D 3630 2030 * CW1W2 0 80 80
W line duplicated.	W 3500 1900 4200 3000 Interrupt
Pin numbers offset to correspond to D line additions.	S 80 V Metal I Interrupt 12 P 3
	S 80 V Metal I Interrupt 13 P 4
	S 90 H Metal P 3 P 4

A partially correct solution was presented. The errors generated manifested themselves in subsequent nets of the routing solution where changes had not been made. An examination of the particulars identified the subtle difficulty. Pin numbers in the cp file have several dependencies. They correspond in a one to one fashion with the D-lines in the file. Also, within a net definition, they can occur in any sequence, but still refer back to the appropriate D-line of that net definition. By introducing an additional net, there were now