INTERNATIONAL ATOMIC ENERGY AGENCY
UNITED NATIONS EDUCATIONAL, SCIENTIFIC AND CULTURAL ORGANIZATION
INTERNATIONAL CENTRE FOR THEORETICAL PHYSICS
I.C.T.P., P.O. BOX 586, 34100 TRIESTE, ITALY. CABLE CENTRATOM TRIESTE

The United Nations University

SMR/748 - 8

ICTP-INFN-UNU-MICROPROCESSOR LABORATORY
THIRD COURSE ON BASIC VLSI DESIGN TECHNIQUES
21 November - 16 December 1994

VHDL, ALLIANCE

Nizar ABDALLAH
Laboratoire MASI - Equipe CAO-VLSI
Universite Pierre et Marie Curie (Paris VI)
Tour 55-65, 2eme etage
4, Place Jussieu
75252 Paris Cedex
FRANCE

NIZAR ABDALLAH

LABORATOIRE MASI - EQUIPE CAO-VLSI

UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)
TOUR 55-65 2EME ETAGE
4, PLACE JUSSIEU, 75252 PARIS CEDEX 05
FRANCE

E-mail: Nizar.Abdallah@masi.ibp.fr
☎: (33-1) 44.27.30.43

OUTLINE

I - INTRODUCTION.

II - DESIGN METHODOLOGY: AN OVERVIEW.

III - ABSTRACTION LEVELS IN ALLIANCE.

IV - VHDL: AN OVERVIEW.

V - VHDL: THE ALLIANCE SUBSET.

VI - ALLIANCE: A COMPLETE DESIGN SYSTEM.

THE MASI LABORATORY

UNIVERSITY PIERRE ET MARIE CURIE
NATIONAL CENTRE OF SCIENTIFIC RESEARCH

168 RESEARCHERS

- ARCHITECTURE 59
- DISTRIBUTED SYSTEMS 36
- NETWORKS & PERFORMANCES 30
- PARALLEL ALGORITHMS 17

THE ARCHITECTURE GROUP

CAD FOR VLSI		ARCHITECTURE	
PORTABLE LIBRARIES	9	SUPERSCALAR PROCESSOR	5
VERIFICATION	7	RCUBE ROUTER	8
LOGIC SYNTHESIS	5	RAPID COPROCESSOR	6
ARCHITECTURE SYNTHESIS	4		
TEST	5		

EDUCATION TARGET

- UNDERGRADUATE STUDENTS: (≈ 80 STUDENTS AND 72 HOURS)
  - ELECTRICAL ENGINEERING
  - COMPUTER SCIENCE
- POSTGRADUATE STUDENTS (≈ 60 STUDENTS AND 300 HOURS)
  - DEA MEMI
  - DESS CIMI

The ALLIANCE System

- A Complete Set of CAD Tools for Digital CMOS VLSI Design.
- Proposes a Design Methodology.
- Portable, Compact and Easy to Learn.
- ALLIANCE is Totally Free.

OUTLINE

I - INTRODUCTION.

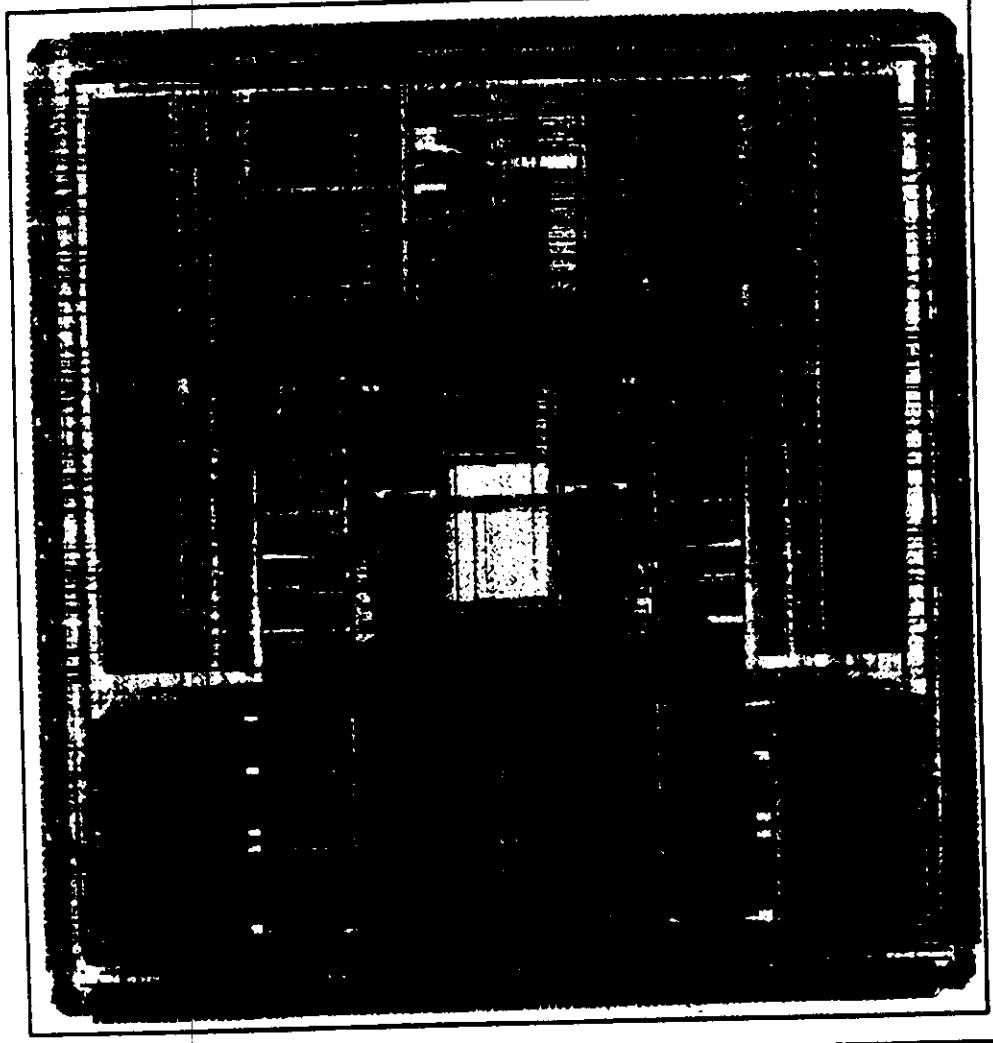
II - DESIGN METHODOLOGY: AN OVERVIEW.

III - ABSTRACTION LEVELS IN ALLIANCE.

IV - VHDL: AN OVERVIEW.

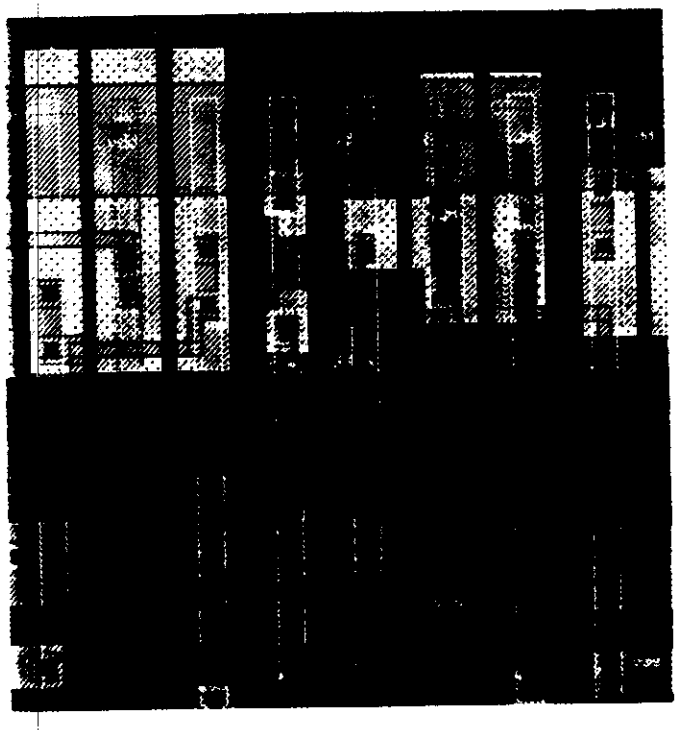
V - VHDL: THE ALLIANCE SUBSET.

VI - ALLIANCE: A COMPLETE DESIGN SYSTEM.



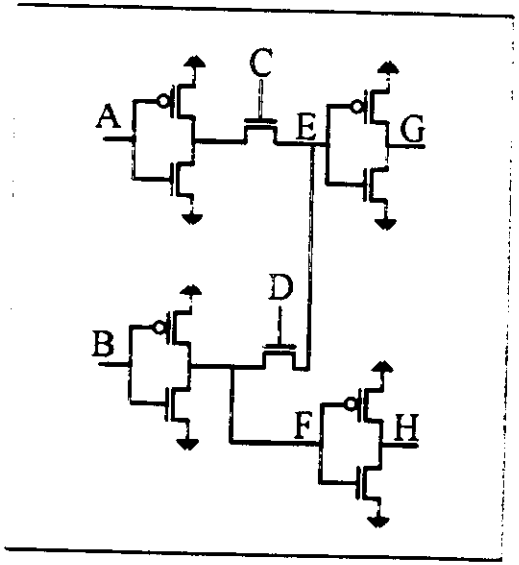
DESIGNER'S DREAM

MILLIONS OF SEGMENTS PUT TOGETHER.



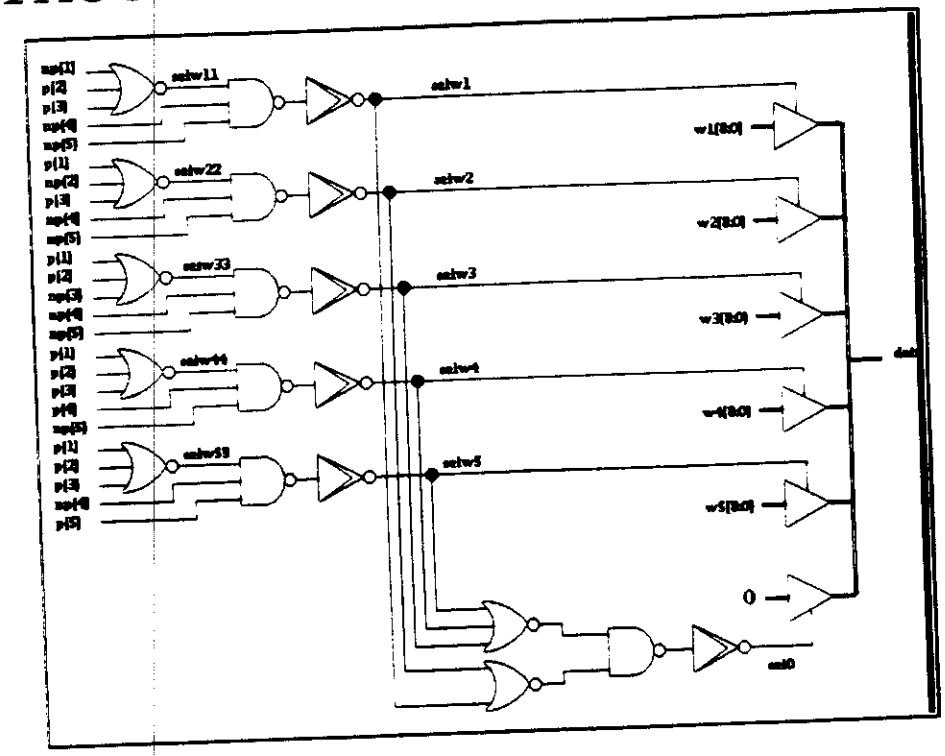
HOW TO DEAL WITH SUCH COMPLEXITY ?

ONE MILLION OF TRANSISTORS CONNECTED TOGETHER.



- **STILL TOO COMPLEX.....!!!**

HUNDRED THOUSAND OF CELLS CONNECTED TOGETHER.



× STILL TOO COMPLEX.....!!!

DOZEN OF FUNCTIONAL BLOCKS THAT COMMUNICATE TOGETHER.

✓ **I UNDERSTAND (OUF !!!)**

empty_IO
PHOTO_GD
PHOTO_S1
PVDD_IO
PEPS_IO
PNTS_IO
PGTS_IO
PVSS_IO
PNTL_IO
PGTL_IO
PNTL_IO
PVSS_IO
PGTL_IO
empty_IO

A SET OF EQUATIONS THAT REFLECT THE WHOLE FUNCTIONNALITY OF THE CIRCUIT.

```
entuty  adder is
port (
        a, b: Bit;
         c, d : bit
       );

architecture  adder is

a <= b or c
d <= b and c;
end;
```

✓ I UNDERSTAND WHAT THIS CIRCUIT IS SUPPOSED TO DO.

SO,

HOW TO DEAL WITH SUCH COMPLEXITY ?

- ✓ ABSTRACTION
- ✓ HIERARCHY

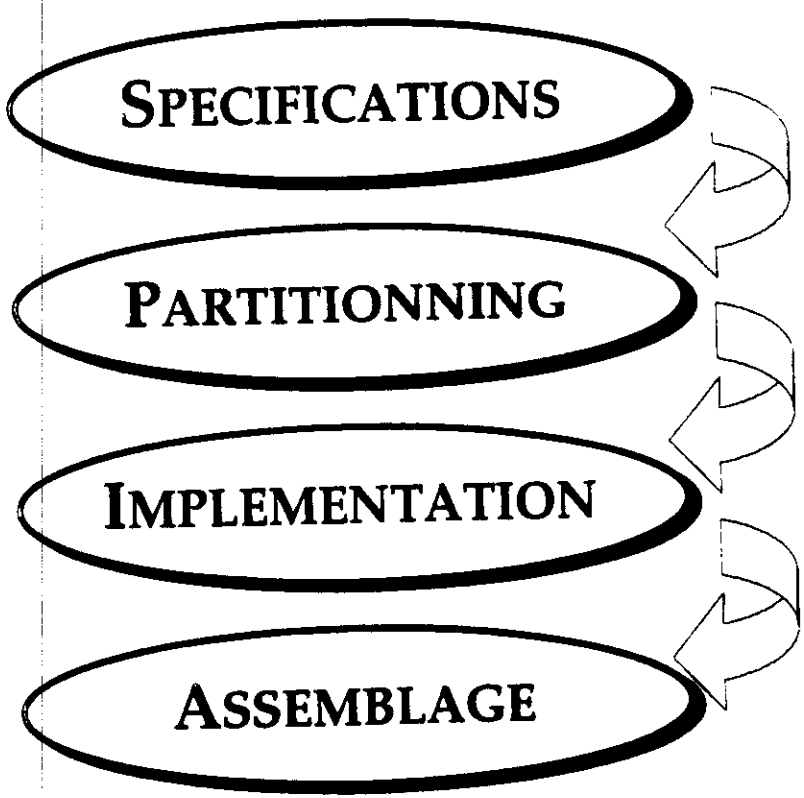
LEVELS OF ABSTRACTION

TO GO ACROSS THESE DIFFERENT LEVELS OF ABSTRACTION

I NEED

A DESIGN METHODOLOGY

DESIGN METHODOLOGY
TOP-DOWN METHODOLOGY



STEP 1: SPECIFICATIONS (1)

PUT DOWN THE CIRCUIT CONCEPT.

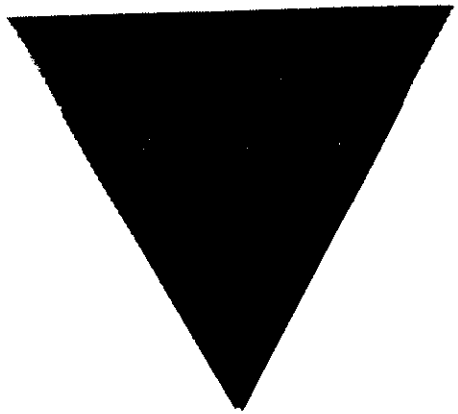
TWO REASONS:

- TO BE ABLE TO CHECK IT BEFORE MANUFACTURING.
- TO HAVE A REFERENCE MANUAL FOR COMMUNICATION.

STEP 1: SPECIFICATIONS (2)

COMMUNICATION LANGUAGE.

BETWEEN DIFFERENT PEOPLE ON THE PROJECT AND BETWEEN PEOPLE AND COMPUTERS.



- ✗ NO ORDINARY LANGUAGE.
- ✓ ACCURATE LANGUAGE.
- ✓ A LANGUAGE THAT CAN BE SIMULATED.

STEP 2: HOW TO ?(1)

VERY DIFFICULT STEP: RELAYS ON THE KNOW-HOW OF THE DESIGNER.

MAIN IDEA: TO SPLIT INTO SEVERAL SMALL PARTS.

DIVIDE AND CONQUER STRATEGY.

HIERARCHY.

STEP 2: HOW TO ? (2)

THE CUTTING IS GUIDED BY:

1. REGULARITY OR NOT.

- IDENTIFY REGULAR BLOCKS.
- IDENTIFY RANDOM LOGIC BLOCKS.

STEP 2: HOW TO ? (3)

THE CUTTING IS GUIDED BY:

2. TIMING ASPECTS.

- COARSE ESTIMATION OF TIMING.
- LOOKING FOR A GOOD BALANCE.

STEP 2: HOW TO ? (4)

THE CUTTING IS GUIDED BY:

3. TOPOLOGY.

- ALREADY IN MIND THE CIRCUIT FORM.
- AN IDEA ABOUT THE SIZE OF EACH PART.
- AN IDEA ABOUT THE ROUTING.
- OPTIMIZING SILICON AREA USAGE.

STEP 2: HOW TO ? (5)

THE CUTTING IS GUIDED BY:

4. TECHNOLOGY.

- USING ASGA OR CMOS ?
- USING PALS OR STANDARD CELLS ?

STEP 2: HOW TO ? (6)

THE CUTTING IS GUIDED BY:

5. CAD TOOLS.

- WHAT TOOLS DO I HAVE TO MAKE MY CIRCUIT ?

EX: NO SYNTHESIS TOOLS SO I TRY TO REDUCE THE RANDOM LOGIC PART.

STEP 3: IMPLEMENTATION

EACH PART WILL BE IMPLEMENTED USING A PARTICULAR METHOD. WHEN I SPLIT MY CIRCUIT, I HAVE ALREADY DECIDED WHICH ONE.

STEP 4: ASSEMBLAGE

THE ASSEMBLAGE IS DONE IN A HIERARCHICAL WAY, STARTING FROM THE LOWEST LEVEL.

CONCLUSION (1)

AT EACH STEP, THE INFORMATION IS ENHANCED:

1. FROM THE IDEA DOWN TO THE SPECIFICATIONS.

2. WHEN STRUCTURING THE MODEL IN AN OTHER WAY.

3. .....

⇒ AT EACH STEP, A VERIFICATION IS TO BE DONE.

CONCLUSION (2)

ALL ALONG THE METHODOLOGY, WE HANDLED DIFFERENT VIEWS:

1. EQUATIONS.
2. NETLISTS.
3. LAYOUT.

CONCLUSION (3)

THERE IS A METHOD.

OUTLINE

I - INTRODUCTION.

II - DESIGN METHODOLOGY: AN OVERVIEW.

III - ABSTRACTION LEVELS IN ALLIANCE.

IV - VHDL: AN OVERVIEW.

V - VHDL: THE ALLIANCE SUBSET.

VI - ALLIANCE: A COMPLETE DESIGN SYSTEM.

3 Different Views

All along the methodology, we handled different views:

1. Behavioural View (Equations).
2. Structural View (Netlists).
3. Layout (View).

BEHAVIOURAL VUE (1)

LOGICAL EQUATIONS

- DESCRIPTION FORMALISM.

A SET OF LOGICAL EQUATIONS (BOOLEAN) REPRESENTING BOOLEAN FUNCTIONS.

EXAMPLE: $U = A.(A+B)$ $V = C.D$ $T = D \oplus E$

$X = U.V$ $Y = V + T + X$ $Z = T.E$

STRUCTURAL VUE (1)

FOR ALL THESE VIEWS, WE ARE LOOKING FOR BASIC CONCEPTS: COMPLETELY INDEPENDENT FROM A GIVEN LANGUAGE.

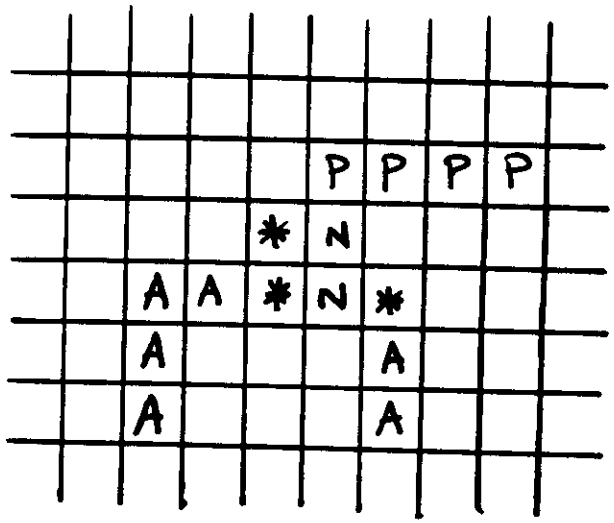
IN THE STRUCTURAL VIEW:

- CONNECTORS: ID, DIRECTION, ETC....
- SIGNALS: ID, TYPE (EXTERNAL OR NOT), ETC....
- INSTANCE: ID, MODEL NAME, PORTS, ETC....

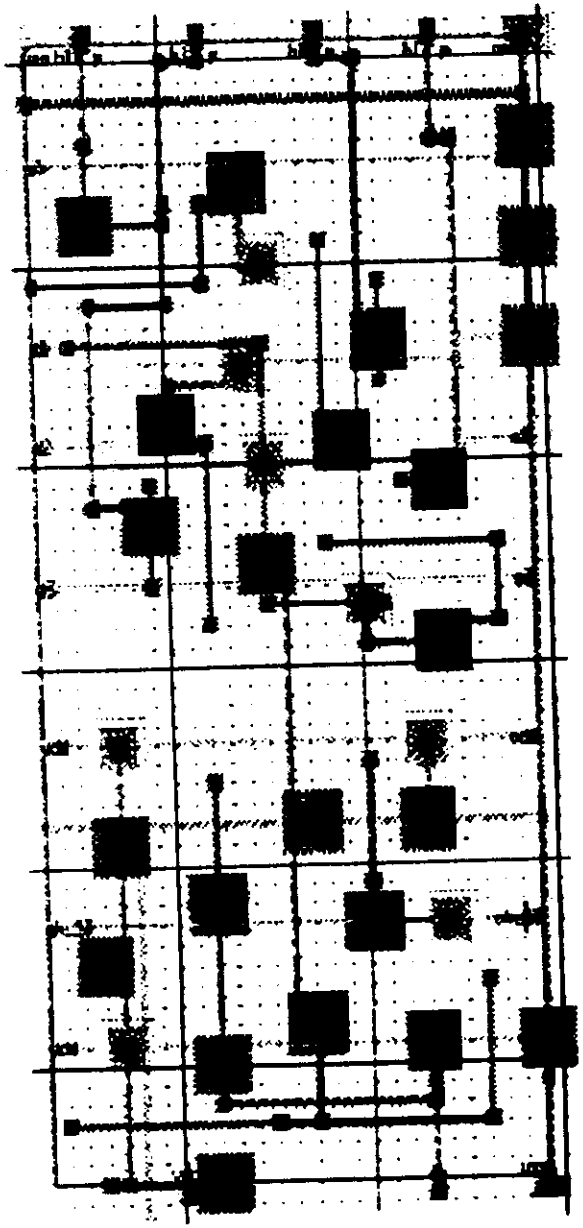
LAYOUT VUE (1)

SYMBOLIC LAYOUT: PRINCIPLES

- PORTABILITY
- SIMPLICITY
- ROBUSTNESS



COARSE GRID



LAYOUT VUE (2)

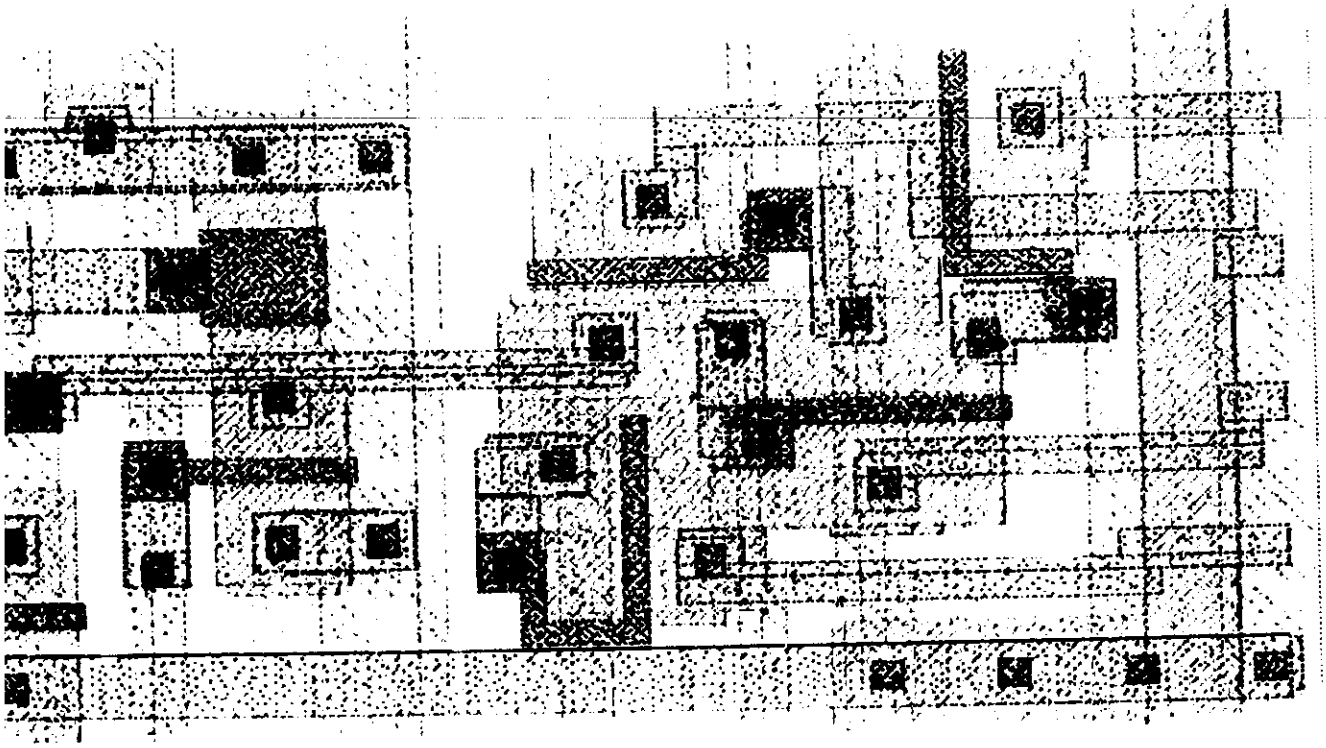
SYMBOLIC LAYOUT: OUR APPROACH

THIN FIXED GRID, SYMBOLIC LAYOUT.

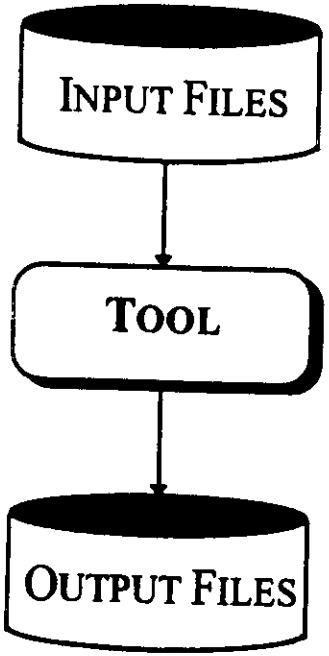
DISTANCES FORM CENTER TO CENTER ⇒ GOOD DENSITIES.

SPECIAL SYMBOLIC LAYOUT EDITOR.

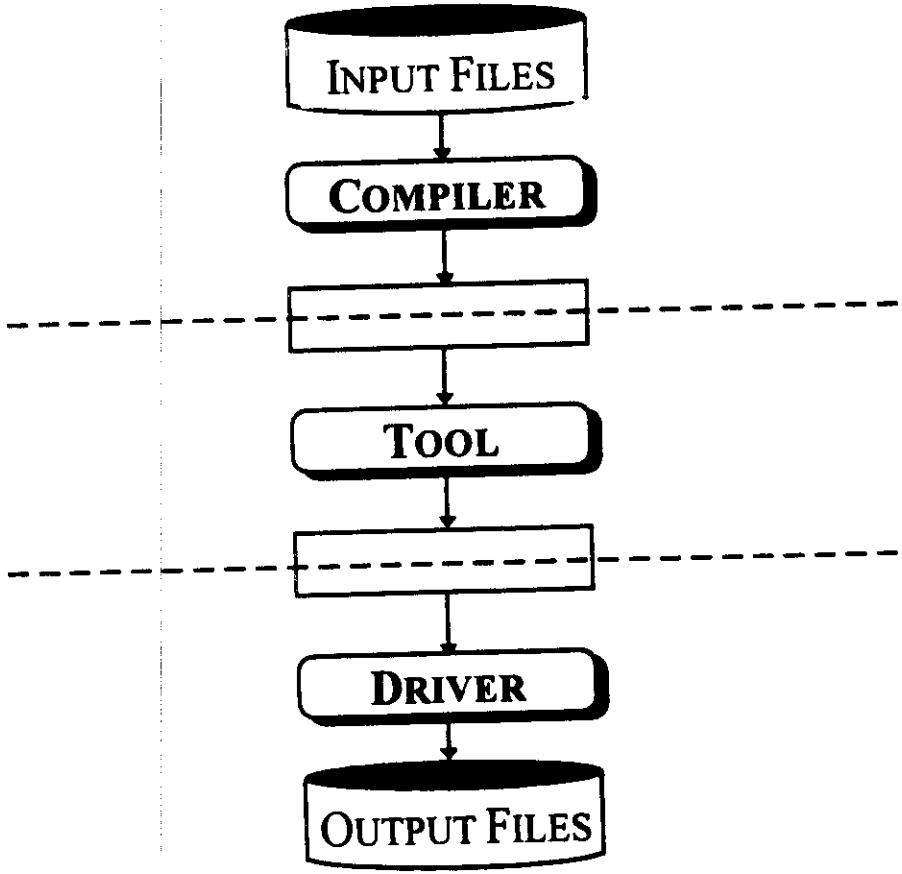
AUTOMATIC TRANSLATION FROM SYMBOLIC TO PHYSICAL.



How to Deal with these Views ? (1)



How to Deal with these Views ? (2)



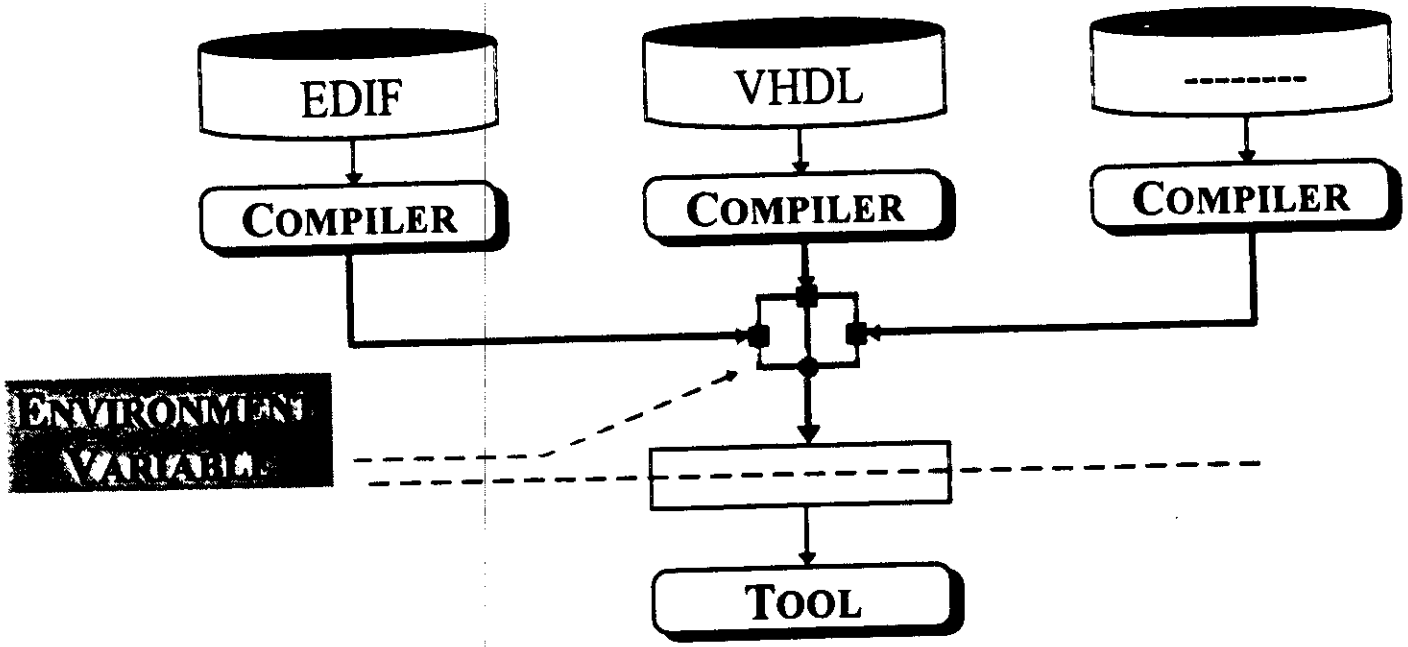
INDEPENDENCE (1)

A MAJOR IDEA IN ALLIANCE IS ITS INDEPENDENCE FROM ANY GIVEN LANGUAGE.

IDENTIFY THE CONCEPTS THAT:

- ✗ DO NOT DEPEND ON A LANGUAGE.
- ✓ DEPENDS ON THE ABSTRACTION LEVEL.

INDEPENDENCE (2)



OUTLINE

I - INTRODUCTION.

II - DESIGN METHODOLOGY: AN OVERVIEW.

III - ABSTRACTION LEVELS IN ALLIANCE.

IV - VHDL: AN OVERVIEW.

V - VHDL: THE ALLIANCE SUBSET.

VI - ALLIANCE: A COMPLETE DESIGN SYSTEM.

VERY HIGH SPEED INTEGRATED CIRCUITS (VHSIC)

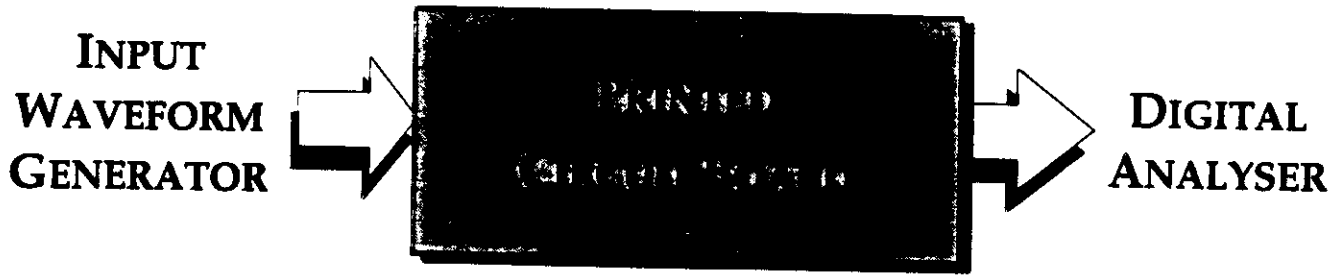
HARDWARE

DESCRIPTION

LANGUAGE

WHY DO WE NEED HDL ? (1)

- HARDWARE SOLUTION LIMITS.



WHY DO WE NEED HDL ? (2)

- INCREASING COMPLEXITY.
- INCREASING COST IN TIME AND INVESTMENT.

⇒A SOFTWARE SOLUTION IS NEEDED.

WHY DO WE NEED HDL ? (3)

✗ A PROGRAMMING LANGUAGE IS NOT SUITED FOR DESCRIBING HARDWARE.

⇒ A SPECIAL PURPOSE LANGUAGE IS NEEDED.

✓HARDWARE DESCRIPTION LANGUAGE.

WHY VHDL ? (1)

- CIRCUIT MANUFACTURERS ARE FULLY SATISFIED WITH THEIR PROPRIETARY HDLS.

WHY VHDL ? (2)

PROBLEM FOR SYSTEM MANUFACTURERS.

DIFFERENT VENDORS ⇒ DIFFERENT INCOMPATIBLE HDLS.

- IMPOSSIBLE TO VERIFY A WHOLE MIXED SYSTEM.

WHY VHDL ? (3)

✘ VENDOR DEPENDENCY.

✘ DESIGN EXCHANGE PROBLEM.

⇒ A NEED FOR A STANDARD HDL FROM THE SYSTEM MANUFACTURER'S POINT OF VIEW.

✓ VHSIC HARDWARE DESCRIPTION LANGUAGE.

HISTORY

- 1981: AN EXTENSIVE PUBLIC REVIEW (DOD).
- 1983: A REQUEST FOR PROPOSAL.
  (INTERMETRICS, IBM, AND TEXAS INSTRUMENTS).
- 1986: VHDL IN THE PUBLIC DOMAIN.
- 1987: A STANDARD LANGUAGE (IEEE-1076).

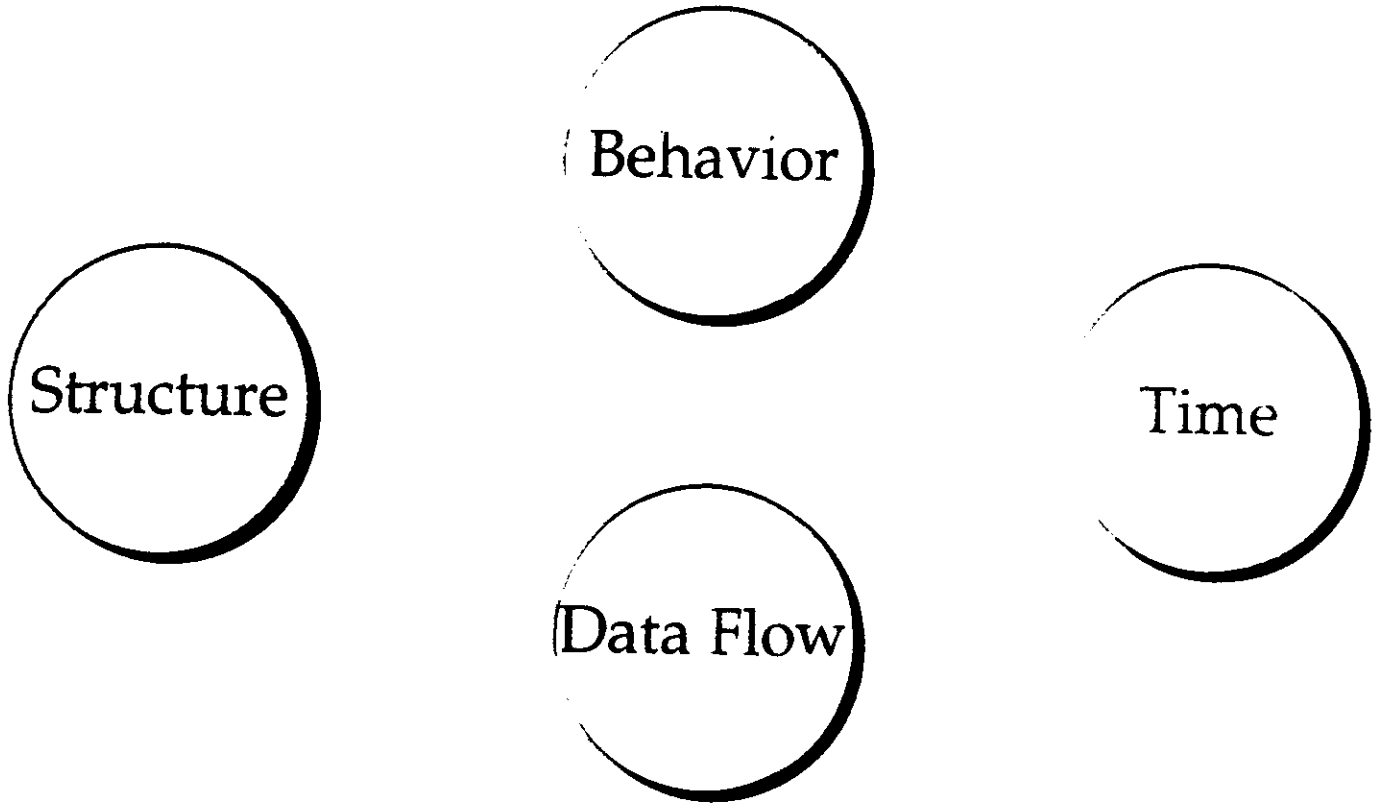
VHDL

STANDARD ⇒ OPEN LANGUAGE

⇓

- ✓ VENDOR INDEPENDENCE
- ✓ USER DEFINABLE
- ✓ WIDE CAPABILITIES
- ✗ COMPLEX TOOLS
- ✗ SLOW TOOLS

VHDL: DIFFERENT VIEWS



VHDL: A MODEL OF BEHAVIOR (1)

THIS IS THE HIGHEST LEVEL OF ABSTRACTION.

A FUNCTIONAL INTERPRETATION OF A DISCRETE SYSTEM.

$\Rightarrow$ SEQUENTIAL INSTRUCTIONS ARE NEEDED (PROGRAMS).

CORRESPONDS TO THE PROGRAMMER POINT OF VIEW.

VHDL: A MODEL OF BEHAVIOR (2)

VHDL OFFERS:

- PROCEDURES AND FUNCTIONS.
- SEQUENTIAL INSTRUCTIONS (IF, CASE, LOOP, NEXT).
- PROCESSES.

VHDL IS A COLLECTION OF PROCESSES RUNNING CONCURRENTLY OR IN PARALLEL.

VHDL: THE DATA FLOW VIEW (1)

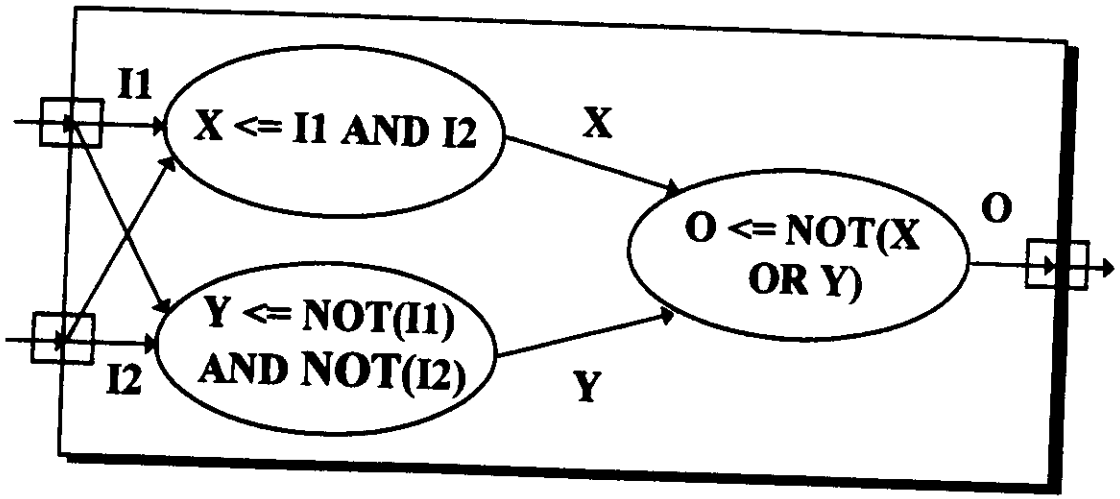
THIS VIEW IS HALFWAY BETWEEN THE STRUCTUREL DESCRIPTION AND THE BEHAVIOUR DESCRIPTION.

THE KEYWORD IS CONCURRENT INSTRUCTIONS.

CORRESPONDS TO THE CIRCUIT DESIGNER POINT OF VIEW.

VHDL: THE DATA FLOW VIEW (2)

THE DISCRETE SYSTEM SHOULD REACT WHENEVER ITS INPUTS CHANGE.



VHDL: A MODEL OF STRUCTURE (1)

WHAT FOR ?

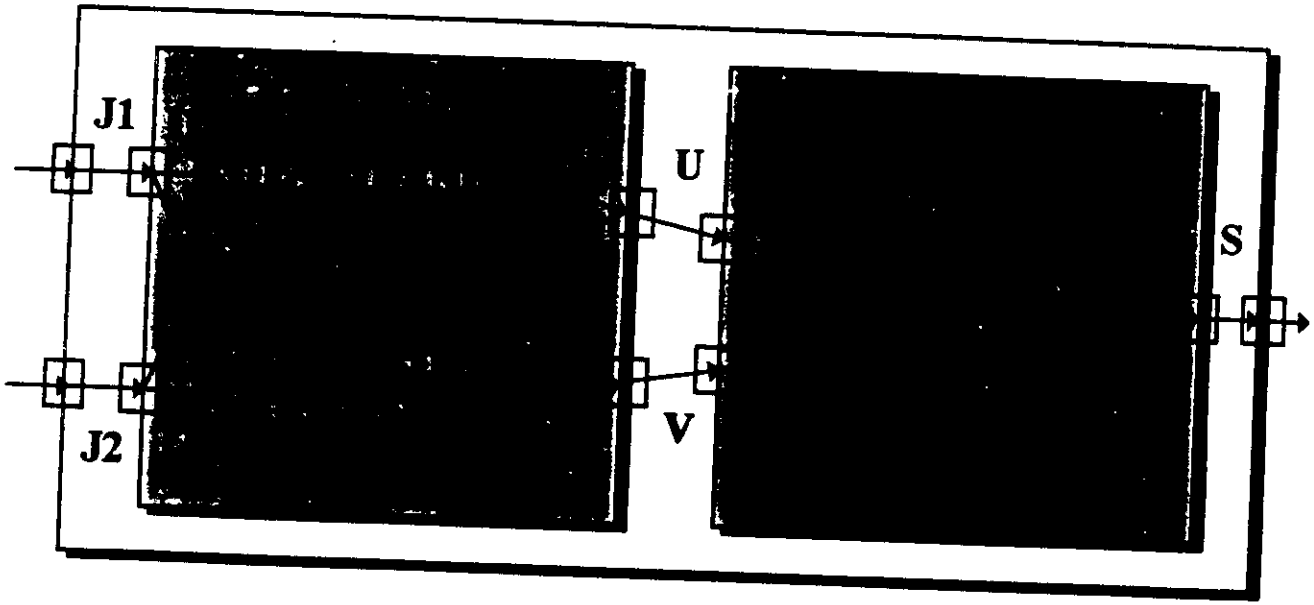
✗ DEALING WITH COMPLEX SYSTEMS IS A HARD TASK.

THE DIVIDE AND CONQUER STRATEGY

COMBINATION OF A NUMBER OF SUBSYSTEMS

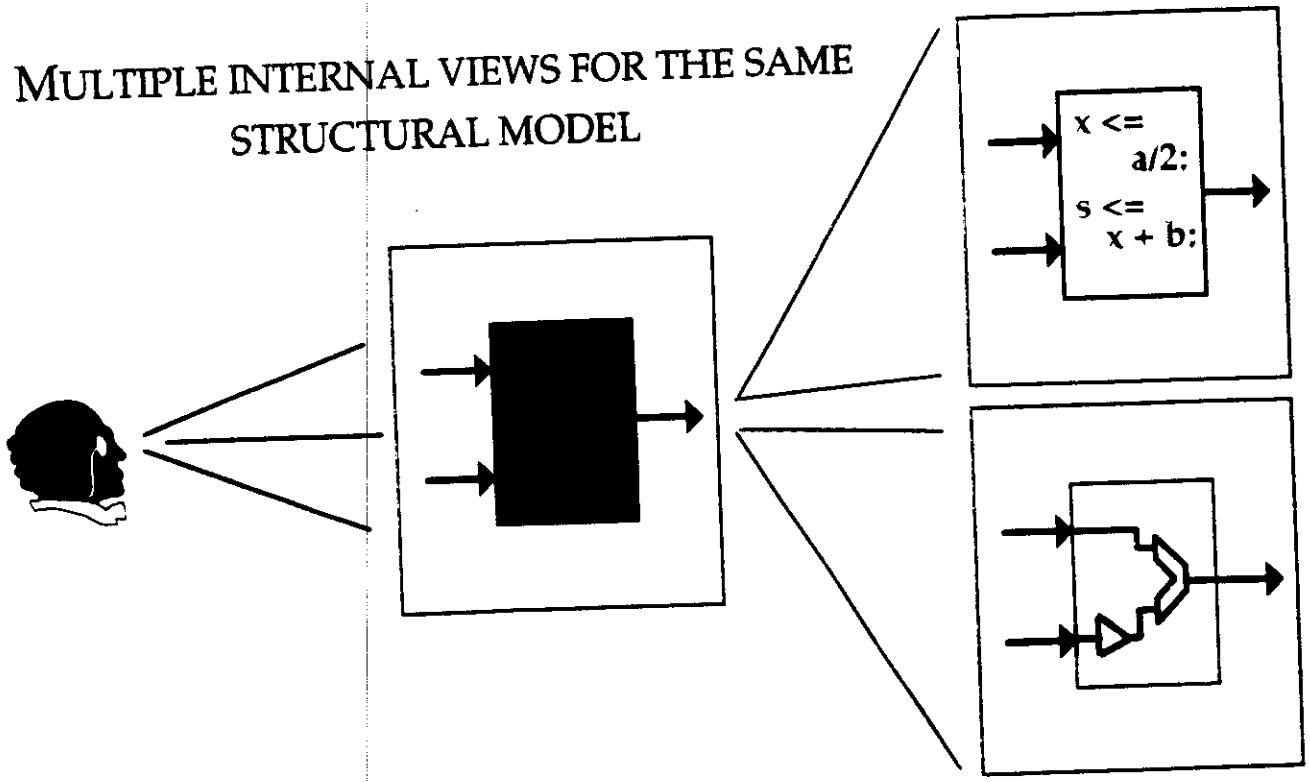
✓ HIERARCHY

VHDL: A MODEL OF STRUCTURE (2)



VHDL: A MODEL OF STRUCTURE (3)

MULTIPLE INTERNAL VIEWS FOR THE SAME STRUCTURAL MODEL



VHDL FEATURES

- TYPES:
  - EACH OBJECT HAS A TYPE
- OBJECTS:
  - EACH OBJECT MAY HAVE AN ELECTRONIC INTERPRETATION (SIGNAL).
  - EACH OBJECT MAY HAVE AN ALGORITHMITC MEANING (VARIABLE).

CONCLUSION (1)

VHDL IS AN OPEN LANGUAGE WITH MANY FEATURES.

WITH VHDL, ANY DISCRETE SYSTEM CAN BE MODELED.

CONCLUSION (2)

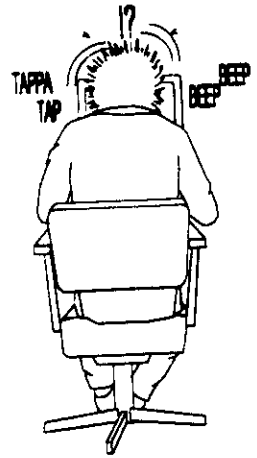
EACH USER HAS ITS OWN NEEDS DEPENDING ON:

- HIS BACKGROUND.
- HIS ENVIRONMENT.

WE DEFINED A SUBSET OF VHDL.

CONCLUSION (3)

WHY ?



COMPLEX LANGUAGE ⇒ DEVELOPING A COMPILER IS HARD AND TIME CONSUMING.

CONCLUSION (4)

WHY ?

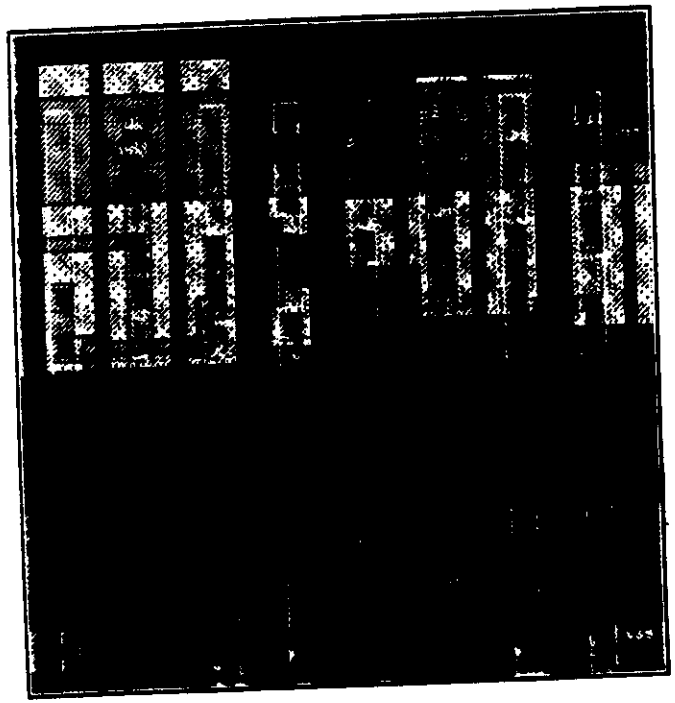
EDUCATIONAL NEEDS:

- UNDERSTANDING TIME.
- UNIVOCAL (ONE WAY FOR DESCRIBING A REGISTER).

CONCLUSION (5)

WHY ?

OUR ENVIRONMENT: VLSI.



OUTLINE

I - INTRODUCTION.

II - DESIGN METHODOLOGY: AN OVERVIEW.

III - ABSTRACTION LEVELS IN ALLIANCE.

IV - VHDL: AN OVERVIEW.

V - VHDL: THE ALLIANCE SUBSET.

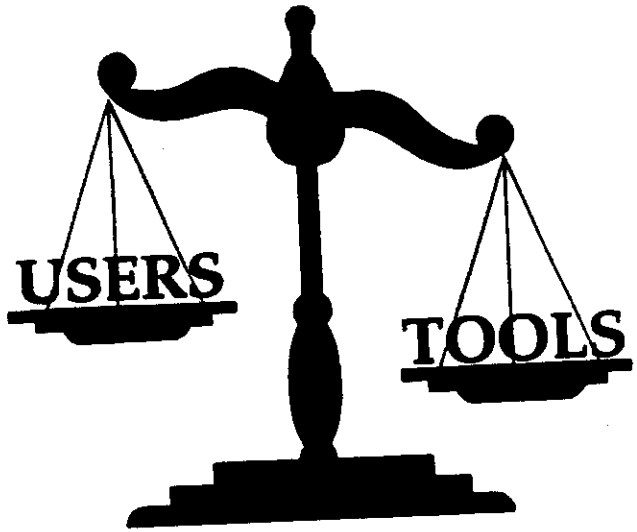
VI - ALLIANCE: A COMPLETE DESIGN SYSTEM.

WHY AND HOW ?

WHY ?

- DEVELOPMENT TIME.
- EDUCATION CONSTRAINTS.
- THE CURRENT ENVIRONMENT.

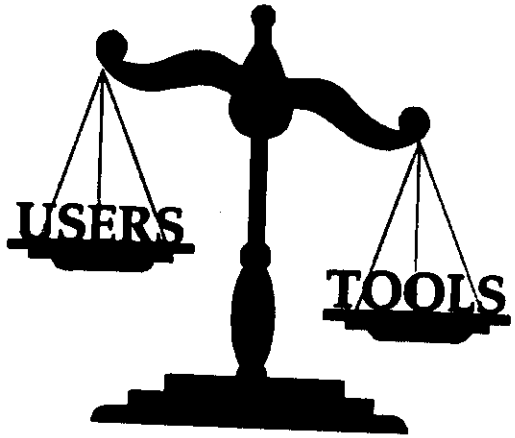
CRITERIONS FOR THE SUBSET DEFINITION.



TOOLS REQUIREMENTS (1)

WHICH TOOLS USE VHDL ?

- SYNTHESIS.
- FORMAL PROOVER.
- PLACER & ROUTER.
- SIMULATOR.
- FUNCTIONAL ABSTRACTOR.



TOOLS REQUIREMENTS (2)

SYNTHESIS TOOLS.

- A REGISTER MUST BE IDENTIFIED IN A SYNTACTICAL WAY.
- A BUS MUST BE IDENTIFIED IN A SYNTACTICAL WAY.
- SIGNALS MUST HAVE THE **BIT** TYPE ('0' , '1').
- NO TIMING.

FORMAL PROOVER.

- A REGISTER MUST BE IDENTIFIED IN A SYNTACTICAL WAY.
- A BUS MUST BE IDENTIFIED IN A SYNTACTICAL WAY.

TOOLS REQUIREMENTS (3)

FUNCTIONAL ABSTRACTOR.

- THE VHDL SUBSET MUST BE AS CLOSE AS POSSIBLE TO THE HARDWARE.

PLACER & ROUTER.

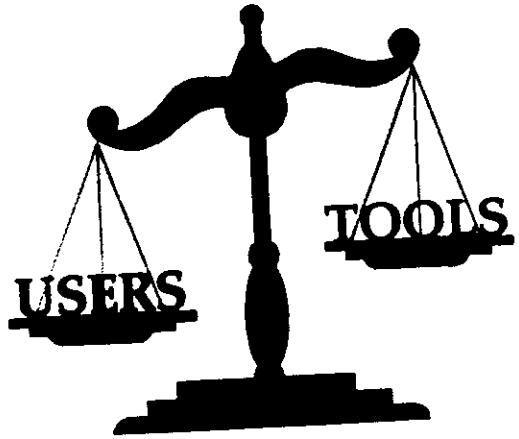
- NO MIXING BETWEEN STRUCTURAL AND BEHAVIORAL VIEWS.

SIMULATOR.

- NO ABSTRACT TYPES.
- NO TIMING.

USERS REQUIREMENTS

LOOKING FOR THE LARGEST SUBSET.



THE GOOD VHDL SUBSET:

- ✓ LETS THE USER DESCRIBE HIS CIRCUIT EASILY.
- ✓ DO NOT DETERIORATE THE TOOL WITH A COMPLEX LANGUAGE.

THE EXTERNAL ASPECT

IN VHDL A CIRCUIT (DESIGN UNIT) HAS TWO ASPECTS:

1. THE EXTERNAL ASPECT: (EXTERNAL VISIBILITY)

What is visible

ALLIANCE

- ✓ NAME
- ✓ INTERFACE (PORT)
- × COLOR
- × TEMPERATURE
- × ————————

THE INTERNAL ASPECT (1)

IN VHDL A CIRCUIT (DESIGN UNIT) HAS TWO ASPECTS:

2.THE INTERNAL ASPECT: (FUNCTIONALITY)

ALLIANCE	✓STRUCTURAL ✓DATA FLOW

how it works

THE INTERNAL ASPECT (2)

IN THE STRUCTURAL INTERNAL ASPECT, WE DESCRIBE THE CIRCUIT AS A NETWORK OF SMALLER CIRCUITS.

THE FOLLOWING OBJECTS ARE USED:

- SIGNAL.
- COMPONENT (MODEL).
- INSTANCE.

EXTERNAL ASPECT: EXAMPLE (1)

```
ENTITY PARITY IS
 PORT (
        A : IN BIT ;
        B : IN BIT;
        C : IN BIT ;
        D : IN BIT;
        P : OUT BIT
       )
 END PARITY;
```

Circuit Name

Port Name

Input/output mode

Type

EXTERNAL ASPECT: EXAMPLE (2)

```
ENTITY ADDER_32 IS
 PORT (
        A : IN BIT_VECTOR (31 DOWNTO 0);
        B : IN BIT_VECTOR (31 DOWNTO 0);
        CIN : IN BIT;
        SUM : OUT BIT_VECTOR (31 DOWNTO 0);
        COUT : OUT BIT
       )
END;
```

INTERNAL STRUCTURAL EXAMPLE (1)

```
ARCHITECTURE PSTRUCT OF PARITY IS
    COMPONENT XOR_Y
        PORT (
                I0 : IN BIT ;
                I1 : IN BIT;
                T : OUT BIT
            );
    END COMPONENT;
SIGNAL PARITY_AB : BIT;
SIGNAL PARITY_CD : BIT;
```

DECLARATIVE PART

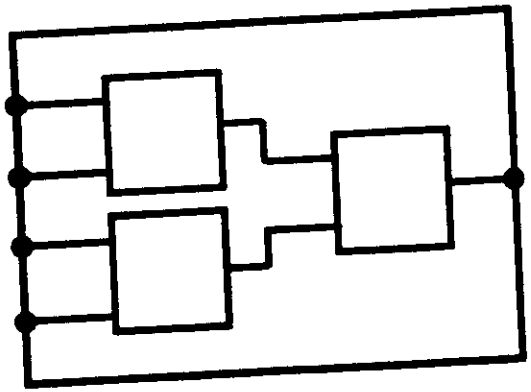
INTERNAL STRUCTURAL EXAMPLE (2)

```
BEGIN
    INSTANCE_AB : XOR_Y
        PORT MAP (
                    I0 => A,
                    I1 => B,
                    T => PARITY_AB
                  );

    INSTANCE_CD : XOR_Y
        PORT MAP (
                    I0 => C,
                    I1 => D,
                    T => PARITY_CD
                  );
```

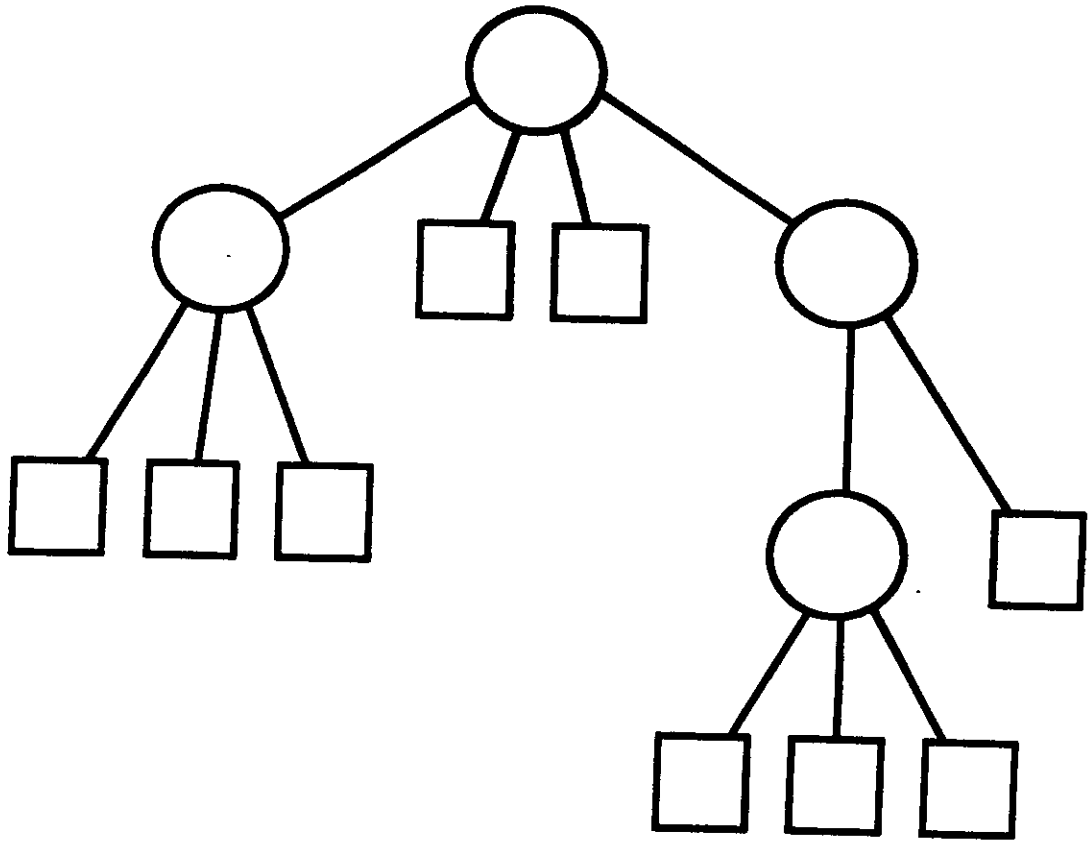
DESCRIPTION PART

INTERNAL STRUCTURAL EXAMPLE (3)

```
INSTANCE_ABCD : XOR_Y
    PORT MAP (
                    I0 => PARITY_AB,
                    I1 => PARITY_CD,
                    T => P
                   );

END;
```



STRUCTURAL & HIERARCHICAL



INTERNAL BEHAVIORAL ASPECT (1)

DESCRIBING EQUATIONS BETWEEN INPUTS AND OUTPUTS.

- BOOLEAN FUNCTIONS:
  - AND
  - OR
  - XOR
  - NAND
  - NOR
  - NOT

ALWAYS USE BRACKETS.

INTERNAL BEHAVIORAL ASPECT (2)

DESCRIBING EQUATIONS BETWEEN INPUTS AND OUTPUTS.

- ASSERT (CONDITION)
  REPORT "MESSAGE"
  SEVERITY LEVEL;

VERY USEFUL IN LARGE-SCALE DESIGN.

- ♦ ALLOWS ENCODING SPECIFIC CONSTRAINTS AND ERROR CONDITIONS
- ♦ PROVIDE USEFUL MESSAGES.
- ♦ STOP THE SIMULATION WHEN CONSTRAINTS ARE NOT MET.

INTERNAL BEHAVIORAL ASPECT (3)

- THREE KINDS OF ASSIGNMENTS:

  SIMPLE ASSIGNMENT:

```
S <= A AND B;
```

  CONDITIONNAL ASSIGNMENT:

```
S <= A AND B WHEN ( C = '0' ) ELSE
     D OR E;
```

Always

INTERNAL BEHAVIORAL ASPECT (4)

SELECTIVE ASSIGNMENT:

```
WITH ADDRESS(3 DOWNTO 0) SELECT
    OUT <= "000100" WHEN "0000",
           "000101" WHEN "0001",
           --------
           "000000" WHEN OTHERS;
```

INTERNAL BEHAVIORAL ASPECT (5)

- REGISTERS:

```
SIGNAL MYREGISTER : REG_BIT REGISTER;


STORE : BLOCK ( CK = '0' AND NOT CK'STABLE )
BEGIN
MYREGISTER <= GUARDED I0;
END BLOCK STORE;
```

INTERNAL BEHAVIORAL ASPECT (6)

- BUS:

SIGNAL MY_BUS1 : MUX_BIT BUS;

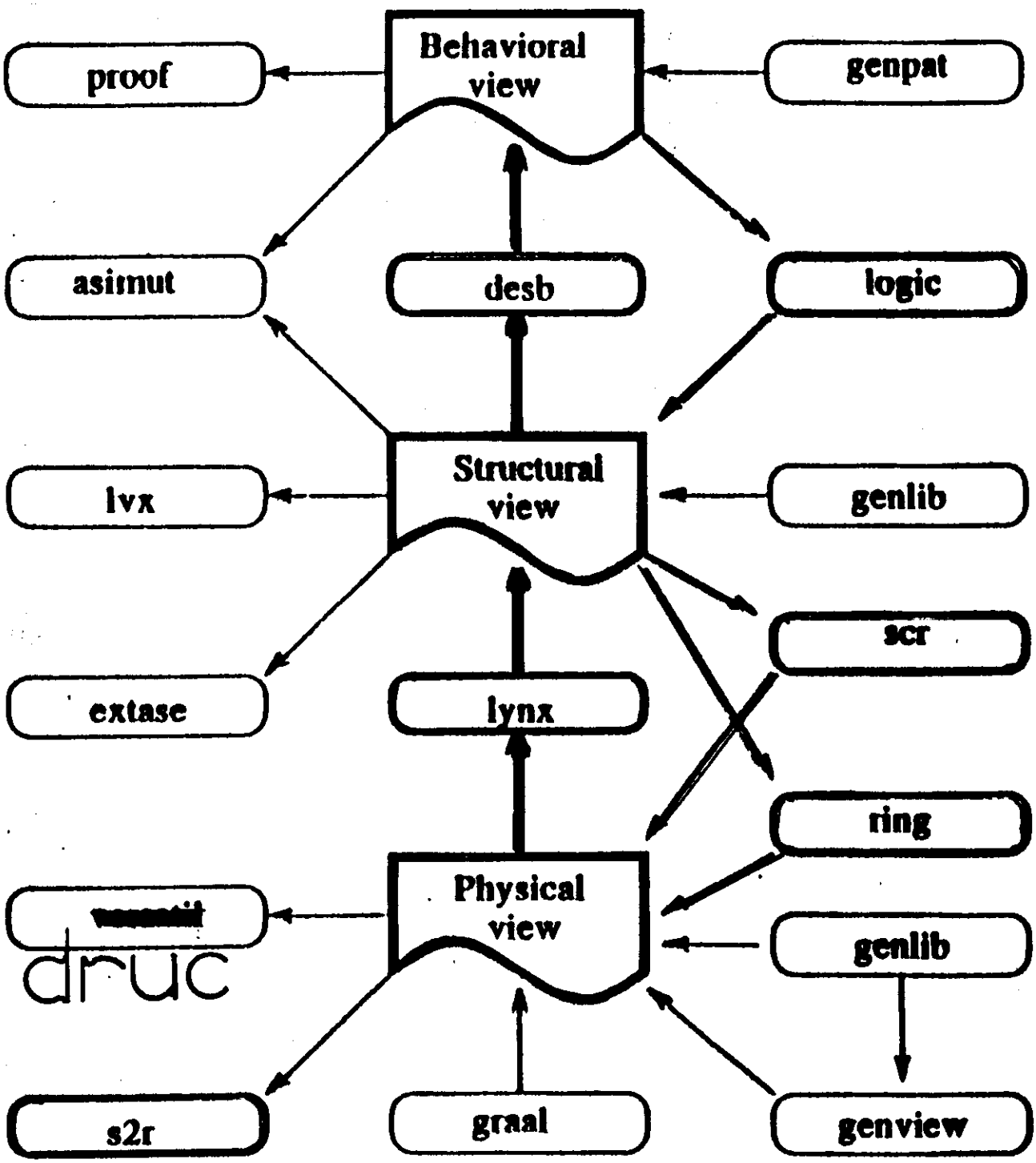
ONLY ONE DRIVER ACTIVE AT THE SAME TIME.

SIGNAL MY_BUS2 : WOR_BIT BUS;

MANY DRIVERS DRIVE THE SAME VALUE.

INTERNAL BEHAVIORAL EXAMPLE

```
ARCHITECTURE DATA_FLOW OF PARITY IS
   SIGNAL PARITY_AB : BIT;
   SIGNAL PARITY_CD : BIT;
BEGIN
      PARITY_AB <= A XOR B;
      PARITY_CD <= C XOR D;
      P <= PARITY_AB XOR PARITY_CD;
End;
```


Behavioral view
proof
genpat
asimut
desb
logic
Structural view
lvx
genlib
scr
extase
lynx
ring
Physical view
druc
genlib
s2r
graal
genview