Flowgrapher - Generation of Conceptual Graphs from Flowcharts

by

Ramprasad Venkatasubramanian

Project report submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

APPROVED:

[illegible]

Dr. W. R. Cyre, Chairman

[illegible]

Dr. J. R. Armstrong

[illegible]

Dr. F. G. Gray

June 1994

Blacksburg, Virginia

c.2

LD
5655
V851
1994
V465
c.2

Flowgrapher - Generation of Conceptual Graphs from Flowcharts

by

Ramprasad Venkat

Dr. W. R. Cyre, Chairman

Electrical Engineering

(ABSTRACT)

This project is part of the Automatic Specification Interpreter (ASPIN), a tool currently under development, that takes the various forms of informal specifications and creates formal engineering models from them. ASPIN has a long-term goal of providing an automated system for digital system synthesis from informal specifications. This report describes an algorithm and a program, *flowgrapher*, that generates conceptual graphs from requirements expressed as flowcharts captured with a commercial tool.

Generation of conceptual graphs from flowcharts is done in two stages. First the flowchart is drawn using Viewlogic's Viewdraw and a netlist representation of the flowchart is obtained. Next, this netlist is parsed to obtain the successor/predecessor relations between the various flowchart blocks and these relations are used to develop the conceptual graph representation of the flowchart. *flowgrapher* has been implemented in the C programming language under the Unix environment.

To

Appa

Acknowledgements

My sincere thanks to my advisor Dr. Walling Cyre for the support and guidance throughout this research and for the patience he displayed during trying times. I would also like to thank Dr. J. R. Armstrong and Dr. F. G. Gray for serving as members of my committee and for their helpful suggestions.

A special word of thanks to Mr. S. Shankaranarayanan for his helpful programming hints during the course of this project. And last but not the least, I wish to thank my wife, Tessy for her loving support through all my efforts.

Table of Contents

List of Illustrations

Chapter 1. Introduction

1.1. The ASPIN System

The Automatic Specification Interpreter (ASPIN), being developed at Virginia Tech is a tool that accepts various forms of informal specifications and creates formal engineering models from them [1]. Simulation and further synthesis can be performed from these models. The target formal model of the system is the VHSIC hardware description language (VHDL), which is becoming a standard language used in the industry for modeling and synthesis. ASPIN uses *flowgrapher,* the software tool described in this report, to generate conceptual graphs from specifications given in the form of flow charts.

System specifications can be in various forms such as block diagrams, timing diagrams, flow charts and natural language. The various forms of specifications are entered into the ASPIN system and mapped into a common knowledge representation called conceptual graphs. The natural language input is processed by the parser and semantic analyzer in order to generate the conceptual graphs. A detailed overview of the conceptual graph knowledge representation is provided in Chapter 2. The resulting conceptual graphs are then processed and used to synthesize VHDL models simulating the behavior and characteristics of the system. They are also used to create a graphical representation of the design information. Synthesis of conceptual graphs into VHDL models is done by a program called the LINKER. Figure 1.1 shows an overview of the ASPIN system.

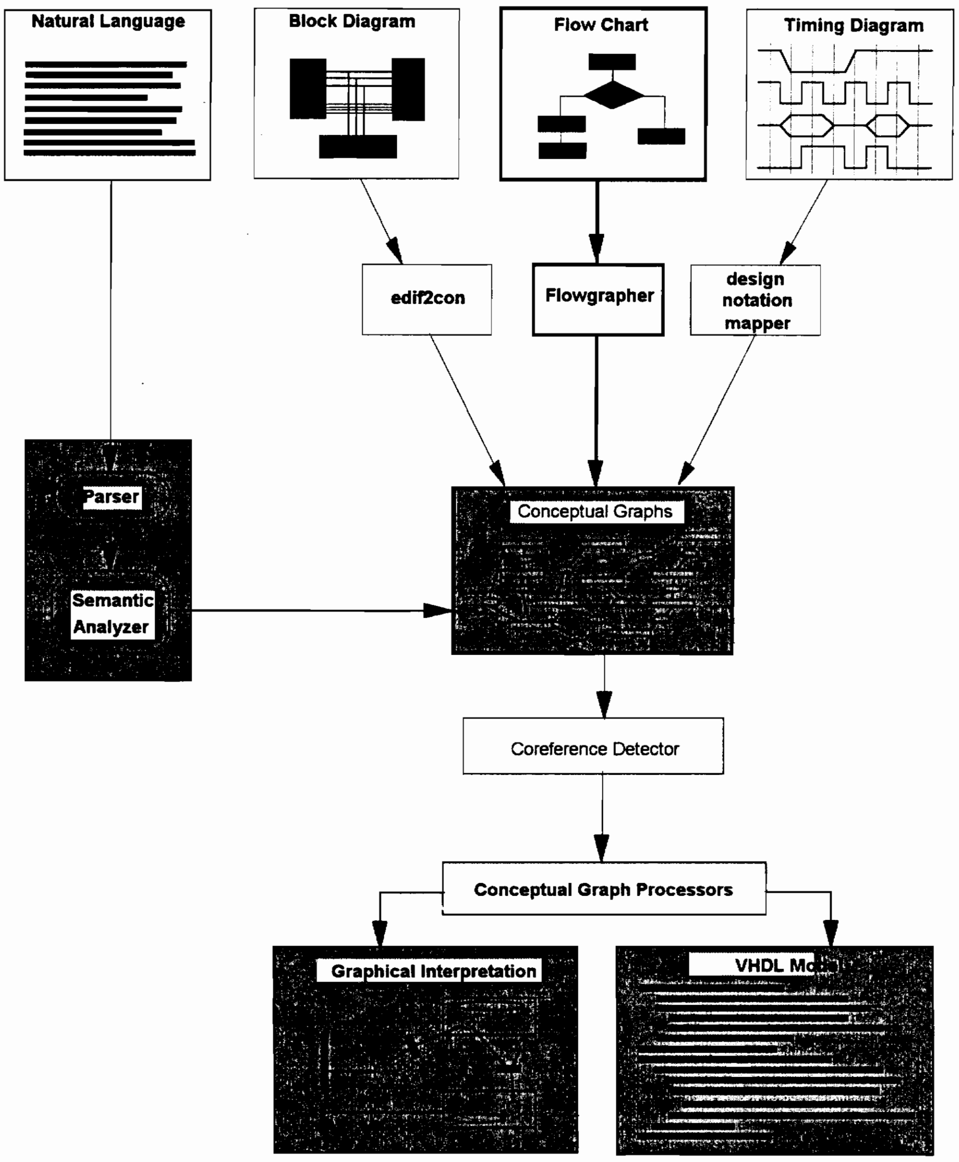


Figure 1.1. ASPIN system overview

Shown in Figure 1.2 is the organization of the ASPIN system. Digital system requirements and specifications are entered by the specifications author. This information is processed and transformed into conceptual graphs. These conceptual graphs are then processed by the Linker and the Model Generator. The Model Generator creates a graphical representation of the conceptual graph that is fed back to the specifications author for verification. The Linker transforms the conceptual graph into a form that can be used by the Modeler's Assistant to create the VHDL models. Once the VHDL models are constructed using the Modeler's Assistant, the designer can continue system development by performing behavioral simulations for further design verification. The VHDL models can also be used to perform further synthesis into low level system models or hardware.

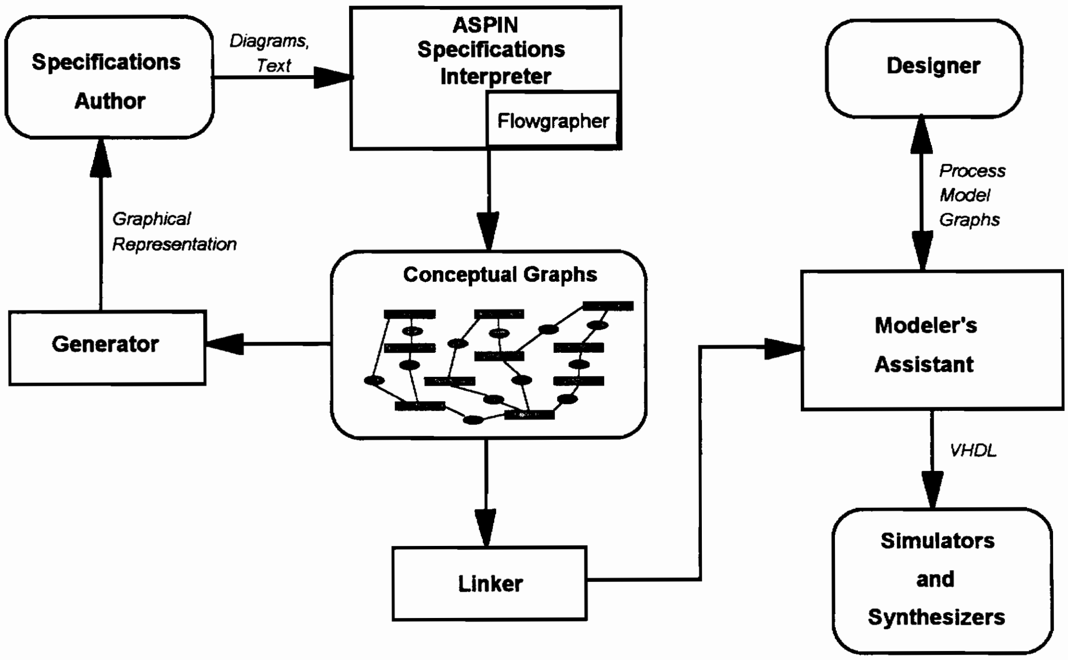


Figure 1.2. ASPIN system organization

1.2. Flowgrapher - An overview

The Flowgrapher is a software tool developed to generate conceptual graphs from system specifications given in the form of flow charts. The conceptual graphs generated by the flowgrapher serve as input to the Linker and the Model Generator. The generation of conceptual graphs from flowcharts is done in two stages: 1) netlist generation and 2) netlist parsing and conceptual graph generation.

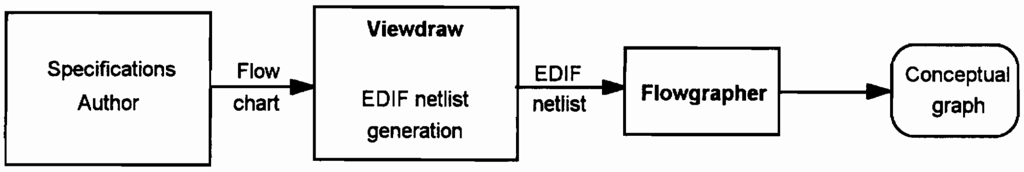


Figure 1.3. Block diagram of *flowgrapher*

The flowchart is first drawn in Viewlogic Viewdraw[1] using a library of symbols created for this purpose. A netlist describing the incidences of the flowlines with the various flow chart blocks is then obtained. This netlist is in the Electronic Data Interchange Format or EDIF, a widely used format for the interchange of data between electronic design systems [5]. The EDIF netlist is then input to the flowgrapher which parses it. The flowlines in the flowchart are obtained as successor/predecessor relations by parsing the netlist. These relations are used to generate the conceptual graph that represents the flow chart.

[1]Viewdraw is a registered trademark of Viewlogic Systems Inc.

1.3. Report Organization

This report is presented in four chapters. Chapter 1 provides an overview of the work done in this project along with related background information. Chapter 2 reviews the conceptual graph formalism which is the target of the work described in this report. An overview of flow charts, basic flow chart blocks, their conceptual graph representation, and the role of flow charts in design specifications is also presented in this chapter. Chapter 3 describes the Electronic Data Interchange Format (EDIF) and how it is used in this project. Also presented in this chapter is an example of a flow chart and its EDIF netlist. Chapter 4 describes the organization of the flowgrapher written in the C programming language. It also provides the rules and restrictions imposed on the user in the process of generating a conceptual graph from a flow chart.

The report has three appendices. Appendix A contains the User Manual of the flowgrapher. Appendix B contains a list of the library blocks created as a part of this project to draw a flowchart in Viewdraw, along with their conceptual graph representations. Appendix C lists a few examples of flowcharts, their EDIF netlists and their conceptual graph representations.

Chapter 2. Related Research

2.1 Flowcharts

A flow chart is a schematic description of an algorithm or a process. It represents the sequence of events in a process along with the transfer of control flow within the process.

Flowcharts are useful at all stages of designing a system. Initially a designer may express the overall formulation of the requirements of a system by means of a rudimentary flow chart. Subsequently, this flow chart can be refined or altered as the attention of the designer focuses on specific details. Verification of the effectiveness and correctness of the design can then be accomplished by tracing all possible alternatives displayed in the flow chart. Flowcharts serve as an efficient means of communicating design specifications and requirements among designers. They are also a very effective way of documenting design specifications. It may be easier for a designer to understand the design methodology of a system by referring to its flow chart than to the actual system itself [2].

2.1.1. Standard Flowcharting Symbols

Of all the standard flowcharting symbols, a subset that is frequently used by designers is dicussed here. Also shown is a discussion about flow lines that are used in a flowchart to denote control flow. Each one of the symbols corresponds to a basic function. These symbols are referred to as *blocks* in this report. An illustration of the flowlines and the basic flowchart blocks followed by a brief description of their sense is given below.

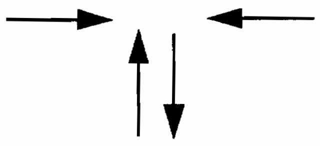


Figure 2.1 : Flow lines

Figure 2.1 shows the representation of flow lines in a flowchart. Flow lines in a flowchart connect all blocks to display the sequence in which operations are to be performed. They denote casual dependencies or control flow rather than information paths.

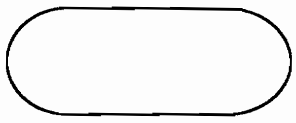


Figure 2.2 : Terminal (or connector) block

Figure 2.2 shows the terminal (or connector) block. This block denotes the beginning or the end of a flowchart.

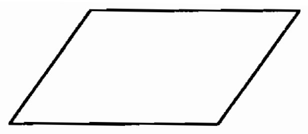


Figure 2.3 : Input / Output block

Shown in Figure 2.3 is the input / output block. This block indicates read or storage operations.

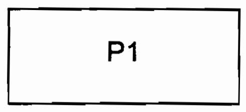


Figure 2.4 : Process block

Figure 2.4 shows the representation of the process block. The process described in this block could be a data transfer, evaluation of an expression or even execution of a block of code. A process in a flowchart can also be another flowchart containing all the basic standard blocks in itself.

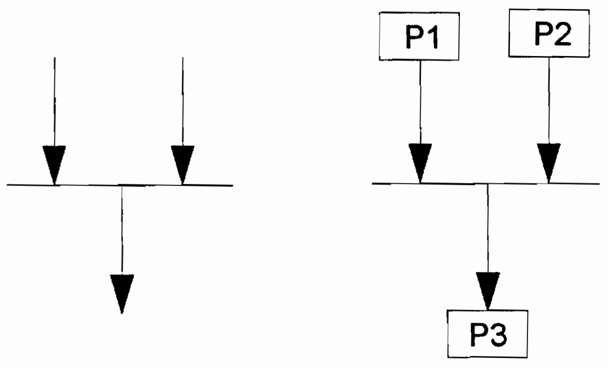


Figure 2.5 : Or-join block

Shown in Figure 2.5 is the representation of an or-join block and an instance of a simple flowchart with an or-join block. The or-join block indicates the transfer of control to the output flowline of the or-join from either of the two input flowlines. In the example, process P3 begins its execution either after process P1 completes its execution *or* after process P2 completes its execution.

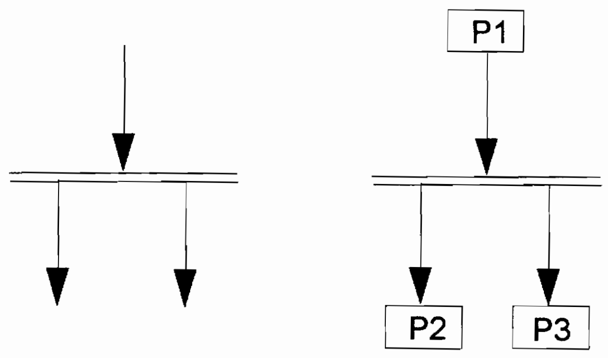


Figure 2.6 : And-fork block

Figure 2.6 shows the representation of an and-fork block and an instance of a simple flowchart with an and-fork block. The and-fork block indicates the beginning of simultaneous processes. In the example, both process P2 *and* process P3 begin their execution after process P1 completes its execution.

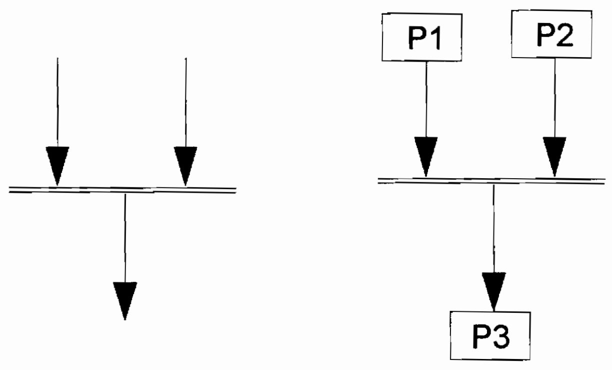


Figure 2.7 : And-join block

Shown in Figure 2.7 is the representation of an and-join block and an instance of a simple flowchart with an and-join block. The and-join block indicates the re-synchronization of simultaneous processes. In the example, process P3 begins its execution after both process P1 *and* process P2 complete their execution.

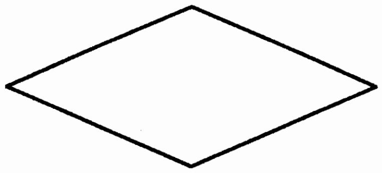


Figure 2.8 : Decision block

Shown in Figure 2.8 is the representation of the decision block. Decision blocks in a flowchart are responsible for transfer of control flow. A test is performed in the decision block and depending upon the outcome of the test, the control flow is transferred to one of the many control paths that arise from the decision block. Consider the simple flowchart shown in Figure 2.9. The result of the check process in the decision block labeled CHECK determines if the process labelled INCR or the process labelled PRINT should be executed following the completion of the process labelled READ. If the result of the test in the decision block CHECK is true (T) then process PRINT is executed after process READ. If the result is false (F) then process INCR is executed after process READ.

Decision blocks may have more than two outcomes depending upon the test conducted within them. A good example is a decision block checking the value on a signal. Possible values on a signal in a four valued logic system are '0', '1', 'X', and 'Z' where 'X' is the unknown state and 'Z' is the high impedance state. Thus the decision block can transfer control flow to one of four different paths depending upon the value on the signal.

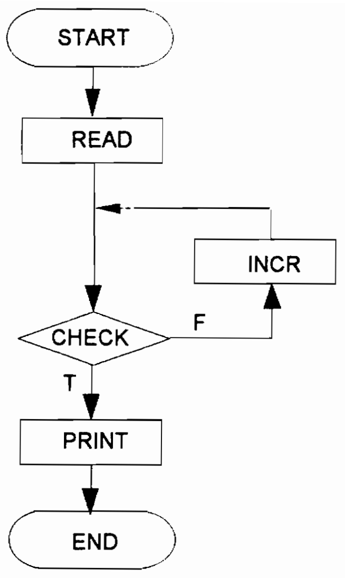


Figure 2.9. Flowchart containing a decision block

2.1.2. Symbols accepted by Flowgrapher

Flowgrapher accepts only a subset of the standard flowcharting symbols. The symbols described earlier are those that are commonly used by designers to describe system specifications. The termination or beginning of a flowchart is treated as a specialized (empty) process and is therefore represented by the symbol of a process block. Similarly, read/store operations are also treated as a specialized (empty) process and are also represented by the symbol of a process block. Apart from these, three different symbols have been created in this project. They are the symbols that represent the standard flowchart operations such as the OR-JOIN, the AND-FORK and the AND-JOIN. A brief discussion on these symbols and the reason for creating them is given below.

Viewdraw is a schematic capture tool that is used in this project to represent a flowchart whose conceptual graph representation is to be obtained and to generate its EDIF netlist. Viewdraw is primarily used to represent digital circuits for the purpose of simulation and to obtain netlists. The flowchart is drawn in Viewdraw as a *schematic* comprising of standard *symbols* located in the symbol library. The representation of the flowchart in Viewdraw has a few restrictions, many of which are imposed by the tool itself. For instance, symbol identifiers in Viewdraw must be between 1 and 255 characters long and may consist only of alphabetic characters, digits and underscores. This imposes a restriction on the label given to each block in the flowchart. Another restriction imposed by the tool, the result of which non-standard flowcharting symbols are used in this project, is explained below.

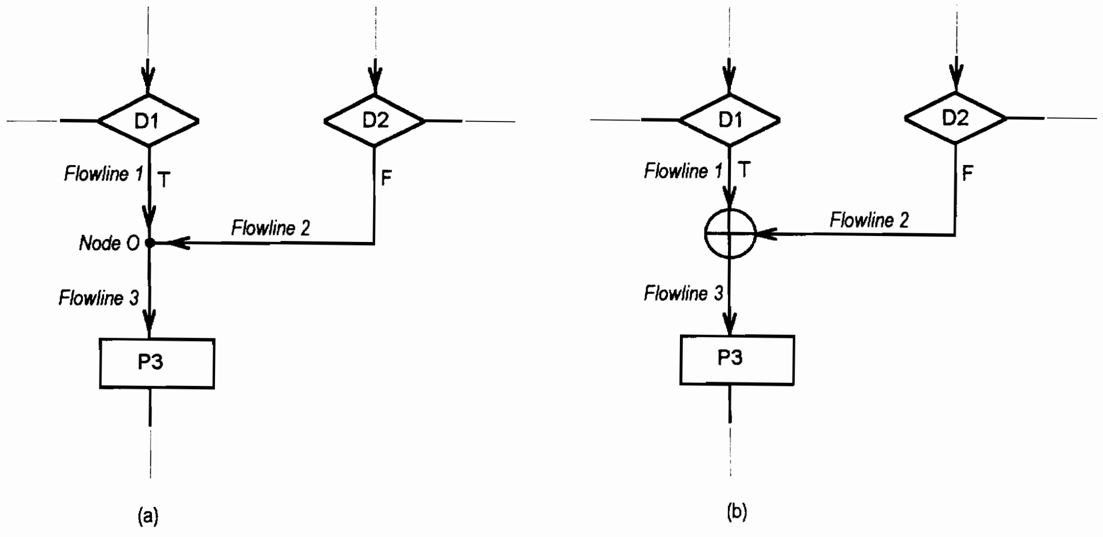


Figure 2.10 Restrictions in Viewdraw

In the flowchart shown in Figure 2.10 Process P3 is executed if the outcome of the test in decision block D1 is 'T' *or* if the outcome of the test in decision block D2 is 'F'. The results of the tests are attached as labels on the flow lines incident on decision blocks D1 and D2. In figure 2.7(a) flow lines 1 and 2 are connected to *Node O* which in turn is connected to *process* block P3 through flow line 3. Since Viewdraw is a schematic capture tool that is mainly used for representing digital circuits it treats all the three flow lines shown, as a single *net* that connects three different components and therefore does not allow assigning two different labels to the same net. In Figure 2.10(b) Node O is replaced by a non-standard flowcharting symbol, the *or-join* block. Thus the three flow lines are now treated as separate nets and therefore the outcome of the test can be attached as labels on the respective flowlines. The other non-standard flowcharting symbols such as the *and-join block* and the *and-fork block* shown below were also created for the same reason.

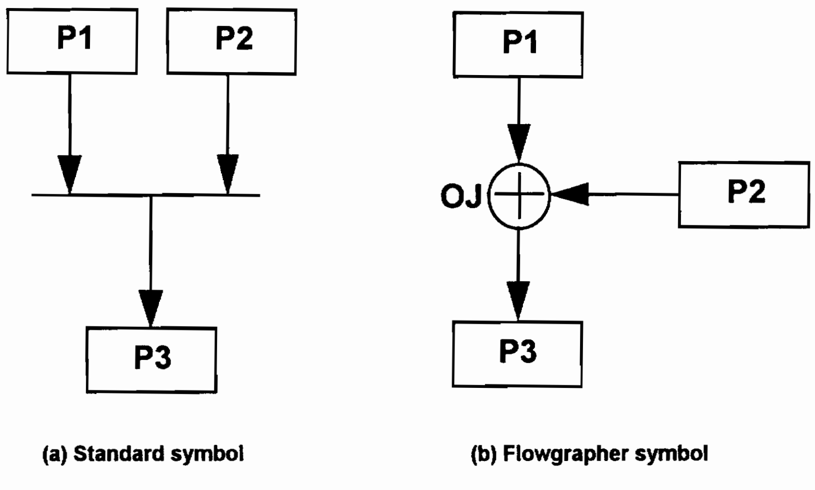


Figure 2.11. The OR-JOIN block

The OR-JOIN symbol OJ shown in Figure 2.11 denotes that process P3 begins execution after the completion of execution of either process P1 or process P2.

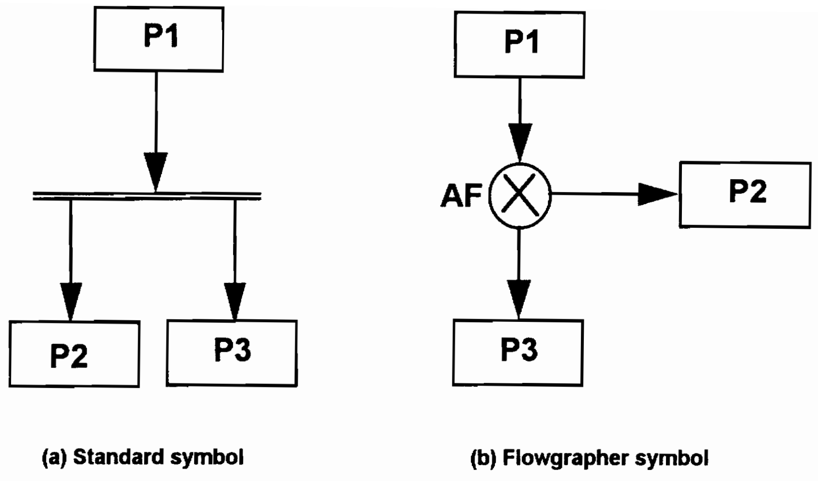


Figure 2.12. The AND-FORK block

The AND-FORK symbol shown in Figure 2.12 denotes the beginning of simultaneous processes. Both processes P2 and P3 begin their execution upon the completion of execution of process P1.

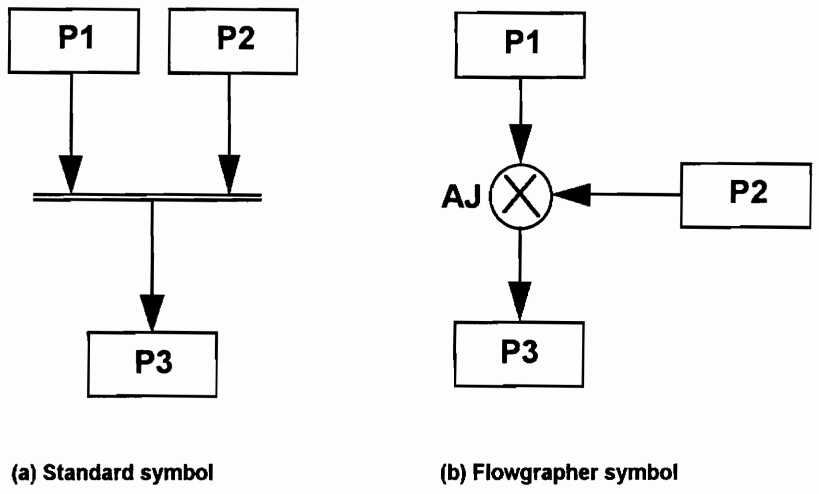


Figure 2.13 : The AND-JOIN block

Figure 2.13 shows the representation of the AND-JOIN block. It denotes the re-synchronization of simultaneous processes. Here process P3 begins its execution upon the completion of execution of both process P1 and process P2.

2.2. Conceptual Graphs

The target representation for the work described in this report is the system of *conceptual graphs* developed by John Sowa [5]. Conceptual graphs are a system of logic based on the existential graphs developed by Charles Sanders Pierce in the nineteenth century. The purpose of the conceptual graph system is to express meaning in a form that is logically precise, humanly readable, and computationally tractable. Conceptual graphs are more general than any of the popular design notations employed by engineers [2]. Since conceptual graphs directly map to natural language, they can serve as an intermediate language for translating computer oriented formalisms to and from natural languages.

By definition, a conceptual graph is a finite, connected, bipartite graph. It consists of two types of nodes. The first type is the concept node which can represent an object, action or a feature. The other type of node is the relation node that describes the relationship among concepts. A conceptual graph is finite because any graph representing some knowledge can have only a finite number of concepts and conceptual relations. Since every arc in a conceptual graph links a concept with a conceptual relation, a conceptual graph is bipartite. Any conceptual graph that is not connected is simply a group of individual conceptual graphs.

Conceptual graphs can be expressed in two ways - graphically and in linear text. In the graphical representation of a conceptual graph, the concepts are drawn as rectangular boxes and the conceptual relations are expressed as ellipses. The arcs linking the concepts and the conceptual relations are drawn as arrows. The concept nodes contain two labels specifying the type of the concept and the referent of the concept. The conceptual relation nodes also contain two labels which specify the role of the relation and its referent. Shown in Figure 2.14. is the graphical representation of a conceptual graph.

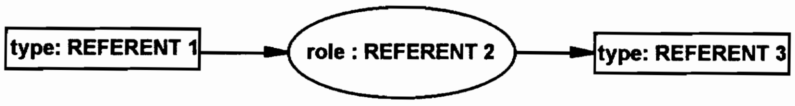


Figure 2.14. Graphical representation of a conceptual graph

In the linear representation of a conceptual graph, the concepts are written in square brackets. An additional field, called the concept number is used in this form to identify the concepts. The concept is written in the form
[*concept #* : *concept type* : *referent*]
The conceptual relations are written within parentheses. Arrows are used to represent the arcs in the conceptual graphs. Figure 2.15 shows the linear representation of the conceptual graph whose graphical representation is shown above.

```
[1 : type : REFERENT 1]
    -> (relation : REFERENT 2) -> [2]
[2 : type : REFERENT 3]
```

Figure 2.15. Linear representation of a conceptual graph

Conceptual graphs model knowledge by classifying concepts and defining relationships between concepts. Figure 2.16 shows a conceptual graph used to model the information contained in the sentence given below.

"The processor reads the data from memory"

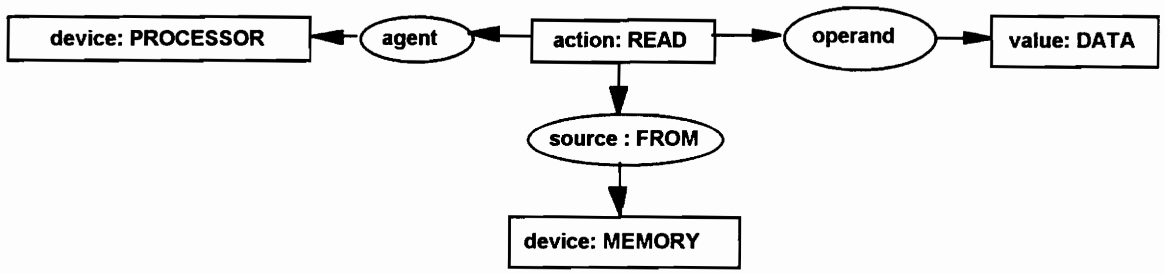


Figure 2.16 : Conceptual graph of the example sentence

In its linear form the graph is represented as shown below.

```
[1 : action : READ]
     ->(agent)-> [2]
     ->(operand)-> [3]
     ->(source : FROM)-> [4]
[2 : device : PROCESSOR]
[3 : value : DATA]
[4 : device : MEMORY]
```

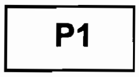
In the above example there are four concepts and three conceptual relations linking them. The processor, concept of type DEVICE is the agent of the read action. The data which is of concept type value is the object of the read action. Finally, the memory, also concept of type DEVICE acts as the source or location from which the data is read.

The concepts represented by the boxes in the above graph denote instances of the concept type specified. The referent label is used to distinguish individual instances from types. For example, in the linear representation of the conceptual graph, [2: device : PROCESSOR] indicates an instance of type processor. The word *PROCESSOR* is used as the referent for the instance of concept type *device* since it is the most specific marker that can be derived from the sentence. If no individual marker exists, an asterisk is used in the referent field. A concept containing an asterisk in the referent field is known as a *generic concept.*

2.3 Conceptual graph representation of flowchart blocks

The conceptual graph representations of several flowchart constructs are shown below. The types of concepts that are used in the conceptual graph used to model a flowchart are PROCESS and CONDITION type concepts. The concept type CONDITION is abbreviated as COND in the conceptual graphs. The process concept represents all the actions that occur in the process block of a flowchart. The conceptual relations that are used in the conceptual graph representation of a flowchart are *successor* (or *succ*), *predecessor* (or *pred*), *successor_if* (or *succ_if*), *predecessor_if* (or *pred_if*), *or,* and *and.* These relations are explained using an example of a flowchart and its conceptual graph representation as shown in figures 2.23 and 2.24.

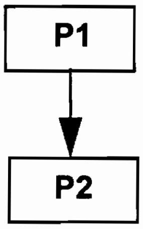
Process :



```
[ 1 : process : P1]
```

Figure 2.17. Conceptual graph representation - Process block

Flow line :

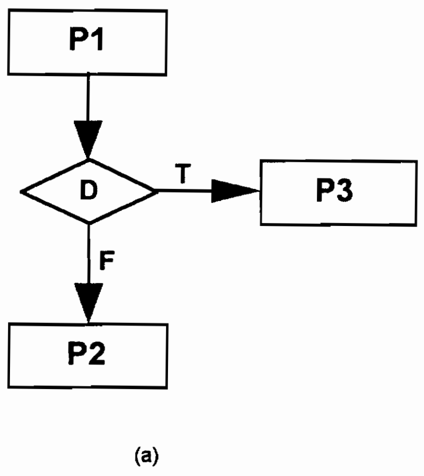


```
[ 1 : process : P1 ] -> (succ) -> [2]
[ 2 : process : P2 ] -> (pred) -> [1]
```

Figure 2.18. Conceptual graph representation - Flowline

The first statement can be explained as concept #1, of type process, named P1 has as its *successor*, concept #2, of type process, named P2. The second statement is interpreted as concept #2, of type process, named P2 has as its *predecessor*, concept #1, of type process, named P1.

Decision :



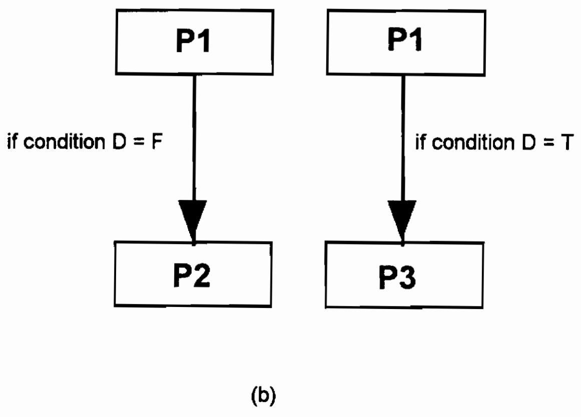


Figure 2.19. Conceptual graph representation - Decision block

```
[ 1 : process : P1 ]
  -> (succ_if) ->
     1->[ 2 : process : P2]
     2->[ cond : [D] -> (=) -> [F]]
  -> (succ_if) ->
     1->[ 3 : process : P3]
     2->[ cond : [D] -> (=) -> [T]]
[ 2 : process : P2 ]
  -> (pred_if) ->
     1->[ 1 : process : P1]
     2->[ cond : [D] -> (=) -> [F]]
[ 3 : process : P3 ]
  -> (pred_if) ->
     1->[ 1 : process : P1]
     2->[ cond : [D] -> (=) -> [T]]
```

Figure 2.19(a) above shows a decision block in a flowchart. The same can be represented as shown in Figure 2.19(b) by replacing ordinary flowlines with conditional flowlines. Thus we see that process P1 has process P2 as its *successor if* condition D evaluates to F, and process P1 has process P3 as its *successor if* condition D evaluates to T. The reverse relation can be explained as, process P2 has process P1 as its *predecessor if* condition D evaluates to F, and process P3 has process P1 as its *predecessor if* condition D evaluates to T. The same is represented in the conceptual graph shown above.

Or-Join :

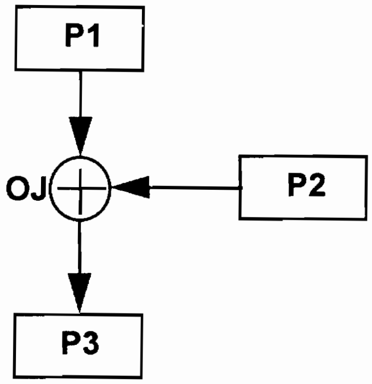


Figure 2.20. Conceptual graph representation - Or-join block

```
[ 1 : process : P3 ]
  -> (pred) -> [process] -> (or)
                1-> [ 2 : process : P1]
                2-> [ 3 : process : P2]
[ 2 : process : P1 ] -> (succ) -> [3]
[ 3 : process : P2 ] -> (succ) -> [3]
```

The above shown conceptual graph is interpreted as concept #1, of type process, named P3, has as its *predecessor*, a process, that is either process P1 *or* process P2.

And Fork :

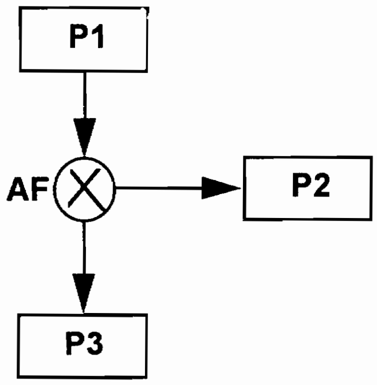


Figure 2.21. Conceptual graph representation - And-fork block

The *and-fork* denotes the beginning of concurrent processes. Here, both process P2 and process P3 begin their execution after process P1 completes its execution. The conceptual graph shown below is interpreted as, concept #1, of type process, named P1, has as its successors, concept #2, of type process, named P2 *and* concept #3, of type process, named P3. As reverse relations, both process P2 and process P3 have process P1 as their *predecessor*. Thus the *and* relation is not explicitly mentioned as in the case of the *or* relation, but is implied by the form of representation.

```
[ 1 : process : P1 ]
  -> (succ) -> [2]
  -> (succ) -> [3]
[ 2 : process : P2 ]
```

```
  -> (pred) -> [1]
[ 3 : process : P3 ]
  -> (pred) -> [1]
```

And-Join :

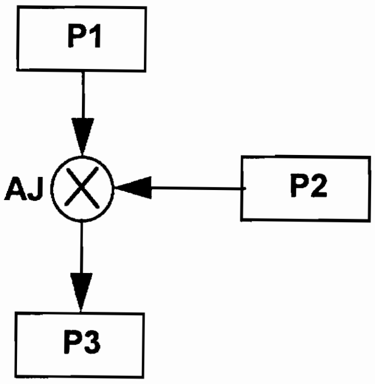
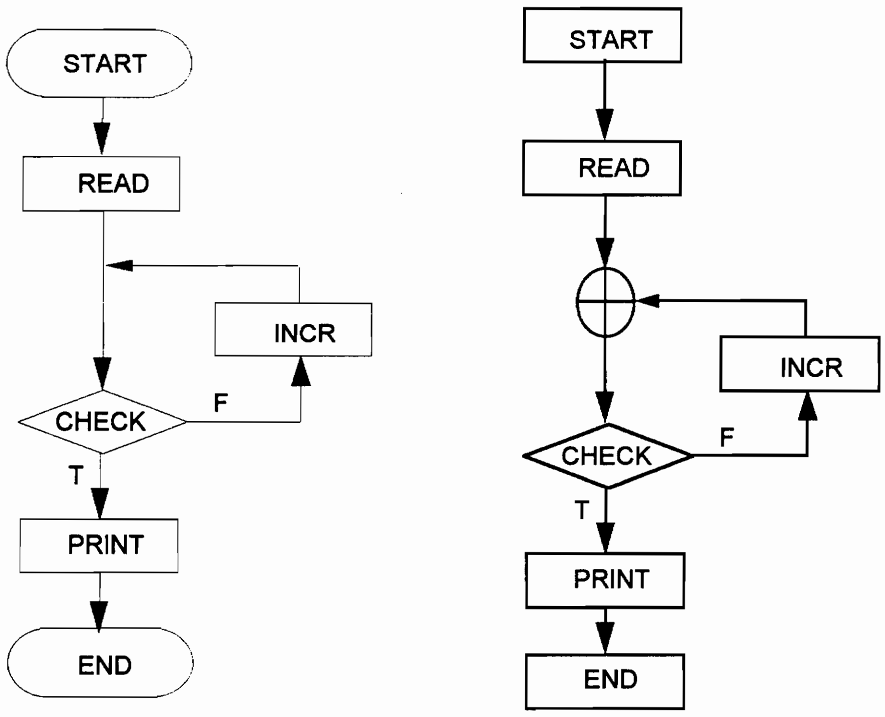


Figure 2.22. Conceptual graph representation - And-join block

The *and-join* denotes the resynchrcnization of concurrent processes. Here, process P3 begins its execution after process P1 *and* process P2 complete their execution. The conceptual graph shown is interpreted as, concept #3, of type process, named P3, has as its predecessors, concept #1, of type process, named P1 *and* concept #2, of type process, named P2. As reverse relations, both process P1 and process P2 have process P3 as their *successor.*

```
[ 1 : process : P1 ]
  -> (succ) -> [3]
[ 2 : process : P2 ]
  -> (succ) -> [3]
[ 3 : process : P3 ]
  -> (pred) -> [1]
  -> (pred) -> [2]
```

2.3.1 Conceptual graph representation of a flowchart



(a) Standard representation **(b) Flowgrapher representation**

Figure 2.23 : Example flowchart

The process blocks in the flow chart shown are START, READ, INCR and END. The or-join block in the flow chart denotes that the decision process in the CHECK block can be executed either after the READ process has completed its execution or after the INCR process has completed its execution.

The conceptual graph that models the above shown flowchart is shown in Figure 2.24.

```
[1 : process : START]
     ->(succ)-> [2]
[2 : process : READ]
     ->(pred)-> [1]
     ->(succ_if)->
          1-> [ 3 : process : INCR]
          2-> [ cond : [CHECK]->(=)->[F]]
     ->(succ_if)->
          1-> [4 : process : PRINT]
          2-> [ cond : [CHECK]->(=)->[T]]
[3 : process : INCR]
     ->(pred_if)->
          1->[1 : process]
               ->(or)->
                    1-> [3 : process : INCR]
                    2-> [2 : process : READ]
          2-> [ cond : [CHECK]->(=)->[F]]
     ->(succ_if)->
          1-> [4 : process : PRINT]
          2-> [ cond : [CHECK]->(=)->[T]]
     ->(succ_if)->
          1-> [ 3 : process : INCR]
          2-> [ cond : [CHECK]->(=)->[F]]
[4 : process : PRINT]
     ->(pred_if)->
          1->[1 : process]
               ->(or)->
                    1-> [ 3 : process : INCR]
                    2-> [ 2 : process : READ]
          2->[2 : cond : [CHECK]->(=)->[T]]
     ->(succ)-> [5]
[5 : process : END]
     ->(pred)-> [4]
```

Figure 2.24 : Conceptual graph representation of the flowchart shown in Figure 2.23

In the first statement of the conceptual graph that models the above flow chart, the START block is expressed as a concept of type process and is assigned a concept number 1. It has a *successor* relation with concept #2, the READ process. Using the direction on the arrows between the *successor* relation and the two concepts, we can interpret this statement as follows. *Concept #1, of type process, named START, has Concept #2 as its successor.* The reverse of this relation is represented in the second statement which can be interpreted as *Concept #2 of type process, named READ, has concept #1 as its predecessor.* The succ_if relation is used in the second statement to model the decision block.

```
[2 : process : READ]
        -> (succ_if) ->
                1-> [ 3 : process : INCR]
                2-> [ cond : [CHECK]->(=)->[F]]
        -> (succ_if) ->
                1-> [ 4 : process : PRINT]
                2-> [ cond : [CHECK]->(=)->[T]]
```

The above shown statement can be interpreted as follows. *Concept #2, of type process, named READ has process INCR as its successor if the condition CHECK = F is met. It has process PRINT as its successor if the condition CHECK = T is met.*

The or-join block is modeled in the next statement. We can see from the flowchart that if the outcome of the test conducted in the decision block CHECK is false (F) then process INCR is executed irrespective of whether the test was conducted after the execution of process READ or after the execution of process INCR. Thus *process INCR has either process READ or process INCR as its predecessor if the condition CHECK = F is met.*

Appendix C lists more examples of flowcharts and their conceptual graphs.

Chapter 3. EDIF and Viewdraw

This chapter describes the Electronic Data Interchange Format or EDIF that is used in this project to represent a flowchart in textual form. It also describes Viewdraw, a schematic capture tool used to draw the graphic representation of the flowchart and obtain its EDIF representation. Listed at the end of this chapter is an example flowchart and its EDIF netlist and a brief explanation of the netlist.

3.1 The Electronic Data Interchange Format

EDIF is a format for the interchange of data between electronic design systems [6]. It can transfer data about all aspects of a design including netlists, schematics and schematic symbols, physical layouts of integrated circuits and printed circuit boards, simulation models and simulation waveforms, timing, back annotation, and physical design rules. EDIF allows users of CAD and CAE systems to integrate tools from various vendors without the need for writing software to link the tools and to migrate designs from older to newer systems.

The EDIF effort was begun in November 1983 with the primary aim of transferring data between CAE tools and ASIC vendors. The first version of EDIF was released in May 1984 and following versions were released in March 1985, November 1985, and June 1986. Refinements in each version were a result of the evolution of tools that required new data items to be represented or more information to be transferred about a particular data item [6].

The syntax of EDIF is similar to the syntax of LISP. It depends on parentheses to delimit constructs and the use of keywords to identify these constructs. EDIF was chosen to be used in this project because the use of parentheses and keywords to delimit constructs makes it easy to parse. Shown in Figure 3.1 is the format of EDIF. The portion shown in italics is the inner construct, delimited from the outer construct by parentheses and the keyword *keyword_2*.

```
(keyword_1 argument_list
    description
    ...................
    ...................
    (keyword_2 argument_list
        description
        .................. ) )
```

Figure 3.1 Format of EDIF

In EDIF, various views such as schematic view, netlist view and symbolic view are used to represent different types of information about the design. This project uses only the netlist view of EDIF which describes the connectivity between the blocks displayed in the flowchart.

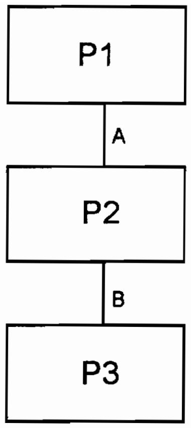


Figure 3.2. Example flowchart

The first block of the EDIF file is a header that contains accounting information such as the time the file was written or generated and who wrote it. Figure 3.3 shows the header block of the EDIF representation of the flowchart shown in Figure 3.2.

```
(edif example
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
     (timeStamp 1994 5 17 23 36 1)
     (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EXAMPLE
   (edifLevel 0)
   (technology (numberDefinition(scale1(E1 -12)(unit
                                              CAPACITANCE))))
```

Figure 3.3. EDIF file - Header block

The next block describes the various blocks in the flowchart. The keyword *cell* is used to refer to these blocks. The name of the type of the block appears following the keyword. Identifiers are used in EDIF to name and refer to various objects. These identifiers must be between 1 and 255 characters long and may consist only of alphabetic characters, digits and underscores. If the first character of an identifier name is not a letter, it must be an ampersand (&) sign. In this section, the EDIF file describes each of the blocks of the flowchart. The types of blocks present in the flowchart of Figure 3.2 are shown in Figure 3.3.

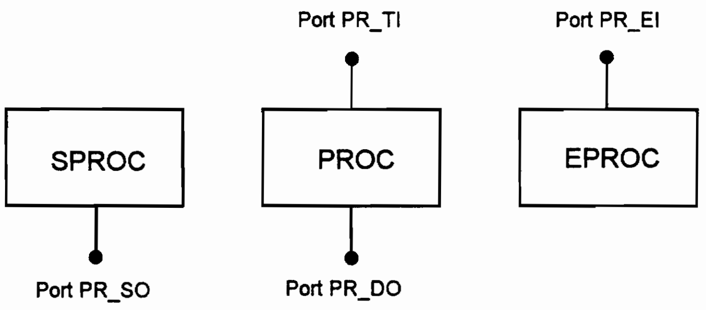


Figure 3.4 Blocks of the flowchart in Fig. 3.2.

Consider the block type PROC. The description of this block contains information about the cell type and the view type. In addition the EDIF file also provides information about the two ports of the block: PR_TI which is the input port (flowline attachment point) and PR_DO which is the output port. The ports of the flowchart blocks have been named in such a way that the last letter in the name of the port specifies whether the port is an input or output port. The other components in the design are also described in a similar fashion. The section of the EDIF file describing the flowchart blocks of the example flowchart is shown in Figure 3.5.

```
(cell SPROC
  (cellType GENERIC)
  (userData VL_PLATFORM CODE (string "SPROC")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface
      (port PR_SO
        (direction OUTPUT)))
  ))

(cell EPROC
  (cellType GENERIC)
  (userData VL_PLATFORM_CODE (string "EPROC")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface
      (port PR_EI
        (direction INPUT)))
  ))


(cell PROC
  (cellType GENERIC)
  (userData VL_PLATFORM_CODE (string "PROC")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface
      (port PR_DO
        (direction OUTPUT))
      (port PR_TI
        (direction INPUT)))
  ))
```

Figure 3.5 EDIF file - Description of flowchart block types

The next section of the EDIF file describes the whole flowchart itself. The keyword *instance* is used to refer to the instantiation of each block in the flowchart. The name of the block as given in the flowchart, appears following the keyword. Since each block has already been described earlier in the EDIF file, a cell reference is quoted which refers to the type of the block using the keyword *cellRef*. For example,

```
(instance P1
  (viewRef view_1 (cellRef SPROC))
```

denotes that instantiation of block referred to by identifier P1 is of type SPROC. This section of the EDIF file that describes the example flowchart is shown in Figure 3.6.

```
(cell EXAMPLE
  (cellType GENERIC)
    (status (written
     (timeStamp 1994 5 17 23 18 45)))
  (userData VL_PLATFORM_CODE (string "edif")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface)
    (contents
      (instance P1
        (viewRef view_1 (cellRef SPROC))
        (property LABEL (string "P1") (owner
                     "Viewlogic_Systems")))
      (instance P3
        (viewRef view_1 (cellRef EPROC))
        (property LABEL (string "P3") (owner
                     "Viewlogic_Systems")))
      (instance P2
        (viewRef view_1 (cellRef PROC))
        (property LABEL (string "P2") (owner
                     "Viewlogic_Systems")))
```

Figure 3.6 EDIF file - Description of the flowchart

The final section of the EDIF netlist file describes the connectivity between the blocks of in the flowchart. The keyword *net* following the parenthesis is used to refer to the label on the flowline connecting two blocks in the flowchart. This construct also specifies the names of the ports on the flowchart blocks that are connected by the net.

```
(net A
  (joined
    (portRef PR_TI (instanceRef P2))
    (portRef PR_SO (instanceRef P1))))
```

The example netlist above denotes that flowline with a label A connects port PR_SO of block P1 and port PR_TI of block P2 in Figure 3.2. Figure 3.7 shows the netlist view of the flowchart of Figure 3.2.

```
(net A
   (joined
     (portRef PR_TI (instanceRef P2))
     (portRef PR_SO (instanceRef P1))))
 (net B
   (joined
     (portRef PR_DO (instanceRef P2))
     (portRef PR_EI (instanceRef P3))))
```

Figure 3.7. EDIF file - Netlist view

3.2. Viewdraw

Viewdraw is the schematic capture tool used in this project to draw and then generate the EDIF netlist of the flowchart whose conceptual graph representation is to be obtained. This tool was developed by Viewlogic Systems Inc., Massachusetts as part of Powerview, a software now widely used by many industries involved in digital systems design, simulation and synthesis.

Viewdraw is a schematic capture tool that is primarily used to represent digital circuits for the purpose of simulation and to obtain netlists. The flowchart is drawn in Viewdraw as a *schematic* comprising of standard *symbols* located in the symbol library. The library of symbols is presented in Appendix B. The EDIF netlist of the flowchart is then obtained by using the 'edifneto' command on the command line prompt. The procedure to be followed to draw the flowchart using Viewdraw is presented in Appendix A.

Chapter 4. Generation of Conceptual Graphs

A flowchart describing the procedure to be followed to generate the conceptual graph of a flowchart is shown in figure 4.1. This chapter describes the algorithm used by the flowgrapher to generate conceptual graphs from the EDIF netlist representation of a flowchart. The process of generating a conceptual graph from the EDIF netlist view of a flowchart is divided into the following sections.

1. Parse the netlist and store net information
2. For each process block
    a. Find successor blocks and update successor information
    b. Find predecessor blocks and update predecessor information
3. Output conceptual graph

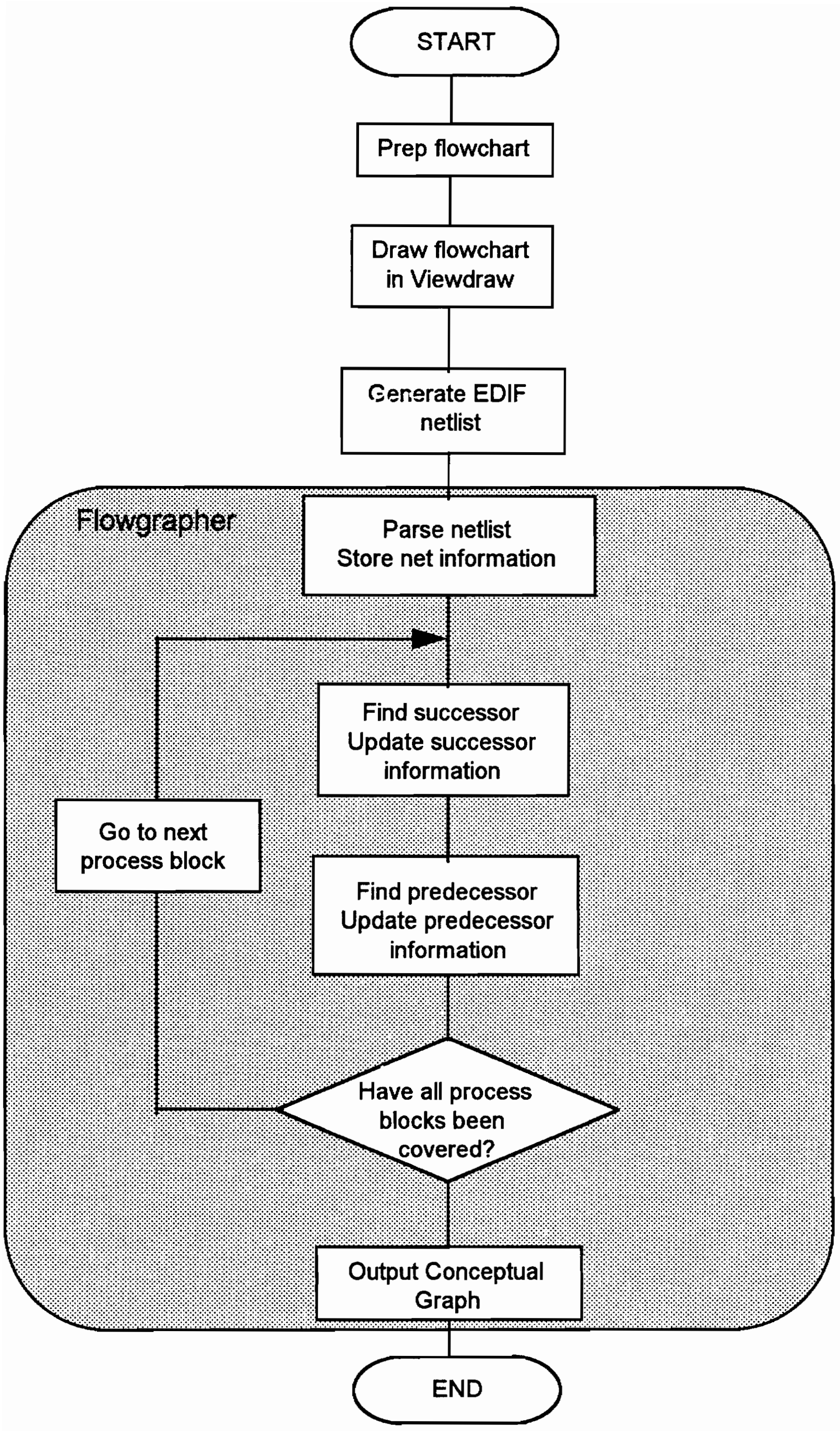

START
Prep flowchart
Draw flowchart in Viewdraw
Generate EDIF netlist
Flowgrapher
Parse netlist Store net information
Find successor Update successor information
Go to next process block
Find predecessor Update predecessor information
Have all process blocks been covered?
Output Conceptual Graph
END

Figure 4.1 Flowchart of the Flowgrapher

4.1 The Parser

The parser is implemented by a function *parse()* in the program. The parser accepts the netlist representation of the flowchart in the Electronic Data Interchange Format (EDIF) as an input and outputs a structure containing the following information about each flowchart block.

1. The name (label) of the block
2. The type of the block
3. The concept number of the block
4. Successor information (singly linked list)
   a. The name (label) of each successor
   b. The block type of each successor
   c. The concept number of each successor
   d. The identifier of the flow line that connects each successor to the process block
5. Predecessor information (singly linked list)
   a. The name (label) of each predecessor
   b. The block type of each predecessor
   c. The concept number of each predecessor
   d. The identifier of the flow line connecting each predecessor to the process block

The Electronic Data Interchange Format depends on parentheses to delimit constructs and the use of keywords to identify these constructs.

Therefore the parser searches for the occurence of an open parenthesis '(' first. When an open parenthesis is found, the parser compares the keyword following the parenthesis with the required keyword. Initially, the parser looks for the keyword *instance*. In the EDIF representation of a flowchart, the keyword *instance* is found following the open parenthesis in the block that describes the components present in the flowchart. This block presents the name of each instance (following the keyword *instance*), and the block type of each instance (following the keyword *cellref*). The parser reads the required information and the respective fields in the structure are updated. The EDIF file of the flowchart of in Figure 3.2 is shown in Figure 4.2. The instance (block) names and block types are shown in bold characters. All the flowchart blocks that are not process blocks, eg. or-join, decision, and-join and and-fork blocks are assigned a concept number 0. Process blocks are assigned concept numbers in ascending order beginning with 1, in the order in which they are read by the parser. In the example shown in Figure 4.2, P1 is an instance of type SPROC, P2 is an instance of type PROC and P3 is an instance of type EPROC. P1 is assigned a concept number 1. Process P3 which is read before process 2 is assigned concept number 2 and process P2, concept number 3. Up to this point in parsing the EDIF file, the keywords sought by the parser are

1. *instance* (the word following this keyword is the block name)
2. *cellref* (the word following this keyword is the block type)

All the other keywords are ignored.

```
(instance P1
  (viewRef view_1 (cellRef SPROC))
  (property LABEL (string "P1")))
(instance P3
  (viewRef view_1 (cellRef EPROC))
  (property LABEL (string "P3")))
(instance P2
  (viewRef view_1 (cellRef PROC))
  (property LABEL (string "P2")))
```

Figure 4.2. Portion of EDIF file showing instance names and block types

The next step in parsing is the updating of the successor and predecessor relations. This is done using the netlist view of the Electronic Data Interchange format. Here, the parser searches the EDIF file for the keyword *net* following an open parenthesis. When the keyword *net* is found the parser stops looking for the keyword *instance*. The netlist block of the EDIF file provides information about the name of the flowline (following the keyword *net*), names of the ports being joined by the net (following the keyword *portRef*), and the names of the flowchart blocks (following the keyword *instanceRef*). An example EDIF file is shown below.

```
(net A
  (joined
    (portRef PR_TI (instanceRef P2))
    (portRef PR_SO (instanceRef P1))))
(net B
  (joined
    (portRef PR_DO (instanceRef P2))
    (portRef PR_EI (instanceRef P3))))
```

Figure 4.3 Portion of EDIF file showing the netlist of the flowchart

The names of the ports of the symbols in the symbol library have been assigned in such a way that the last character in the name of a port specifies whether the port is an input port or an output port. The parser makes use of this property to decide whether an instance is a successor or a predecessor to another. In the example shown above, net A connects input port PR_TI of instance P2 and output port PR_SO of instance P1. As a result, the parser stores process P2 as a successor of P1 in the structure describing process block P1, and process P1 as a predecessor of process P2 in the structure describing process block P2. The name of the net and the concept number of the corresponding block is also stored in the structure. A tabular form of the structure obtained as an output from the parser for the EDIF file of the example flowchart of Figure 4.4 is shown in Figure 4.5.

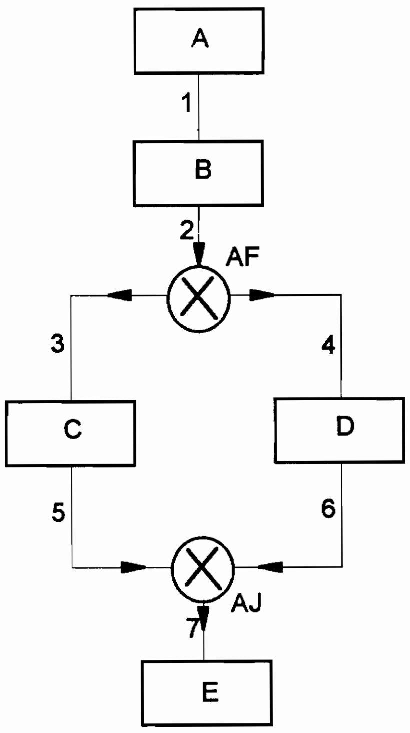


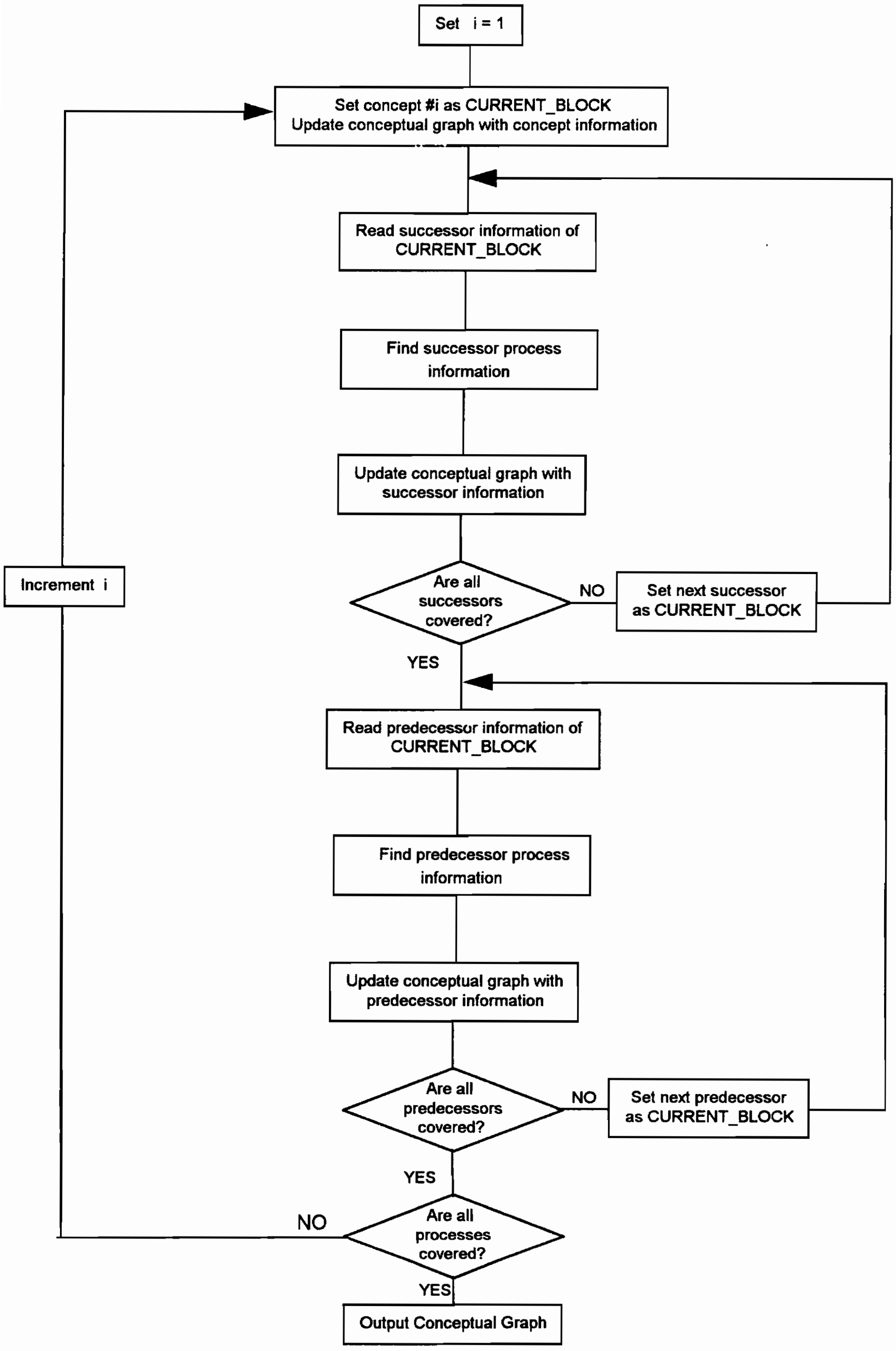
Figure 4.4. Example flowchart

Name of Block	Block type	Concept number	Successor Information			Predecessor Information		
			Name	Block Type	Net name	Name	Block Type	Net name
A	Process	2	B	Process	1			
B	Process	1	AF	And-fork	2	A	Process	1
AF	And-fork	0	C	Process	3	B	Process	2
			D	Process	4			
C	Process	3	AJ	And-join	5	AF	And-fork	3
D	Process	4	AJ	And-join	6	AF	And-fork	4
AJ	And-join	0	E	Process	7	C	Process	5
						D	Process	6
E	Process	5				AJ	And-join	7

Figure 4.5. Structure output by the parser

4.2 The Conceptual Graph Generator

The generation of the conceptual graph from the output of the parser is accomplished by the function *con_gen()*. A flowchart explaining the algorithm of this function is shown in Figure 4.6. The explanation of the algorithm is presented following the flowchart.



Set i = 1
Set concept #i as CURRENT_BLOCK
Update conceptual graph with concept information
Read successor information of CURRENT_BLOCK
Find successor process information
Update conceptual graph with successor information
Are all successors covered?
NO
Set next successor as CURRENT_BLOCK
YES
Read predecessor information of CURRENT_BLOCK
Find predecessor process information
Update conceptual graph with predecessor information
Are all predecessors covered?
NO
Set next predecessor as CURRENT_BLOCK
YES
Are all processes covered?
NO
Increment i
YES
Output Conceptual Graph

Figure 4.6. Flowchart of the algorithm of con_gen()

When the parser writes the block information of each flowchart block into their respective structures, process blocks are assigned concept numbers in the ascending order beginning with 1, in the order in which they are read by the parser. The block information stored in these structures is read by the function *con_gen()*, to generate the conceptual graph of the flowchart described by the EDIF netlist file.

The structure containing the information about the process block with concept number 1 is read first. The conceptual graph is updated with the concept number, concept type and the referent of the concept (representing the block instance). The successor and predecessor information is updated next. The following rules are followed while updating the successor information.

For each successor,

- If the successor block is not a process block, then the structure describing the successor block is accessed to find the successor of that block. This process is continued recursively until a process block is reached and until all the successors of a given process block are searched exhaustively.

- The name on an output of a decision block corresponds to the outcome of the test conducted within the decision block, that would transfer the flow of control to that net. Hence, if the successor is a decision block, the successor information of the decision block is read along with the name of the net connecting the decision block to its successor. This information is used to update the conceptual graph with the *succ_if* relation of the process block with its successor.

Similar rules are also applied for updating the conceptual graph with the predecessor information. After all the successors and predecessors of a process block have been searched exhaustively, the structure containing the information about the process bearing the next higher concept number is read. This procedure is continued repetitively until all the process blocks have been covered.

Shown in Figure 4.7 is an example flowchart, Its EDIF netlist representation and the contents of the structure obtained as the output of the parser and the conceptual graph representation of the flowchart are shown in Figures 4.8, 4.9, and 4.10 respectively. The contents of Figure 4.9 are summarized in the table shown in Figure 4.5.

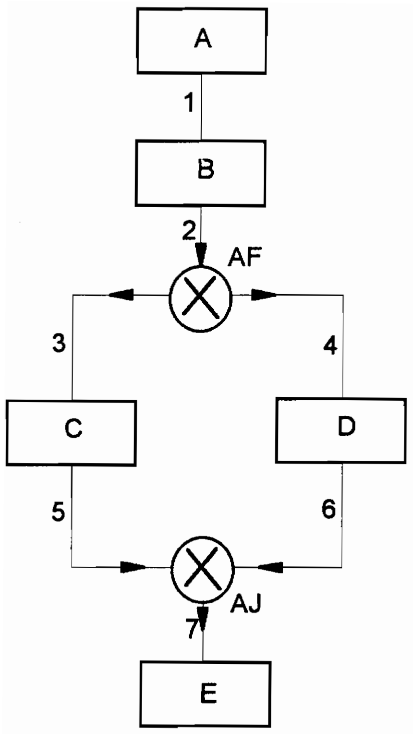


Figure 4.7 Example flowchart

EDIF netlist of example flowchart

```
(edif example
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 13 1 37 41)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EXAMPLE
    (edifLevel 0)
    (technology (numberDefinition (scale 1 (E 1 -12) (unit
CAPACITANCE))))

    (cell SPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 230016930800 SPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_SO
            (direction OUTPUT))
      ))

    (cell PROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 311691258600 PROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_DO
            (direction OUTPUT))
          (port PR_TI
            (direction INPUT)))
      ))

    (cell AND_FORK_L
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 242186273300 AND_FORK_L")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port AF_DO
            (property PIN (string "") (owner "Viewlogic_Systems")))
          (port AF_LO
            (property PIN (string "") (owner "Viewlogic_Systems")))
          (port AF_TI
            (property PIN (string "") (owner "Viewlogic_Systems"))))
      ))
```

Figuire 4.8. EDIF netlist of example flowchart of Fig. 4.7. (continued)

```
(cell AND_JOIN_L
  (cellType GENERIC)
  (userData VL_PLATFORM_CODE (string "K 240975675100 AND_JOIN_L")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface
      (port AJ_DO
        (property PIN (string "") (owner "Viewlogic_Systems")))
      (port AJ_LI
        (property PIN (string "") (owner "Viewlogic_Systems")))
      (port AJ_TI
        (property PIN (string "") (owner "Viewlogic_Systems"))))
  ))

(cell EPROC
  (cellType GENERIC)
  (userData VL_PLATFORM_CODE (string "K 128442474600 EPROC")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface
      (port PR_EI
        (direction INPUT)))
  ))

(cell EXAMPLE
  (cellType GENERIC)
    (status (written
     (timeStamp 1994 6 13 1 34 28)))
  (userData VL_PLATFORM_CODE (string "K 296887807400 example")
    (owner "Viewlogic_Systems"))
  (view view_1
    (viewType NETLIST)
    (interface)
    (contents
      (instance A
        (viewRef view_1 (cellRef SPROC))
        (property LABEL (string "A") (owner "Viewlogic_Systems")))
      (instance B
        (viewRef view_1 (cellRef PROC))
        (property LABEL (string "B") (owner "Viewlogic_Systems")))
      (instance AF
        (viewRef view_1 (cellRef AND_FORK_L))
        (property LABEL (string "AF") (owner "Viewlogic_Systems")))
      (instance C
        (viewRef view_1 (cellRef PROC))
        (property LABEL (string "C") (owner "Viewlogic_Systems")))
      (instance D
        (viewRef view_1 (cellRef PROC))
        (property LABEL (string "D") (owner "Viewlogic_Systems")))
      (instance AJ
        (viewRef view_1 (cellRef AND_JOIN_L))
        (property LABEL (string "AJ") (owner "Viewlogic_Systems")))
```

Figuire 4.8. EDIF netlist of example flowchart of Fig. 4.7. (continued)

```
          (instance E
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "E") (owner "Viewlogic_Systems")))
          (net &1
            (joined
              (portRef PR_TI (instanceRef B))
              (portRef PR_SO (instanceRef A))))
          (net &2
            (joined
              (portRef AF_TI (instanceRef AF))
              (portRef PR_DO (instanceRef B))))
          (net &3
            (joined
              (portRef AF_LO (instanceRef AF))
              (portRef PR_TI (instanceRef C))))
          (net &4
            (joined
              (portRef AF_DO (instanceRef AF))
              (portRef PR_TI (instanceRef D))))
          (net &5
            (joined
              (portRef AJ_LI (instanceRef AJ))
              (portRef PR_DO (instanceRef C))))
          (net &6
            (joined
              (portRef AJ_TI (instanceRef AJ))
              (portRef PR_DO (instanceRef D))))
          (net &7
            (joined
              (portRef AJ_DO (instanceRef AJ))
              (portRef PR_EI (instanceRef E))))
        )
      ))

  )
  (design ROOT
    (cellRef EXAMPLE
      (libraryRef EXAMPLE)))
)
```

Figuire 4.8. EDIF netlist of example flowchart of Fig. 4.7.

Parser Output showing individual block information.

```
BLOCK INFORMATION
Number of process blocks = 5
-----------------------------
Instance is A
Block type is PROCESS
Concept Number is 2

SUCCESSORS
Successor instance is B
Successor block type is PROCESS
Net name is 1

PREDECESSORS


Instance is B
Block type is PROCESS
Concept Number is 1

SUCCESSORS
Successor instance is AF
Successor block type is AND FORK
Net name is 2

PREDECESSORS
Predecessor instance is A
Predecessor block type is PROCESS
Net name is 1


Instance is AF
Block type is AND FORK
Concept Number is 0

SUCCESSORS
Successor instance is C
Successor block type is PROCESS
Net name is 3

Successor instance is D
Successor block type is PROCESS
Net name is 4
```

Figure 4.9. Parser Output showing individual block information (continued)

```
PREDECESSORS
Predecessor instance is B
Predecessor block type is PROCESS
Net name is 2
Instance is C
Block type is PROCESS
Concept Number is 3

SUCCESSORS
Successor instance is AJ
Successor block type is AND JOIN
Net name is 5

PREDECESSORS
Predecessor instance is AF
Predecessor block type is AND FORK
Net name is 3


Instance is D
Block type is PROCESS
Concept Number is 4

SUCCESSORS
Successor instance is AJ
Successor block type is AND JOIN
Net name is 6

PREDECESSORS
Predecessor instance is AF
Predecessor block type is AND FORK
Net name is 4


Instance is AJ
Block type is AND JOIN
Concept Number is 0

SUCCESSORS
Successor instance is E
Successor block type is PROCESS
Net name is 7

PREDECESSORS
Predecessor instance is C
Predecessor block type is PROCESS
Net name is 5
```

Figure 4.9. Parser Output showing individual block information (continued)

```
Predecessor instance is D
Predecessor block type is PROCESS
Net name is 6



Instance is E
Block type is PROCESS
Concept Number is 5

SUCCESSORS

PREDECESSORS
Predecessor instance is AJ
Predecessor block type is AND JOIN
Net name is 7
```

Figure 4.9. Parser Output showing individual block information.

Conceptual graph representation of the flowchart

```
[ 1 : process : B ]
      ->( succ ) -> [ 3 ]
      ->( succ ) -> [ 4 ]
      ->( pred ) -> [ 2 ]

[ 2 : process : A ]
      ->( succ ) -> [ 1 ]

[ 3 : process : C ]
      ->( succ ) -> [ 5 ]
      ->( pred ) -> [ 1 ]

[ 4 : process : D ]
      ->( succ ) -> [ 5 ]
      ->( pred ) -> [ 1 ]

[ 5 : process : E ]
      ->( pred ) -> [ 3 ]
      ->( pred ) -> [ 4 ]
```

Figure 4.10. Conceptual graph representation of the flowchart of Fig. 4,7.

Explanation of the example

In the example shown above, the structure containing information on process block B shown below is read first.

```
Instance is B
Block type is PROCESS
Concept Number is 1
```

The conceptual graph is updated with the concept information as shown.

```
[ 1 : process : B ]
```

where *'1'* is the concept number, *'process'* is the concept type and *'B'* is the concept referent derived from the name of the instance.

The successor information of the process block B shown below is read next.

```
SUCCESSORS
Successor instance is AF
Successor block type is AND FORK
Net name is 2
```

Since the successor is an and-fork block and not a process block, the structure containing the information on the and-fork block is accessed and the information about the successors of the and-fork block shown below is read.

```
Instance is AF
Block type is AND FORK
Concept Number is 0

SUCCESSORS
Successor instance is C
Successor block type is PROCESS
Net name is 3

Successor instance is D
Successor block type is PROCESS
Net name is 4
```

The and-fork block has two successors and both are process blocks. The structure containing the information about the two successors is accessed to find their concept numbers and the conceptual graph is now updated with the successor information as shown.

```
[ 1 : process : B ]
     ->( succ ) -> [ 3 ]
     ->( succ ) -> [ 4 ]
```

[3] and [4] are concept numbers of process blocks C and D. Since all the successors of the and-fork block have now been covered, the successor information of process block B is read for its next successor. Since the and-fork block is the only successor of process block B, the predecessor information of process block B shown below is read next.

```
PREDECESSORS
Predecessor instance is A
Predecessor block type is PROCESS
Net name is 1
```

Since the predecessor is a process block, the structure containing the information about the predecessor is accessed to find its concept numbers and the conceptual graph is now updated with the predecessor information as shown.

```
[ 1 : process : B ]
     ->( succ ) -> [ 3 ]
     ->( succ ) -> [ 4 ]
     ->( pred ) -> [ 2 ]
```

In a similar way, the structures containing the information about the other flowchart blocks are read and the conceptual graph is updated at every stage.

Appendix C lists more examples of flowcharts, their EDIF netlists and the conceptual graph representation of those flowcharts obtained from the Flowgrapher. A summary of the results is shown in Figure 4.11.

	Ex. 1	Ex. 2	Ex. 3	Ex. 4	Ex. 5	Ex. 6
Flowchart Information						
Number of process blocks	5	5	3	4	10	19
Number of and-fork blocks	1	0	0	0	0	0
Number of and-join blocks	1	0	0	0	0	0
Number of or-join blocks	0	0	1	0	3	12
Number of decision blocks	0	1	1	2	3	15
Number of flowlines	7	5	5	5	18	31
Total number of blocks	7	6	5	6	16	46
Conceptual Graph Information						
Number of concepts	5	9	7	10	22	76
Number of relations	10	8	6	6	24	142

Figure 4.11. Summary of results

REFERENCES

[1] Angevine, "An Introduction to EDIF", Semicustom Design Guide, pp. 26-35 1987.

[2] H. Boillot, G.M. Gleason, and L.W. Horn, Essentials of Flowcharting , Wm. C. Brown, Company Publishers, Dubuque, IA, 1979.

[3] W. Cyre, "Integrating Specification requirements for Automated Interpretation", Second International Workshop on Rapid System Prototyping, 1991.

[4] H.S.Delugach, "Specifying Multiple Viewed Software Requirements with Conceptual Graphs", Journal of Systems Software, vol 19 pp 207-224, 1992.

[5] M.V. Farina, Flowcharting, Prentice Hall Inc., Englewood Cliffs, NJ, 1970.

[6] "EDIF Specification : EDIF Electronic Design Interchange Format Version 2.0.0", Electronic Industries Association, Washington DC, 1987.

[7] J.F. Sowa, Conceptual Structures : Information Processing in Mind and Machine, Addisson-Wesley, Reading, MA, 1984.

[8] R.J. Traister, Mastering C Pointers, Academic Press inc. CA, 1990.

[9] "Viewdraw Reference Manual", Viewlogic Systems Inc., MA, 1992.

Appendix A. User Manual

The procedure to be followed to use the Flowgrapher to generate the conceptual graph of a flowchart is explained in this appendix. The material presented here is written towards helping the user to draw a flowchart using Powerview's Viewdraw on a SunOS platform, and to generate its EDIF netlist. Also explained is the procedure to be followed to obtain the conceptual graph representation of the flowchart from the EDIF netlist using *fl2con*, the executable code of the program described in this project.

The procedure to be followed to generate a the conceptual graph of a flowchart using the Flowgrapher has been divided into three sections as listed below.

1. Preparing the flowchart to be drawn in Viewdraw
2. Drawing the flowchart in Viewdraw and generating its EDIF netlist
3. Generating the conceptual graph with Flowgrapher

A.1. Preparing

Viewdraw, the tool used to draw the flowchart and generate its EDIF netlist view representation, imposes a few restrictions on the representation of the flowchart. Hence, the following changes must be made to the flowchart before it can be drawn using Viewdraw.

1. Since the Flowgrapher accepts only a subset of the standard flowcharting symbols, ensure that the flowchart comprises only of the symbols available in the standard library developed. The symbol library is presented in Appendix B.

2. Component identifiers in Viewdraw can be up to 256 characters long, comprising of alphabets, digits and underscores only. The first character in those names must be an alphabet. Therefore, replace the text in the blocks of the flowchart with unique identifiers. Also, ensure that no two flowchart blocks have the same label. A separate list relating the new block

identifiers to the text that was previously present in the blocks can be maintained for future reference.

3. Give unique labels to the flowlines. These identifiers as described earlier, can be of utmost 256 characters in length comprising of alphabets, digits and underscores only. Ensure that no two flowlines have the same label. If two or more decision blocks have the same outputs (such as T,F or Y,N) use numeric characters to differentiate between them. (such as T1, F1, T2, F2 etc.). Another helpful suggestion would be to attach the identifier of the decision process as a prefix to the identifier of the flowline incident on it (such as D1T, D3F, D5N, D6Y, etc.,).

4. Replace the standard flowcharting symbols for the or-join, and-join and the and-fork blocks with the respective symbols developed for use in the Flowgrapher. These symbols are listed in the symbol library shown in Appendix B. After introducing the new symbols, give unique labels to the flowlines incident on them. Examples of flowcharts with the standard symbols for the or-join, and-fork and and-join operations, and flowcharts with the corresponding modified symbols are shown in Figure A.1.

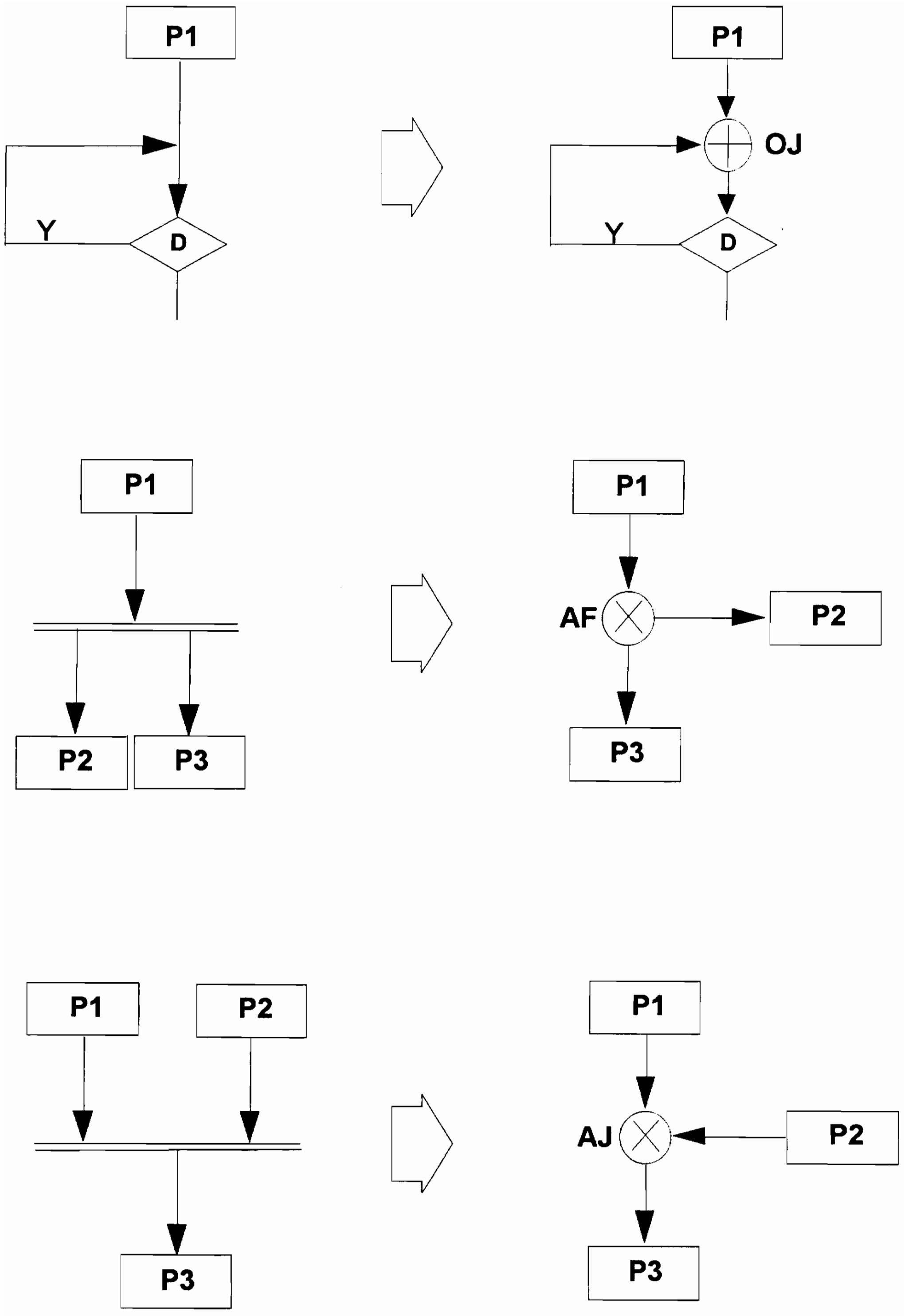
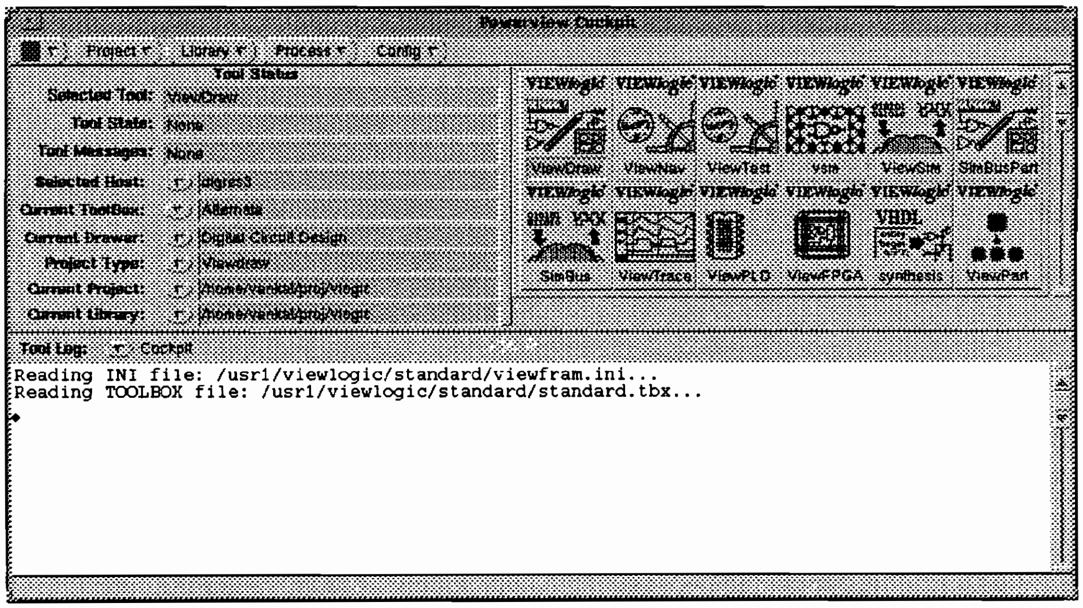


Figure A.1. Modifications of standard symbols

A.2 Drawing a flowchart in Viewdraw

The procedure to be followed to draw a flowchart using Powerview Viewdraw on a Sun SPARC workstation is explained below. The same can be used to draw the flowchart using Workview Viewdraw on a PC. However the appearance of the windows will be different on a PC from the ones shown below.

Powerview can be invoked on the Sun SPARC stations by typing *powerview* on the command line prompt. When Powerview is invoked, the Powerview Cockpit, that is used to invoke the various tools offered by Powerview, appears as shown. Viewdraw can be started by clicking on the appropriate icon on the Powerview Cockpit.



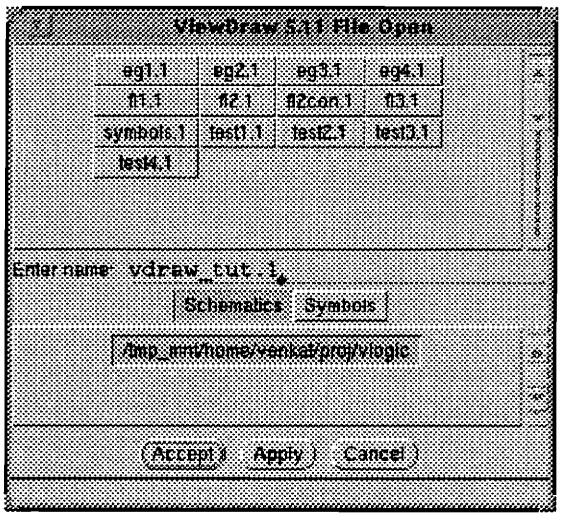
Tool Status
Selected Tool: ViewDraw
Tool State: None
Tool Messages: None
Selected Host: digres3
Current ToolBox: Alternate
Current Drawer: Digital Circuit Design
Project Type: Viewdraw
Current Project: /home/venkat/proj/vlogic
Current Library: /home/venkat/proj/vlogic
Tool Log: Cockpit

```
Reading INI file: /usr1/viewlogic/standard/viewfram.ini...
Reading TOOLBOX file: /usr1/viewlogic/standard/standard.tbx...
◆
```

Figure A.2. Powerview Cockpit

Procedure to draw a flowchart using Viewdraw

1. Start Viewdraw by clicking the left mouse button on the Viewdraw icon on the Powerview cockpit. The Viewdraw File Open window shown in Figure A.3 appears.



Enter name: vdraw_tut.1

/tmp_mnt/home/venkat/proj/vlogic

Figure A.3. Viewdraw File Open window

2. The flowchart is drawn as a *schematic* in Viewdraw. Therefore, with the "Schematics" option highlighted in the Viewdraw File Open window, provide a filename for the schematic of the flowchart with an extension '1'.

   For example : flowchart.1

   Click on 'Accept'.

   A schematic window with schematic sheet dimensions 1700x1100 appears. These dimensions can be altered using the *Block-> Size -> Z=WxH* sequence in the *Change* menu. and entering the desired dimensions for the width and the height of the schematic sheet in the window that appears.

The actions performed by the three mouse buttons are displayed on the top of the schematics window.

3. Choose the *Comp* option in the *Add* menu. A window containing the symbols in the symbol library as shown in Figure A.4 appears. Click on the desired symbol and drag it into the drawing area. The symbol can be positioned anywhere in the drawing area using the middle mouse button.

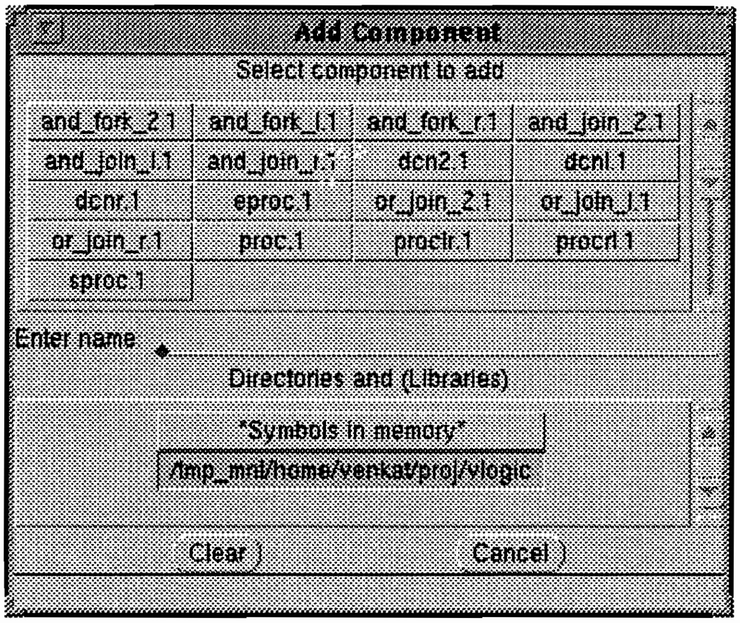


Figure A.4. Add Component window

4. After all the blocks in the flowchart have been placed, the name of each block is attached as a label to the block. This is done as follows. Select the desired flowchart block using the left mouse button. Choose the *label* option in the *Add* menu. The add Label window as shown in Figure A.5

appears. Enter the desired label. The label can either be made visible or invisible by choosing the *Visibility* option that appears in this window. The other two options, *Scope* and *Sense* are not applicable here.

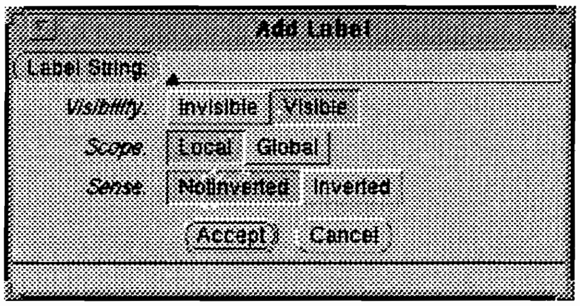


Figure A.5. Add Label window

5. Draw the flowlines between the flowchart blocks. This is done by choosing the Net option in the Add menu. Click the middle mouse button on one of the two ports that need to be connected. Drag the net that appears to the other port and click the middle mouse button again. After drawing the flowline, select the flowline using the left mouse button and follow the procedure described in Step 4 to attach a label to the flowline.

Figure A.6 shows the Viewdraw schematic window. Figure A.6(a) shows all the flowchart blocks of an example flowchart placed in position. Figure A.6(b) shows the completed flowchart with all the flowlines drawn and labeled.

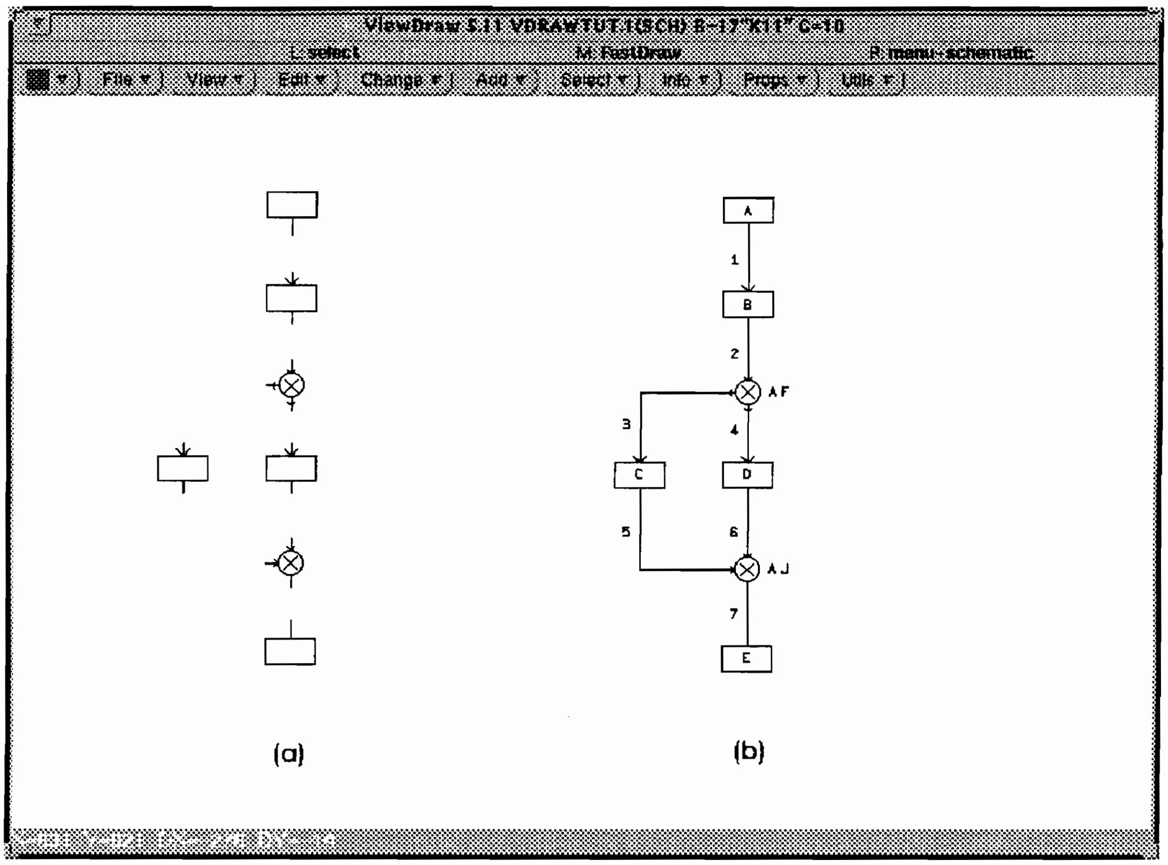


Figure A.6. Viewdraw schematics window

6. Once the flowchart has been drawn, choose the Write option in the File menu to write the flowchart to file. Viewdraw would prompt any possible errors in the schematic at this time.

Table 1. Description of selected Viewdraw menu options

Name of Menu	Option	Description
Red Square Menu	Dismiss Window	Closes current window
	Plot Setup	Setup for printing a schematic or a symbol
	Plot Go	Issues print command
	Quit Viewdraw	Exits Viewdraw
File Menu	Open	Opens a schematic or symbol file
	Write	Saves a file
	Write to	Saves a file under a different name
View Menu	Full	Shows full schematic window
	Zoom	Zooms in marked area
	In	Zooms in, in steps
	Out	Zooms out in steps
Edit Menu	Buffer - Cut	Deletes selection and stores in buffer
	Buffer - Copy	Copies selection into buffer
	Buffer - Append	Appends selection to contents of buffer
	Copy	Copies selection
	Delete	Deletes selection
	Move	Moves selection
Change Menu	Block - Size	Changes size of drawing sheet
	Label - Visibility	Changes label visibility
	Label String	Changes text in label
	Text - Font	Changes font of selected text
	Text - Size	Changes size of selected text
Add Menu	Comp	Adds component
	Label	Adds label
	Net	Adds lines connecting blocks
	Graphics - Text	Adds plain text
Screen Menu	Fast	Adds net

A.3. Generating EDIF netlist

The EDIF netlist of the flowchart drawn in Viewdraw can be generated using the following command on the command line prompt.

edifneto <schematic file name without the .1 extension>

For instance, if the schematic file name is *fl_chart.1*, then the command to generate the EDIF netlist of the flowchart would be *edifneto fl_chart.* The EDIF file, *fl_chart.edn* will be written to the current working directory. If an error message saying that the WIR file(s) are not found is received, check the current working directory to make sure that it is correct.

A.4. Conceptual Graph generation

The conceptual graph representation of a flowchart can be obtained from its EDIF netlist representation using the following command on the command line prompt.

fl2con < EDIF file name>

For example, if the EDIF representation of a flowchart is named *fl_chart.edn*, then the command to generate the conceptual graph representation of the flowchart would be *fl2con fl_chart.edn.* This outputs the conceptual graph to the screen. The conceptual graph output can also be re-directed to a file using the command

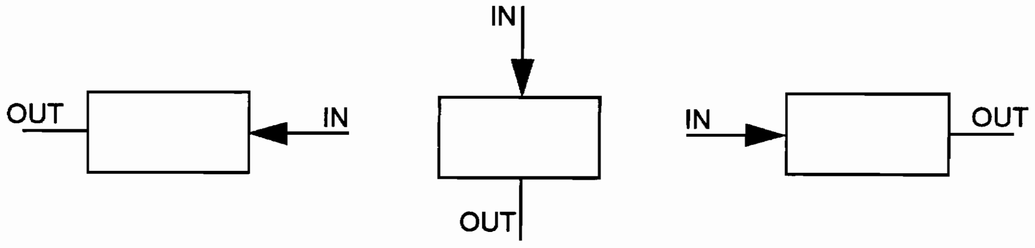
fl2con EDIF_file_name > Output_file

For example, the command *fl2con fl_chart.edn > fl_chart.cg* would write the conceptual graph representation of the flowchart, into the output file *fl_chart.cg*.

Appendix B. Viewdraw Symbol Library

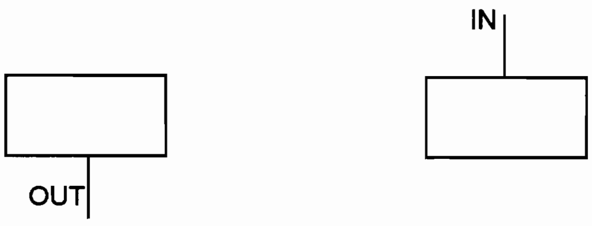
The Viewdraw symbol library containing the symbols representing the various flowchart blocks is presented below. Different versions of the same symbol have been created to enhance ease of use in representing a flowchart in Viewdraw. The input and output ports of each symbol are labeled. Also present is the name of each symbol.

Process blocks



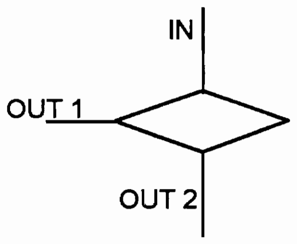
PROCLR.1 PROC.1 PROCRL.1

Terminal (connector) blocks

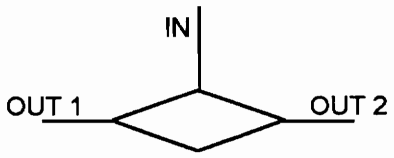


SPROC.1 EPROC.1

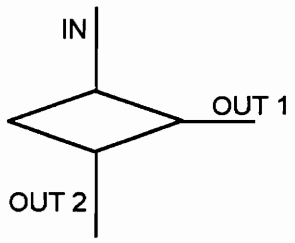
Decision blocks



DCN_L.1

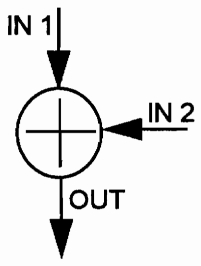


DCN2.1

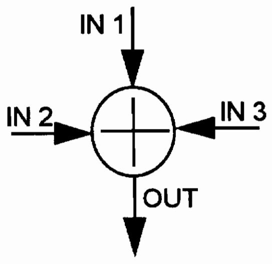


DCN_R.1

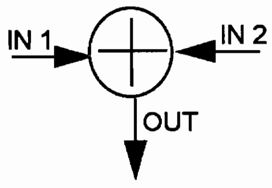
Or-join blocks



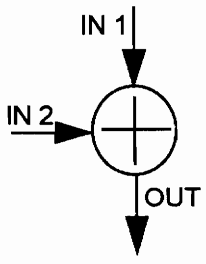
OR_JOIN_R.1



OR_JOIN3.1

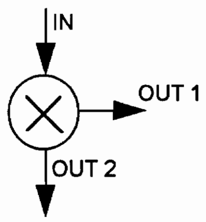


OR_JOIN_RL.1

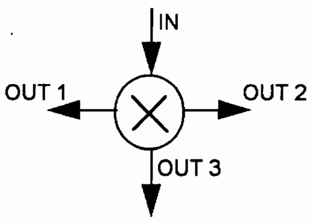


OR_JOIN_L.1

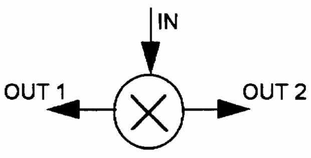
And-fork blocks



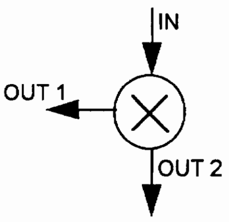
AND_FORK_R.1



AND_FORK3.1

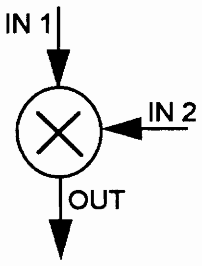


AND_FORK_RL.1

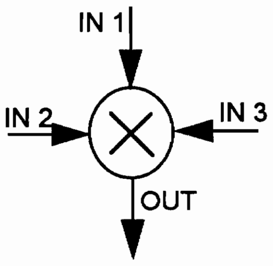


AND_FORK_L.1

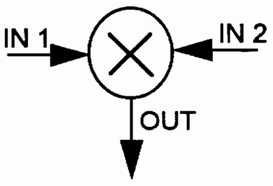
And-join blocks



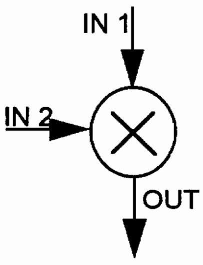
AND_JOIN_R.1



AND_JOIN3.1



AND_JOIN_RL.1

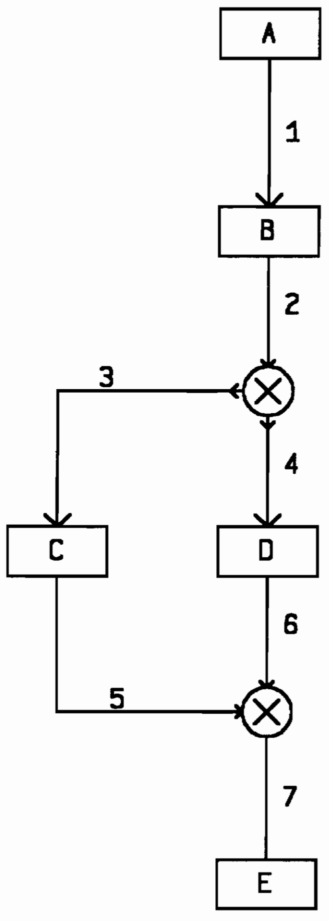


AND_JOIN_L.1

Appendix C. Examples

Examples of flowcharts, the EDIF netlist views of the flowcharts and the conceptual graph representations of the flowcharts are presented in this appendix.

Example 1 : Flowchart



Example 1 : EDIF netlist view of the given flowchart

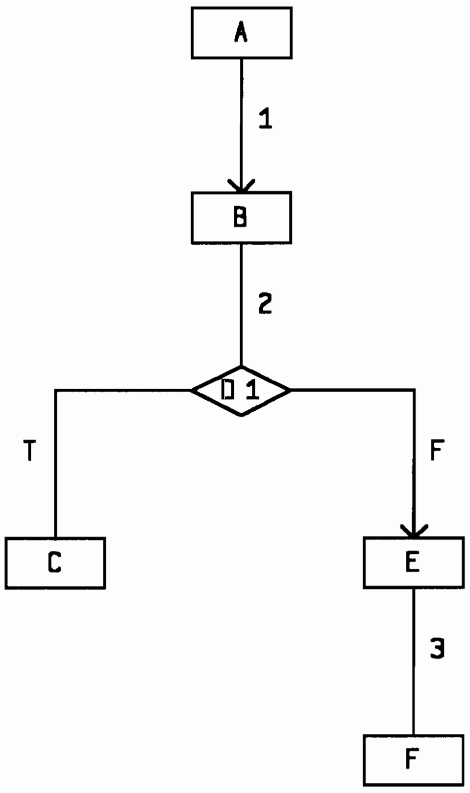
```
(edif example1
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 3 0 44 3)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EXAMPLE1
    (edifLevel 0)
    (technology (numberDefinition(scale 1(E 1 -12) (unit
CAPACITANCE))))

    (cell SPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 230016930800
SPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_SO
            (direction OUTPUT)))
      ))

    (cell AND_JOIN_L
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 240975675100
AND_JOIN_L")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port AJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port AJ_LI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port AJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell PROC
      (cellType GENERIC)
```

```
       (userData VL_PLATFORM_CODE (string "K 311691258600
PROC")
         (owner "Viewlogic_Systems"))
       (view view_1
         (viewType NETLIST)
         (interface
           (port PR_DO
             (direction OUTPUT))
           (port PR_TI
             (direction INPUT)))
       ))

     (cell EPROC
       (cellType GENERIC)
       (userData VL_PLATFORM_CODE (string "K 128442474600
EPROC")
         (owner "Viewlogic_Systems"))
       (view view_1
         (viewType NETLIST)
         (interface
           (port PR_EI
             (direction INPUT)))
       ))

     (cell AND_FORK_L
       (cellType GENERIC)
       (userData VL_PLATFORM_CODE (string "K 242186273300
AND_FORK_L")
         (owner "Viewlogic_Systems"))
       (view view_1
         (viewType NETLIST)
         (interface
           (port AF_DO
             (property PIN (string "") (owner
"Viewlogic_Systems")))
           (port AF_LO
             (property PIN (string "") (owner
"Viewlogic_Systems")))
           (port AF_TI
             (property PIN (string "") (owner
"Viewlogic_Systems"))))
       ))

     (cell EXAMPLE1
       (cellType GENERIC)
         (status (written
          (timeStamp 1994 6 2 15 41 54)))
       (userData VL_PLATFORM_CODE (string "K 11647372740O
example1")
         (owner "Viewlogic_Systems"))
       (view view_1
        (viewType NETLIST)
        (interface)
        (contents
          (instance A
            (viewRef view_1 (cellRef SPROC))
            (property LABEL (string "A") (owner
"Viewlogic_Systems")))
          (instance AJ
            (viewRef view_1 (cellRef AND_JOIN_L))
            (property LABEL (string "AJ") (owner
"Viewlogic_Systems")))
          (instance C
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "C") (owner
"Viewlogic_Systems")))
          (instance E
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "E") (owner
"Viewlogic_Systems")))
          (instance B
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "B") (owner
"Viewlogic_Systems")))
          (instance AF
            (viewRef view_1 (cellRef AND_FORK_L))
            (property LABEL (string "AF") (owner
"Viewlogic_Systems")))
          (instance D
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "D") (owner
"Viewlogic_Systems")))
          (net &1
            (joined
              (portRef PR_TI (instanceRef B))
              (portRef PR_SO (instanceRef A))))
          (net &2
            (joined
              (portRef AF_TI (instanceRef AF))
              (portRef PR_DO (instanceRef B))))
          (net &3
            (joined
              (portRef AF_LO (instanceRef AF))
              (portRef PR_TI (instanceRef C))))
          (net &4
            (joined
              (portRef AF_DO (instanceRef AF))
              (portRef PR_TI (instanceRef D))))
          (net &5
            (joined
              (portRef AJ_LI (instanceRef AJ))
              (portRef PR_DO (instanceRef C))))
          (net &6
            (joined
              (portRef AJ_TI (instanceRef AJ))
              (portRef PR_DO (instanceRef D))))
          (net &7
            (joined
              (portRef AJ_DO (instanceRef AJ))
              (portRef PR_EI (instanceRef E))))
        )
      ))

  )
  (design ROOT
    (cellRef EXAMPLE1
      (libraryRef EXAMPLE1)))
)
```

Example 1 : Conceptual graph representation of the given flowchart

```
[ 1 : process : B ]
     ->( succ ) -> [ process ] -> ( and )
             -> [ 3 : process : C ]
             -> [ 4 : process : D ]
     ->( pred ) -> [ 2 ]

[ 2 : process : A ]
     ->( succ ) -> [ 1 ]

[ 3 : process : C ]
     ->( succ ) -> [ 5 ]
     ->( pred ) -> [ 1 ]

[ 4 : process : D ]
     ->( succ ) -> [ 5 ]
     ->( pred ) -> [ 1 ]

[ 5 : process : E ]
     ->( pred ) -> [ process ] -> ( and )
             -> [ 3 : process : C ]
             -> [ 4 : process : D ]
```

Example 2 : Flowchart



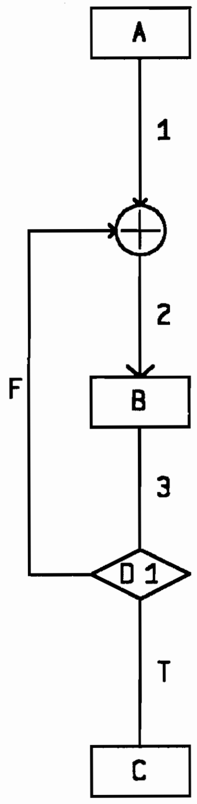
Example 2 : EDIF netlist view of the given flowchart

```
(edif eg2
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 3 0 44 13)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EG2
    (edifLevel 0)
    (technology (numberDefinition (scale 1 (E 1 -12) (unit
CAPACITANCE))))

    (cell SPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 230016930800
SPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_SO
            (direction OUTPUT)))
      ))

    (cell PROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 311691258600
PROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_DO
            (direction OUTPUT))
          (port PR_TI
            (direction INPUT)))
      ))

    (cell DCN2
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 22890886490
DCN2")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_RO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell EPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 128442474600
EPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_EI
            (direction INPUT)))
      ))

    (cell EG2
      (cellType GENERIC)
        (status (written
         (timeStamp 1994 6 2 15 47 45)))
      (userData VL_PLATFORM_CODE (string "K 178014049900
eg2")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface)
        (contents
          (instance A
            (viewRef view_1 (cellRef SPROC))
            (property LABEL (string "A") (owner
"Viewlogic_Systems")))
          (instance B
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "B") (owner
"Viewlogic_Systems")))
          (instance D1
            (viewRef view_1 (cellRef DCN2))
            (property LABEL (string "D1") (owner
"Viewlogic_Systems")))
          (instance C
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "C") (owner
"Viewlogic_Systems")))
          (instance E
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "E") (owner
"Viewlogic_Systems")))
          (instance F
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "F") (owner
"Viewlogic_Systems")))
          (net &1
            (joined
              (portRef PR_TI (instanceRef B))
              (portRef PR_SO (instanceRef A))))
          (net &2
            (joined
              (portRef DC_TI (instanceRef D1))
              (portRef PR_DO (instanceRef B))))
          (net &3
            (joined
              (portRef PR_EI (instanceRef F))
              (portRef PR_DO (instanceRef E))))
          (net F
            (joined
              (portRef DC_RO (instanceRef D1))
              (portRef PR_TI (instanceRef E))))
          (net T
            (joined
              (portRef DC_LO (instanceRef D1))
              (portRef PR_EI (instanceRef C)))))))
)
  (design ROOT
    (cellRef EG2
      (libraryRef EG2)))
)
```

Example 2 : Conceptual graph representation of the given flowchart

```
[ 1 : process : B ]
     ->( succ_if )
       1 ->[ 4 : process : E ]
       2 ->[ condition :[ D1 ]->(=)->[ F ]]
     ->( succ_if )
       1 ->[ 5 : process : C ]
       2 ->[ condition :[ D1 ]->(=)->[ T ]]
     ->( pred ) -> [ 2 ]

[ 2 : process : A ]
     ->( succ ) -> [ 1 ]

[ 3 : process : F ]
     ->( pred ) -> [ 4 ]

[ 4 : process : E ]
     ->( succ ) -> [ 3 ]
     ->( pred_if )
       1 ->[ 1 : process : B ]
       2 ->[ condition :[ D1 ]->(=)->[ F ]]

[ 5 : process : C ]
     ->( pred_if )
       1 ->[ 1 : process : B ]
       2 ->[ condition :[ D1 ]->(=)->[ T ]]
```

Example 3 : Flowchart



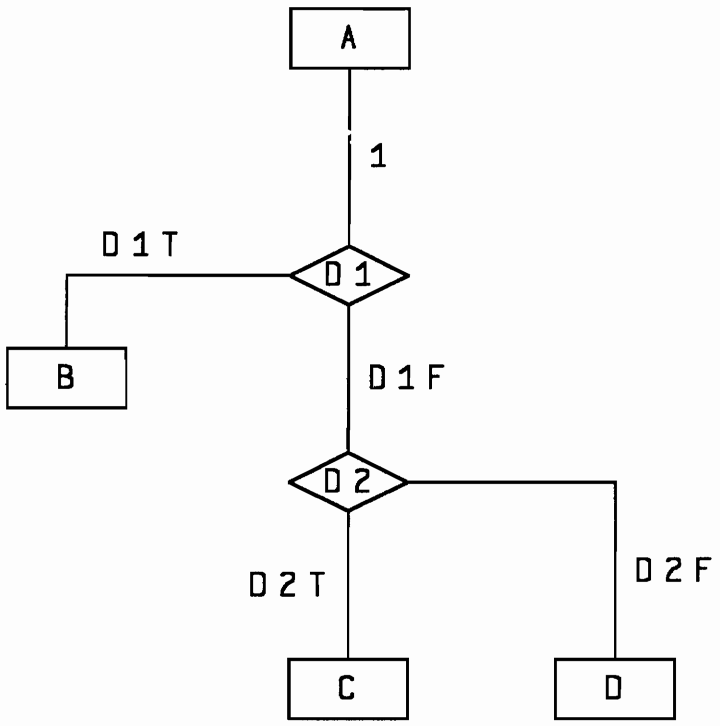
Example 3 : EDIF netlist view of the given flowchart

```
(edif eg3
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 3 0 44 23)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EG3
    (edifLevel 0)
    (technology (numberDefinition (scale 1 (E 1 -12) (unit
CAPACITANCE))))

    (cell SPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 230016930800
SPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_SO
            (direction OUTPUT)))
      ))

    (cell OR_JOIN_L
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 233118481900
OR_JOIN_L")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port OJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_LI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell PROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 311691258600
PROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_DO
            (direction OUTPUT))
          (port PR_TI
            (direction INPUT)))
      ))

    (cell DCNL
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 273091207900
DCNL")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell EPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 128442474600
EPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_EI
            (direction INPUT)))
      ))

    (cell EG3
      (cellType GENERIC)
        (status (written
         (timeStamp 1994 6 2 15 51 33)))
      (userData VL_PLATFORM_CODE (string "K 41594205690
eg3")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface)
        (contents
          (instance A
            (viewRef view_1 (cellRef SPROC))
            (property LABEL (string "A") (owner
"Viewlogic_Systems")))
          (instance OJ
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ") (owner
"Viewlogic_Systems")))
          (instance B
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "B") (owner
"Viewlogic_Systems")))
          (instance D1
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D1") (owner
"Viewlogic_Systems")))
          (instance C
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "C") (owner
"Viewlogic_Systems")))
          (net &1
            (joined
              (portRef OJ_TI (instanceRef OJ))
              (portRef PR_SO (instanceRef A))))
          (net &2
            (joined
              (portRef OJ_DO (instanceRef OJ))
              (portRef PR_TI (instanceRef B))))
          (net &3
            (joined
              (portRef DC_TI (instanceRef D1))
              (portRef PR_DO (instanceRef B))))
          (net F
            (joined
              (portRef OJ_LI (instanceRef OJ))
              (portRef DC_LO (instanceRef D1))))
          (net T
            (joined
              (portRef DC_DO (instanceRef D1))
              (portRef PR_EI (instanceRef C))))
        )
      ))

  )
  (design ROOT
    (cellRef EG3
      (libraryRef EG3)))
)
```

Example 3 : Conceptual graph representation of the given flowchart

```
[ 1 : process : A ]
     ->( succ ) -> [ 2 ]

[ 2 : process : B ]
     ->( succ_if )
       1 ->[ 2 : process : B ]
       2 ->[ condition :[ D1 ]->(=)->[ F ]]
     ->( succ_if )
       1 ->[ 3 : process : C ]
       2 ->[ condition :[ D1 ]->(=)->[ T ]]
     ->( pred ) -> [ 1 ]
     ->( pred_if )
       1 ->[ 2 : process : B ]
       2 ->[ condition :[ D1 ]->(=)->[ F ]]

[ 3 : process : C ]
     ->( pred_if )
       1 ->[ 2 : process : B ]
       2 ->[ condition :[ D1 ]->(=)->[ T ]]
```

Example 4 : Flowchart



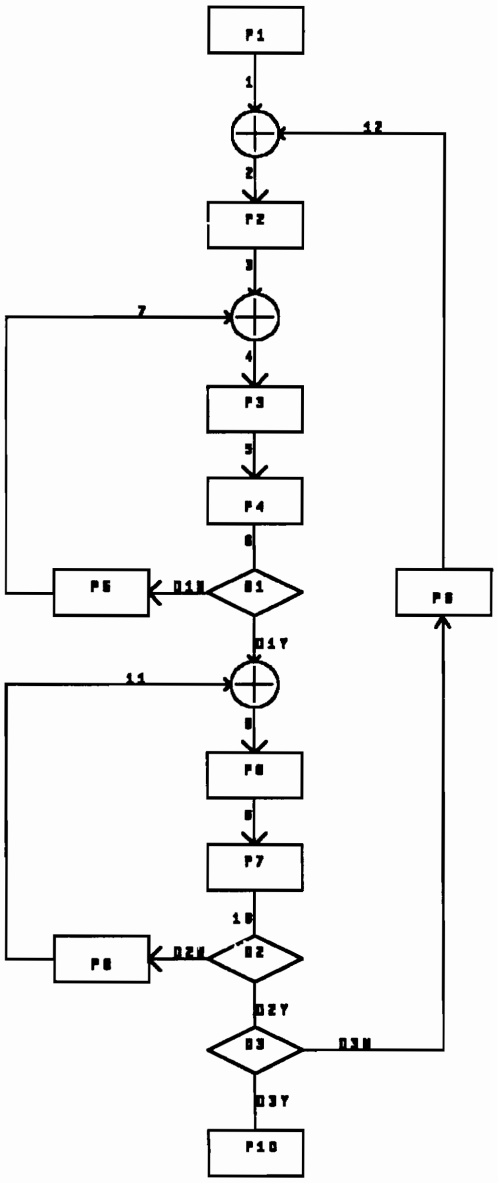
Example 4 : EDIF netlist view of the given flowchart

```
(edif eg4
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 3 0 45 18)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EG4
    (edifLevel 0)
    (technology (numberDefinition (scale 1 (E 1 -12) (unit
CAPACITANCE))))

    (cell SPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 230016930800
SPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_SO
            (direction OUTPUT)))
      ))

    (cell DCNL
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 273091207900
DCNL")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell DCNR
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 22890883290
DCNR")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_RO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell EPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 128442474600
EPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_EI
            (direction INPUT)))
      ))

    (cell EG4
      (cellType GENERIC)
        (status (written
         (timeStamp 1994 6 2 15 55 41)))
      (userData VL_PLATFORM_CODE (string "K 274789249800
eg4")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface)
        (contents
          (instance A
            (viewRef view_1 (cellRef SPROC))
            (property LABEL (string "A") (owner
"Viewlogic_Systems")))
          (instance D1
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D1") (owner
"Viewlogic_Systems")))
          (instance D2
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D2") (owner
"Viewlogic_Systems")))
          (instance B
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "B") (owner
"Viewlogic_Systems")))
          (instance C
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "C") (owner
"Viewlogic_Systems")))
          (instance D
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "D") (owner
"Viewlogic_Systems")))
          (net &1
            (joined
              (portRef DC_TI (instanceRef D1))
              (portRef PR_SO (instanceRef A))))
          (net D1F
            (joined
              (portRef DC_TI (instanceRef D2))
              (portRef DC_DO (instanceRef D1))))
          (net D1T
            (joined
              (portRef PR_EI (instanceRef B))
              (portRef DC_LO (instanceRef D1))))
          (net D2F
            (joined
              (portRef PR_EI (instanceRef D))
              (portRef DC_RO (instanceRef D2))))
          (net D2T
            (joined
              (portRef PR_EI (instanceRef C))
              (portRef DC_DO (instanceRef D2))))))))
  (design ROOT
    (cellRef EG4
      (libraryRef EG4))))
```

Example 4 : Conceptual graph representation of the given flowchart

```
[ 1 : process : A ]
     ->( succ_if )
       1 ->[ 3 : process : D ]
       2 ->[ condition :[ D1 ]->(=)->[ D1F ] [ D2 ]->(=)->
           [ D2F ]]
     ->( succ_if )
       1 ->[ 4 : process : C ]
       2 ->[ condition :[ D1 ]->(=)->[ D1F ] [ D2 ]->(=)->
           [ D2T ]]
     ->( succ_if )
       1 ->[ 2 : process : B ]
       2 ->[ condition :[ D1 ]->(=)->[ D1T ]]

[ 2 : process : B ]
     ->( pred_if )
       1 ->[ 1 : process : A ]
       2 ->[ condition :[ D1 ]->(=)->[ D1T ]]

[ 3 : process : D ]
     ->( pred_if )
       1 ->[ 1 : process : A ]
       2 ->[ condition :[D2]->(=)->[D2F]  [D1]->(=)->[ D1F ]]

[ 4 : process : C ]
     ->( pred_if )
       1 ->[ 1 : process : A ]
       2 ->[ condition :[D2]->(=)->[D2T]  [D1]->(=)->[D1F]]
```

Example 5 : Flowchart



Example 5 : EDIF netlist view of the given flowchart

```
(edif eg5
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 16 20 21 1)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library EG5
    (edifLevel 0)
    (technology(numberDefinition (scale1 (E 1 -12) (unit
CAPACITANCE))))

    (cell SPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 230016930800
SPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_SO
            (direction OUTPUT)))
      ))

    (cell DCNL
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 273091207900
DCNL")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell PROCLR
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 284861872200
PROCLR")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port PR_RI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell PROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 311691258600
PROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_DO
            (direction OUTPUT))
          (port PR_TI
            (direction INPUT)))
      ))

    (cell OR_JOIN_L
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 233118481900
OR_JOIN_L")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port OJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_LI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell DCNR
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 22890883290
DCNR")
```

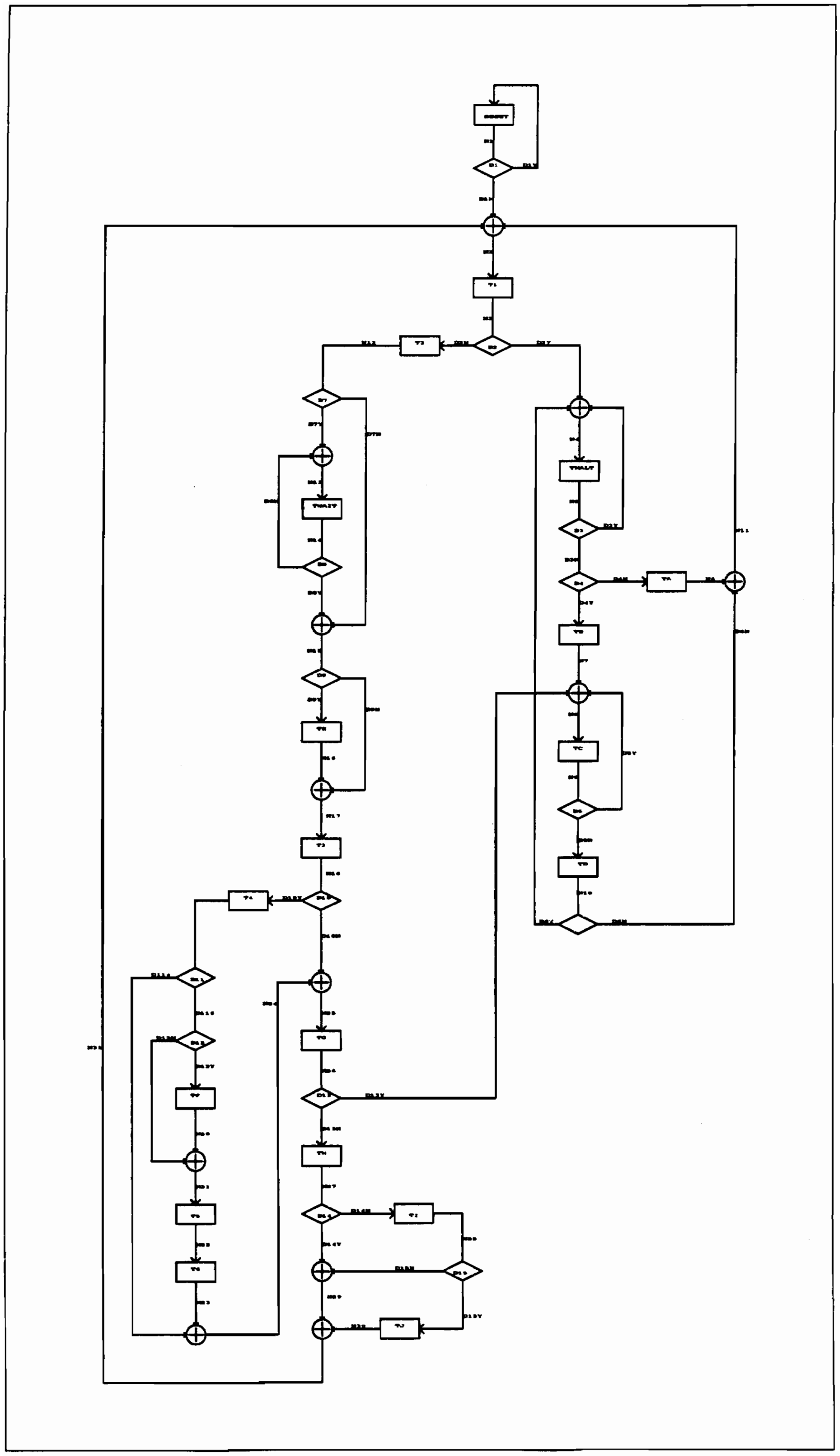
```
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_RO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell OR_JOIN_R
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 54294120690
OR_JOIN_R")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port OJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_RI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell EPROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 128442474600
EPROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_EI
            (direction INPUT)))
      ))

    (cell EG5
      (cellType GENERIC)
        (status (written
         (timeStamp 1994 6 16 20 19 6)))
      (userData VL_PLATFORM_CODE (string "K 335648411000
eg5")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface)
        (contents
          (instance P1
            (viewRef view_1 (cellRef SPROC))
            (property LABEL (string "P1") (owner
"Viewlogic_Systems")))
          (instance D1
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D1") (owner
"Viewlogic_Systems")))
          (instance P5
            (viewRef view_1 (cellRef PROCLR))
            (property LABEL (string "P5") (owner
"Viewlogic_Systems")))
          (instance P2
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "P2") (owner
"Viewlogic_Systems")))
          (instance OJ2
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ2") (owner
"Viewlogic_Systems")))
          (instance D2
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D2") (owner
"Viewlogic_Systems")))
          (instance P8
            (viewRef view_1 (cellRef PROCLR))
            (property LABEL (string "P8") (owner
"Viewlogic_Systems")))
          (instance P7
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "P7") (owner
"Viewlogic_Systems")))
          (instance P6
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "P6") (owner
"Viewlogic_Systems")))
          (instance OJ3
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ3") (owner
"Viewlogic_Systems")))
          (instance P3
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "P3") (owner
"Viewlogic_Systems")))
```

```
          (instance D3
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D3") (owner
"Viewlogic_Systems")))
          (instance OJ1
            (viewRef view_1 (cellRef OR_JOIN_R))
            (property LABEL (string "OJ1") (owner
"Viewlogic_Systems")))
          (instance P4
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "P4") (owner
"Viewlogic_Systems")))
          (instance P9
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "P9") (owner
"Viewlogic_Systems")))
          (instance P10
            (viewRef view_1 (cellRef EPROC))
            (property LABEL (string "P10") (owner
"Viewlogic_Systems")))
          (net &1
            (joined
              (portRef PR_SO (instanceRef P1))
              (portRef OJ_TI (instanceRef OJ1))))
          (net &10
            (joined
              (portRef DC_TI (instanceRef D2))
              (portRef PR_DO (instanceRef P7))))
          (net &11
            (joined
              (portRef OJ_LI (instanceRef OJ3))
              (portRef PR_LO (instanceRef P8))))
          (net &12
            (joined
              (portRef OJ_RI (instanceRef OJ1))
              (portRef PR_TI (instanceRef P9))))
          (net &2
            (joined
              (portRef PR_TI (instanceRef P2))
              (portRef OJ_DO (instanceRef OJ1))))
          (net &3
            (joined
              (portRef OJ_TI (instanceRef OJ2))
              (portRef PR_DO (instanceRef P2))))
          (net &4
            (joined
              (portRef OJ_DO (instanceRef OJ2))
              (portRef PR_TI (instanceRef P3))))
          (net &5
            (joined
              (portRef PR_TI (instanceRef P4))
              (portRef PR_DO (instanceRef P3))))
          (net &6
            (joined
              (portRef DC_TI (instanceRef D1))
              (portRef PR_DO (instanceRef P4))))
          (net &7
            (joined
              (portRef OJ_LI (instanceRef OJ2))
              (portRef PR_LO (instanceRef P5))))
          (net &8
            (joined
              (portRef PR_TI (instanceRef P6))
              (portRef OJ_DO (instanceRef OJ3))))
          (net &9
            (joined
              (portRef PR_DO (instanceRef P6))
              (portRef PR_TI (instanceRef P7))))
          (net D1N
            (joined
              (portRef DC_LO (instanceRef D1))
              (portRef PR_RI (instanceRef P5))))
          (net D1Y
            (joined
              (portRef OJ_TI (instanceRef OJ3))
              (portRef DC_DO (instanceRef D1))))
          (net D2N
            (joined
              (portRef DC_LO (instanceRef D2))
              (portRef PR_RI (instanceRef P8))))
          (net D2Y
            (joined
              (portRef DC_DO (instanceRef D2))
              (portRef DC_TI (instanceRef D3))))
          (net D3N
            (joined
              (portRef DC_RO (instanceRef D3))
              (portRef PR_DO (instanceRef P9))))
          (net D3Y
            (joined
              (portRef DC_DO (instanceRef D3))
              (portRef PR_EI (instanceRef P10))))
        )
      ))

  )
  (design ROOT
    (cellRef EG5
      (libraryRef EG5)))
)
```

Example 5 : Conceptual graph representation of the given flowchart

```
[ 1 : process : P1 ]
     ->( succ_if )
       1 ->[ 1 : process : P1 ]
       2 ->[ condition :[ CHECK ]->(=)->[ N ]]
     ->( succ_if )
       1 ->[ 5 : process : P2 ]
       2 ->[ condition :[ CHECK ]->(=)->[ Y ]]
     ->( pred_if )
       1 ->[ 1 : process : P1 ]
       2 ->[ condition :[ CHECK ]->(=)->[ N ]]

[ 2 : process : P7 ]
     ->( succ_if )
       1 ->[ 3 : process : P8 ]
       2 ->[ condition :[ D2 ]->(=)->[ D2N ]]
     ->( succ_if )
       1 ->[ 4 : process : P9 ]
       2 ->[ condition :[D2]->(=)->[D2Y] [D3]->(=)->[D3N]]
     ->( succ_if )
       1 ->[ 10 : process : P10 ]
       2 ->[ condition :[D2]->(=)->[D2Y] [D3]->(=)->[D3Y]]
     ->( pred ) -> [ 9 ]

[ 3 : process : P8 ]
     ->( succ ) -> [ 9 ]
     ->( pred_if )
       1 ->[ 2 : process : P7 ]
       2 ->[ condition :[ D2 ]->(=)->[ D2N ]]

[ 4 : process : P9 ]
     ->( succ ) -> [ 5 ]
     ->( pred_if )
       1 ->[ 2 : process : P7 ]
       2 ->[ condition :[D3]->(=)->[D3N]  [D2]->(=)->[D2Y]]

[ 5 : process : P2 ]
     ->( succ ) -> [ 6 ]
     ->( pred ) -> [ 4 ]
     ->( pred_if )
       1 ->[ 1 : process : P1 ]
       2 ->[ condition :[ CHECK ]->(=)->[ Y ]]
[ 6 : process : P3 ]
     ->( succ ) -> [ 7 ]
     ->( pred ) -> [ process ] -> (or)
             -> [ 5 : process : P2 ]
             -> [ 8 : process : P5 ]
```

```
[ 7 : process : P4 ]
     ->( succ_if )
       1 ->[ 8 : process : P5 ]
       2 ->[ condition :[ D1 ]->(=)->[ D1N ]]
     ->( succ_if )
       1 ->[ 9 : process : P6 ]
       2 ->[ condition :[ D1 ]->(=)->[ D1Y ]]
     ->( pred ) -> [ 6 ]

[ 8 : process : P5 ]
     ->( succ ) -> [ 6 ]
     ->( pred_if )
       1 ->[ 7 : process : P4 ]
       2 ->[ condition :[ D1 ]->(=)->[ D1N ]]

[ 9 : process : P6 ]
     ->( succ ) -> [ 2 ]
     ->( pred ) -> [ 3 ]
     ->( pred_if )
       1 ->[ 7 : process : P4 ]
       2 ->[ condition :[ D1 ]->(=)->[ D1Y ]]

[ 10 : process : P10 ]
     ->( pred_if )
       1 ->[ 2 : process : P7 ]
       2 ->[ condition :[D3]->(=)->[D3Y]  [D2]->(=)->[D2Y]]
```



Example 6 : EDIF netlist view of the given flowchart

```
(edif final
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordMap (keywordLevel 0))
  (status
    (written
      (timeStamp 1994 6 23 0 39 24)
      (program "VIEWlogic's edifnet" (version "4.20"))))
  (library FINAL
    (edifLevel 0)
    (technology (numberDefinition (scale 1 (E 1 -12) (unit
CAPACITANCE))))

    (cell PROC
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 311691258600
PROC")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_DO
            (direction OUTPUT))
          (port PR_TI
            (direction INPUT)))
      ))

    (cell PROCRL
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 294888880200
PROCRL")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port PR_LI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port PR_RO
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell DCNL
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 273091207900
DCNL")
```

```
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell OR_JOIN_L
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 233118481900
OR_JOIN_L")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port OJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_LI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell DCNR
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 22890883290
DCNR")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_RO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
```

```
      ))

    (cell OR_JOIN_R
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 54294120690
OR_JOIN_R")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port OJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_RI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell OR_JOIN_2
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 607029470
OR_JOIN_2")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port OJ_DO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_LI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_RI
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port OJ_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell DCN2
      (cellType GENERIC)
      (userData VL_PLATFORM_CODE (string "K 22890886490
DCN2")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface
          (port DC_LO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_RO
            (property PIN (string "") (owner
"Viewlogic_Systems")))
          (port DC_TI
            (property PIN (string "") (owner
"Viewlogic_Systems"))))
      ))

    (cell FINAL
      (cellType GENERIC)
        (status (written
         (timeStamp 1994 6 23 0 38 58)))
      (userData VL_PLATFORM_CODE (string "K 236280124500
final")
        (owner "Viewlogic_Systems"))
      (view view_1
        (viewType NETLIST)
        (interface)
        (contents
          (instance RESET
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "RESET") (owner
"Viewlogic_Systems")))
          (instance T4
            (viewRef view_1 (cellRef PROCRL))
            (property LABEL (string "T4") (owner
"Viewlogic_Systems")))
          (instance D11
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D11") (owner
"Viewlogic_Systems")))
          (instance D12
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D12") (owner
"Viewlogic_Systems")))
          (instance TF
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TF") (owner
"Viewlogic_Systems")))
          (instance OJ8
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ8") (owner
"Viewlogic_Systems")))
          (instance TA
            (viewRef view_1 (cellRef PROCRL))
            (property LABEL (string "TA") (owner
"Viewlogic_Systems")))
          (instance T5
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "T5") (owner
"Viewlogic_Systems")))
          (instance T6
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "T6") (owner
"Viewlogic_Systems")))
          (instance OJ10
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ10") (owner
"Viewlogic_Systems")))
          (instance OJ9
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ9") (owner
"Viewlogic_Systems")))
          (instance TG
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TG") (owner
"Viewlogic_Systems")))
          (instance D13
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D13") (owner
"Viewlogic_Systems")))
          (instance TH
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TH") (owner
"Viewlogic_Systems")))
          (instance D14
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D14") (owner
"Viewlogic_Systems")))
          (instance OJ11
            (viewRef view_1 (cellRef OR_JOIN_R))
            (property LABEL (string "OJ11") (owner
"Viewlogic_Systems")))
          (instance OJ12
            (viewRef view_1 (cellRef OR_JOIN_R))
            (property LABEL (string "OJ12") (owner
"Viewlogic_Systems")))
          (instance OJ1
            (viewRef view_1 (cellRef OR_JOIN_2))
            (property LABEL (string "OJ1") (owner
"Viewlogic_Systems")))
          (instance TI
            (viewRef view_1 (cellRef PROCRL))
            (property LABEL (string "TI") (owner
"Viewlogic_Systems")))
          (instance D15
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D15") (owner
"Viewlogic_Systems")))
```

```
          (instance TJ
            (viewRef view_1 (cellRef PROCRL))
            (property LABEL (string "TJ") (owner
"Viewlogic_Systems")))
          (instance OJ4
            (viewRef view_1 (cellRef OR_JOIN_R))
            (property LABEL (string "OJ4") (owner
"Viewlogic_Systems")))
          (instance TB
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TB") (owner
"Viewlogic_Systems")))
          (instance OJ3
            (viewRef view_1 (cellRef OR_JOIN_2))
            (property LABEL (string "OJ3") (owner
"Viewlogic_Systems")))
          (instance TC
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TC") (owner
"Viewlogic_Systems")))
          (instance D5
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D5") (owner
"Viewlogic_Systems")))
          (instance TD
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TD") (owner
"Viewlogic_Systems")))
          (instance D1
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D1") (owner
"Viewlogic_Systems")))
          (instance D6
            (viewRef view_1 (cellRef DCN2))
            (property LABEL (string "D6") (owner
"Viewlogic_Systems")))
          (instance T1
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "T1") (owner
"Viewlogic_Systems")))
          (instance D2
            (viewRef view_1 (cellRef DCN2))
            (property LABEL (string "D2") (owner
"Viewlogic_Systems")))
          (instance OJ2
            (viewRef view_1 (cellRef OR_JOIN_2))
            (property LABEL (string "OJ2") (owner
"Viewlogic_Systems")))
          (instance T2
            (viewRef view_1 (cellRef PROCRL))
            (property LABEL (string "T2") (owner
"Viewlogic_Systems")))
          (instance D7
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D7") (owner
"Viewlogic_Systems")))
          (instance OJ5
            (viewRef view_1 (cellRef OR_JOIN_L))
            (property LABEL (string "OJ5") (owner
"Viewlogic_Systems")))
          (instance TWAIT
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TWAIT") (owner
"Viewlogic_Systems")))
          (instance THALT
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "THALT") (owner
"Viewlogic_Systems")))
          (instance D8
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D8") (owner
"Viewlogic_Systems")))
          (instance OJ6
            (viewRef view_1 (cellRef OR_JOIN_R))
            (property LABEL (string "OJ6") (owner
"Viewlogic_Systems")))
          (instance D9
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D9") (owner
"Viewlogic_Systems")))
          (instance TE
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "TE") (owner
"Viewlogic_Systems")))
          (instance OJ7
            (viewRef view_1 (cellRef OR_JOIN_R))
            (property LABEL (string "OJ7") (owner
"Viewlogic_Systems")))
          (instance T3
            (viewRef view_1 (cellRef PROC))
            (property LABEL (string "T3") (owner
"Viewlogic_Systems")))
          (instance D10
            (viewRef view_1 (cellRef DCNL))
            (property LABEL (string "D10") (owner
"Viewlogic_Systems")))
          (instance D3
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D3") (owner
"Viewlogic_Systems")))
          (instance D4
            (viewRef view_1 (cellRef DCNR))
            (property LABEL (string "D4") (owner
"Viewlogic_Systems")))
          (net D10N
            (joined
              (portRef DC_DO (instanceRef D10))
              (portRef OJ_TI (instanceRef OJ9))))
          (net D10Y
            (joined
              (portRef DC_LO (instanceRef D10))
              (portRef PR_LI (instanceRef T4))))
          (net D114
            (joined
              (portRef DC_LO (instanceRef D11))
              (portRef OJ_TI (instanceRef OJ10))))
          (net D116
            (joined
              (portRef DC_DO (instanceRef D11))
              (portRef DC_TI (instanceRef D12))))
          (net D12N
            (joined
              (portRef DC_LO (instanceRef D12))
              (portRef OJ_LI (instanceRef OJ8))))
          (net D12Y
            (joined
              (portRef DC_DO (instanceRef D12))
              (portRef PR_TI (instanceRef TF))))
          (net D13N
            (joined
              (portRef DC_DO (instanceRef D13))
              (portRef PR_TI (instanceRef TH))))
          (net D13Y
            (joined
              (portRef OJ_LI (instanceRef OJ3))
              (portRef DC_RO (instanceRef D13))))
          (net D14N
            (joined
              (portRef DC_RO (instanceRef D14))
              (portRef PR_LI (instanceRef TI))))
          (net D14Y
            (joined
              (portRef DC_DO (instanceRef D14))
              (portRef OJ_TI (instanceRef OJ11))))
          (net D15N
            (joined
              (portRef OJ_RI (instanceRef OJ11))
              (portRef DC_LO (instanceRef D15))))
          (net D15Y
            (joined
              (portRef PR_LI (instanceRef TJ))
              (portRef DC_DO (instanceRef D15))))
```

```
(net D1N
  (joined
    (portRef OJ_TI (instanceRef OJ1))
    (portRef DC_DO (instanceRef D1))))
(net D1Y
  (joined
    (portRef DC_RO (instanceRef D1))
    (portRef PR_TI (instanceRef RESET))))
(net D2N
  (joined
    (portRef PR_LI (instanceRef T2))
    (portRef DC_LO (instanceRef D2))))
(net D2Y
  (joined
    (portRef OJ_TI (instanceRef OJ2))
    (portRef DC_RO (instanceRef D2))))
(net D3N
  (joined
    (portRef DC_TI (instanceRef D4))
    (portRef DC_DO (instanceRef D3))))
(net D3Y
  (joined
    (portRef OJ_RI (instanceRef OJ2))
    (portRef DC_RO (instanceRef D3))))
(net D4N
  (joined
    (portRef PR_LI (instanceRef TA))
    (portRef DC_RO (instanceRef D4))))
(net D4Y
  (joined
    (portRef PR_TI (instanceRef TB))
    (portRef DC_DO (instanceRef D4))))
(net D5N
  (joined
    (portRef PR_TI (instanceRef TD))
    (portRef DC_DO (instanceRef D5))))
(net D5Y
  (joined
    (portRef OJ_RI (instanceRef OJ3))
    (portRef DC_RO (instanceRef D5))))
(net D6N
  (joined
    (portRef DC_RO (instanceRef D6))
    (portRef OJ_TI (instanceRef OJ4))))
(net D6Y
  (joined
    (portRef DC_LO (instanceRef D6))
    (portRef OJ_LI (instanceRef OJ2))))
(net D7N
  (joined
    (portRef DC_RO (instanceRef D7))
    (portRef OJ_RI (instanceRef OJ6))))
(net D7Y
  (joined
    (portRef DC_DO (instanceRef D7))
    (portRef OJ_TI (instanceRef OJ5))))
(net D8N
  (joined
    (portRef OJ_LI (instanceRef OJ5))
    (portRef DC_LO (instanceRef D8))))
(net D8Y
  (joined
    (portRef DC_DO (instanceRef D8))
    (portRef OJ_TI (instanceRef OJ6))))
(net D9N
  (joined
    (portRef DC_RO (instanceRef D9))
    (portRef OJ_RI (instanceRef OJ7))))
(net D9Y
  (joined
    (portRef DC_DO (instanceRef D9))
    (portRef PR_TI (instanceRef TE))))
(net N1
  (joined
    (portRef DC_TI (instanceRef D1))
    (portRef PR_DO (instanceRef RESET))))
(net N10
  (joined
    (portRef DC_TI (instanceRef D6))
    (portRef PR_DO (instanceRef TD))))
(net N11
  (joined
    (portRef OJ_RI (instanceRef OJ1))
    (portRef OJ_DO (instanceRef OJ4))))
(net N12
  (joined
    (portRef PR_RO (instanceRef T2))
    (portRef DC_TI (instanceRef D7))))
(net N13
  (joined
    (portRef OJ_DO (instanceRef OJ5))
    (portRef PR_TI (instanceRef TWAIT))))
(net N14
  (joined
    (portRef PR_DO (instanceRef TWAIT))
    (portRef DC_TI (instanceRef D8))))
(net N15
  (joined
    (portRef OJ_DO (instanceRef OJ6))
    (portRef DC_TI (instanceRef D9))))
(net N16
  (joined
    (portRef PR_DO (instanceRef TE))
    (portRef OJ_TI (instanceRef OJ7))))
(net N17
  (joined
    (portRef OJ_DO (instanceRef OJ7))
    (portRef PR_TI (instanceRef T3))))
(net N18
  (joined
    (portRef PR_DO (instanceRef T3))
    (portRef DC_TI (instanceRef D10))))
(net N19
  (joined
    (portRef PR_RO (instanceRef T4))
    (portRef DC_TI (instanceRef D11))))
(net N2
  (joined
    (portRef PR_TI (instanceRef T1))
    (portRef OJ_DO (instanceRef OJ1))))
(net N20
  (joined
    (portRef PR_DO (instanceRef TF))
    (portRef OJ_TI (instanceRef OJ8))))
(net N21
  (joined
    (portRef OJ_DO (instanceRef OJ8))
    (portRef PR_TI (instanceRef T5))))
(net N22
  (joined
    (portRef PR_DO (instanceRef T5))
    (portRef PR_TI (instanceRef T6))))
(net N23
  (joined
    (portRef PR_DO (instanceRef T6))
    (portRef OJ_LI (instanceRef OJ10))))
(net N24
  (joined
    (portRef OJ_LI (instanceRef OJ9))
    (portRef OJ_DO (instanceRef OJ10))))
(net N25
  (joined
    (portRef OJ_DO (instanceRef OJ9))
    (portRef PR_TI (instanceRef TG))))
(net N26
  (joined
    (portRef PR_DO (instanceRef TG))
    (portRef DC_TI (instanceRef D13))))
(net N27
  (joined
    (portRef PR_DO (instanceRef TH))
    (portRef DC_TI (instanceRef D14))))
(net N28
            (joined
              (portRef PR_RO (instanceRef TI))
              (portRef DC_TI (instanceRef D15))))
          (net N29
            (joined
              (portRef OJ_DO (instanceRef OJ11))
              (portRef OJ_TI (instanceRef OJ12))))
          (net N3
            (joined
              (portRef DC_TI (instanceRef D2))
              (portRef PR_DO (instanceRef T1))))
          (net N30
            (joined
              (portRef OJ_RI (instanceRef OJ12))
              (portRef PR_RO (instanceRef TJ))))
          (net N31
            (joined
              (portRef OJ_LI (instanceRef OJ1))
              (portRef OJ_DO (instanceRef OJ12))))
          (net N4
            (joined
              (portRef PR_TI (instanceRef THALT))
              (portRef OJ_DO (instanceRef OJ2))))
          (net N5
            (joined
              (portRef PR_DO (instanceRef THALT))
              (portRef DC_TI (instanceRef D3))))
          (net N6
            (joined
              (portRef PR_RO (instanceRef TA))
              (portRef OJ_RI (instanceRef OJ4))))
          (net N7
            (joined
              (portRef OJ_TI (instanceRef OJ3))
              (portRef PR_DO (instanceRef TB))))
          (net N8
            (joined
              (portRef OJ_DO (instanceRef OJ3))
              (portRef PR_TI (instanceRef TC))))
          (net N9
            (joined
              (portRef PR_DO (instanceRef TC))
              (portRef DC_TI (instanceRef D5))))
        )
      ))

  )
  (design ROOT
    (cellRef FINAL
      (libraryRef FINAL)))
)
```

Example 6 : Conceptual graph representation of the given flowchart

```
[ 1 : process : T4 ]
     ->( succ_if )
       1 ->[ 17 : process : TG ]
       2 ->[ condition :[ D11 ]->(=)->[ D114 ]]
     ->( succ_if )
       1 ->[ 15 : process : T5 ]
       2 ->[condition:[D11]->(=)->[D116]  [D12]->(=)->[D12N]]
     ->( succ_if )
       1 ->[ 2 : process : TF ]
       2 ->[ condition :[D11]->(=)->[D116] [D12]->(=)->
           [D12Y]]
     ->( pred_if )
       1 ->[ 13 : process : T3 ]
       2 ->[ condition :[ D10 ]->(=)->[ D10Y ]]

[ 2 : process : TF ]
     ->( succ ) -> [ 15 ]
     ->( pred_if )
       1 ->[ 1 : process : T4 ]
       2 ->[ condition :[ D12 ]->(=)->[ D12Y ]  [D11]->(=)->
           [ D116 ]]

[ 3 : process : TH ]
     ->( succ_if )
       1 ->[ 4 : process : TI ]
       2 ->[ condition :[ D14 ]->(=)->[ D14N ]]
     ->( succ_if )
       1 ->[ 14 : process : T1 ]
       2 ->[ condition :[ D14 ]->(=)->[ D14Y ]]
     ->( pred_if )
       1 ->[ 17 : process : TG ]
       2 ->[ condition :[ D13 ]->(=)->[ D13N ]]

[ 4 : process : TI ]
     ->( succ_if )
       1 ->[ 14 : process : T1 ]
       2 ->[ condition :[ D15 ]->(=)->[ D15N ]]
     ->( succ_if )
       1 ->[ 5 : process : TJ ]
       2 ->[ condition :[ D15 ]->(=)->[ D15Y ]]
     ->( pred_if )
       1 ->[ 3 : process : TH ]
       2 ->[ condition :[ D14 ]->(=)->[ D14N ]]

[ 5 : process : TJ ]
     ->( succ ) -> [ 14 ]
     ->( pred_if )
```

```
        1 ->[ 4 : process : TI ]
        2 ->[ condition :[ D15 ]->(=)->[ D15Y ]]

[ 6 : process : RESET ]
     ->( succ_if )
        1 ->[ 14 : process : T1 ]
        2 ->[ condition :[ D1 ]->(=)->[ D1N ]]
     ->( succ_if )
        1 ->[ 6 : process : RESET ]
        2 ->[ condition :[ D1 ]->(=)->[ D1Y ]]
     ->( pred_if )
        1 ->[ 6 : process : RESET ]
        2 ->[ condition :[ D1 ]->(=)->[ D1Y ]]

[ 7 : process : T2 ]
     ->( succ_if )
        1 ->[ 13 : process : T3 ]
        2 ->[ condition :[ D7 ]->(=)->[ D7N ]  [ D9 ]->(=)->
            [ D9N ]]
     ->( succ_if )
        1 ->[ 11 : process : TE ]
        2 ->[ condition :[ D7 ]->(=)->[ D7N ][ D9 ]->(=)->
            [ D9Y ]]
     ->( succ_if )
        1 ->[ 12 : process : TWAIT ]
        2 ->[ condition :[ D7 ]->(=)->[ D7Y ]]
     ->( pred_if )
        1 ->[ 14 : process : T1 ]
        2 ->[ condition :[ D2 ]->(=)->[ D2N ]]

[ 8 : process : TA ]
     ->( succ ) -> [ 14 ]
     ->( pred_if )
        1 ->[ 18 : process : THALT ]
        2 ->[ condition :[ D4 ]->(=)->[ D4N ]  [ D3 ]->(=)->
            [ D3N ]]

[ 9 : process : TB ]
     ->( succ ) -> [ 19 ]
     ->( pred_if )
        1 ->[ 18 : process : THALT ]
        2 ->[ condition :[ D4 ]->(=)->[ D4Y ]  [ D3 ]->(=)->
            [ D3N ]]

[ 10 : process : TD ]
     ->( succ_if )
        1 ->[ 14 : process : T1 ]
        2 ->[ condition :[ D6 ]->(=)->[ D6N ]]
     ->( succ_if )
        1 ->[ 18 : process : THALT ]
        2 ->[ condition :[ D6 ]->(=)->[ D6Y ]]
     ->( pred_if )
       1 ->[ 19 : process : TC ]
       2 ->[ condition :[ D5 ]->(=)->[ D5N ]]

[ 11 : process : TE ]
     ->( succ ) -> [ 13 ]
     ->( pred_if )
       1 ->[ 7 : process : T2 ]
       2 ->[ condition :[ D9 ]->(=)->[ D9Y ] [ D7 ]->(=)->
          [ D7N ]]
     ->( pred_if )
       1 ->[ 12 : process : TWAIT ]
       2 ->[ condition :[ D9 ]->(=)->[ D9Y ] [ D8 ]->(=)->
          [ D8Y ]]

[ 12 : process : TWAIT ]
     ->( succ_if )
       1 ->[ 12 : process : TWAIT ]
       2 ->[ condition :[ D8 ]->(=)->[ D8N ]]
     ->( succ_if )
       1 ->[ 13 : process : T3 ]
       2 ->[ condition :[ D8 ]->(=)->[ D8Y ] [ D9 ]->(=)->
          [ D9N ]]
     ->( succ_if )
       1 ->[ 11 : process : TE ]
       2 ->[ condition :[ D8 ]->(=)->[ D8Y ][ D9 ]->(=)->
          [ D9Y ]]
     ->( pred_if )
       1 ->[ 7 : process : T2 ]
       2 ->[ condition :[ D7 ]->(=)->[ D7Y ]]
     ->( pred_if )
       1 ->[ 12 : process : TWAIT ]
       2 ->[ condition :[ D8 ]->(=)->[ D8N ]]

[ 13 : process : T3 ]
     ->( succ_if )
       1 ->[ 17 : process : TG ]
       2 ->[ condition :[ D10 ]->(=)->[ D10N ]]
     ->( succ_if )
       1 ->[ 1 : process : T4 ]
       2 ->[ condition :[ D10 ]->(=)->[ D10Y ]]
     ->( pred_if )
       1 ->[ 7 : process : T2 ]
       2 ->[ condition :[ D9 ]->(=)->[ D9N ] [ D7 ]->(=)->
          [ D7N ]]
     ->( pred_if )
       1 ->[ 12 : process : TWAIT ]
       2 ->[ condition :[ D9 ]->(=)->[ D9N ] [ D8 ]->(=)->
          [ D8Y ]]
     ->( pred ) -> [ 11 ]
```

```
[ 14 : process : T1 ]
    ->( succ_if )
      1 ->[ 7 : process : T2 ]
      2 ->[ condition :[ D2 ]->(=)->[ D2N ]]
    ->( succ_if )
      1 ->[ 18 : process : THALT ]
      2 ->[ condition :[ D2 ]->(=)->[ D2Y ]]
    ->( pred ) -> [ process ] -> (or)
                -> [ 5 : process : TJ ]
                -> [ 8 : process : TA ]
    ->( pred_if )
      1 ->[ 6 : process : RESET ]
      2 ->[ condition :[ D1 ]->(=)->[ D1N ]]
    ->( pred_if )
      1 ->[ 10 : process : TD ]
      2 ->[ condition :[ D6 ]->(=)->[ D6N ]]
    ->( pred_if )
      1 ->[ 3 : process : TH ]
      2 ->[ condition :[ D14 ]->(=)->[ D14Y ]]
    ->( pred_if )
      1 ->[ 4 : process : TI ]
      2 ->[ condition :[ D15 ]->(=)->[ D15N ]]

[ 15 : process : T5 ]
    ->( succ ) -> [ 16 ]
    ->( pred_if )
      1 ->[ 1 : process : T4 ]
      2 ->[ condition :[D12]->(=)->[ D12N ]  [ D11 ]->(=)->
         [ D116 ]]
    ->( pred ) -> [ 2 ]

[ 16 : process : T6 ]
    ->( succ ) -> [ 17 ]
    ->( pred ) -> [ 15 ]

[ 17 : process : TG ]
    ->( succ_if )
      1 ->[ 3 : process : TH ]
      2 ->[ condition :[ D13 ]->(=)->[ D13N ]]
    ->( succ_if )
      1 ->[ 19 : process : TC ]
      2 ->[ condition :[ D13 ]->(=)->[ D13Y ]]
    ->( pred_if )
      1 ->[ 13 : process : T3 ]
      2 ->[ condition :[ D10 ]->(=)->[ D10N ]]
    ->( pred_if )
      1 ->[ 1 : process : T4 ]
      2 ->[ condition :[ D11 ]->(=)->[ D114 ]]
    ->( pred ) -> [ 16 ]
```

```
[ 18 : process : THALT ]
     ->( succ_if )
       1 ->[ 8 : process : TA ]
       2 ->[ condition :[ D3 ]->(=)->[ D3N ][ D4 ]->(=)->
           [ D4N ]]
     ->( succ_if )
       1 ->[ 9 : process : TB ]
       2 ->[ condition :[ D3 ]->(=)->[ D3N ][ D4 ]->(=)->
           [ D4Y ]]
     ->( succ_if )
       1 ->[ 18 : process : THALT ]
       2 ->[ condition :[ D3 ]->(=)->[ D3Y ]]
     ->( pred_if )
       1 ->[ 14 : process : T1 ]
       2 ->[ condition :[ D2 ]->(=)->[ D2Y ]]
     ->( pred_if )
       1 ->[ 18 : process : THALT ]
       2 ->[ condition :[ D3 ]->(=)->[ D3Y ]]
     ->( pred_if )
       1 ->[ 10 : process : TD ]
       2 ->[ condition :[ D6 ]->(=)->[ D6Y ]]

[ 19 : process : TC ]
     ->( succ_if )
       1 ->[ 10 : process : TD ]
       2 ->[ condition :[ D5 ]->(=)->[ D5N ]]
     ->( succ_if )
       1 ->[ 19 : process : TC ]
       2 ->[ condition :[ D5 ]->(=)->[ D5Y ]]
     ->( pred_if )
       1 ->[ 17 : process : TG ]
       2 ->[ condition :[ D13 ]->(=)->[ D13Y ]]
     ->( pred_if )
       1 ->[ 19 : process : TC ]
       2 ->[ condition :[ D5 ]->(=)->[ D5Y ]]
     ->( pred ) -> [ 9 ]
```

Vita

Ramprasad Venkatasubramanian was born in Neyveli, India, on the 10th of March, 1965. He received his Bachelor of Engineering degree in Electronics Engineering from Bangalore University at Bangalore, India, in 1987. After that he began employment as a Customer Engineer with Modi Xerox, Bangalore, India.

Ramprasad joined Virginia Polytechnic Institute and State University in August, 1991. His major area of study at Virginia Tech was digital design and VLSI design areas of Computer Engineering. He graduated from Virginia Tech in June 1994.