LOW-LEVEL DESIGN ENTRY

9

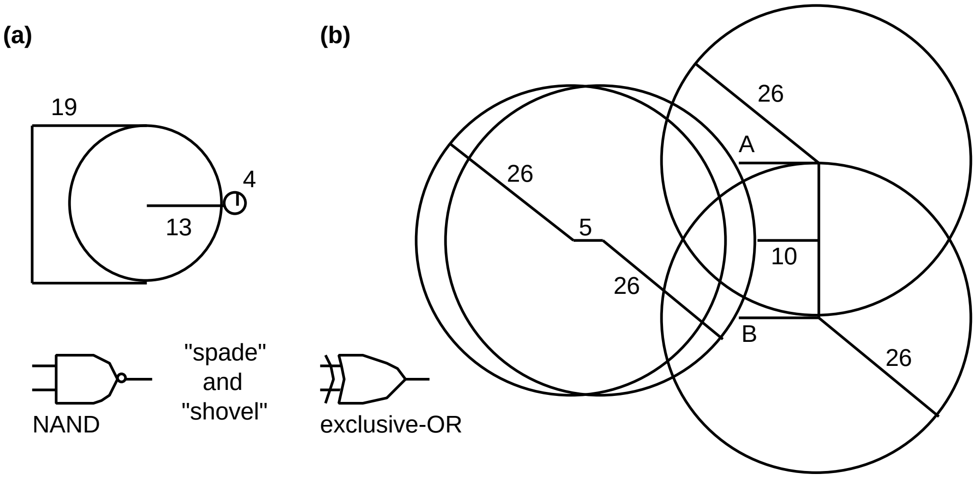
Key concepts: design entry • electronic-design automation (EDA) • schematic • connectivity • schematic entry • schematic capture • netlist • documentation • hardware description language (HDL) • logic synthesis • low-level design-entry

9.1 Schematic Entry

Key terms and concepts: graphical design entry • transforms an idea to a computer file • an "old" method that periodically regains popularity • schematic sheets • frame • border • "spades" and "shovels" • component or device • low-cost

ANSI (American National Standards Institute) and ISO (International Standards Organization) schematic sheet sizes

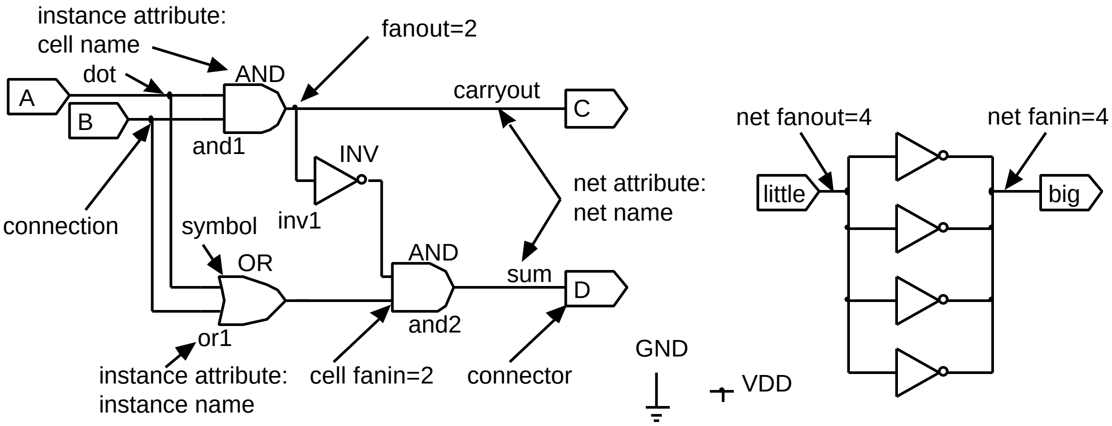
ANSI sheet	Size (inches)	ISO sheet	Size (cm)
A	8.5 × 11	A5	21.0 × 14.8
B	11 × 17	A4	29.7 × 21.0
C	17 × 22	A3	42.0 × 29.7
D	22 × 34	A2	59.4 × 42.0
E	34 × 44	A1	84.0 × 59.4
		A0	118.9 × 84.0



IEEE-recommended dimensions and their construction for logic-gate symbols

(a) NAND gate

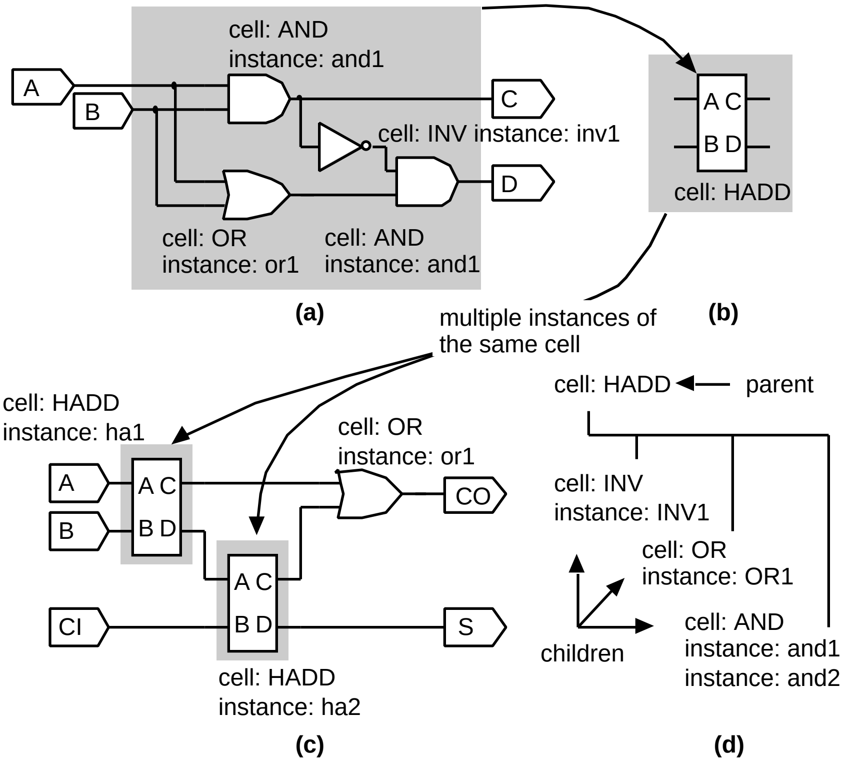
(b) exclusive-OR gate (an OR gate is a subset)



Terms used in circuit schematics

9.1.1 Hierarchical Design

Key terms and concepts: use of hierarchy to hide complexity • hierarchical design • subschematic • child • parent • flat design • flat netlist



Schematic example showing hierarchical design

(a) The schematic of a half-adder, the subschematic of cell HADD

(b) A schematic symbol for the half adder

(c) A schematic that uses the half-adder cell

(d) The hierarchy of cell HADD

9.1.2 The Cell Library

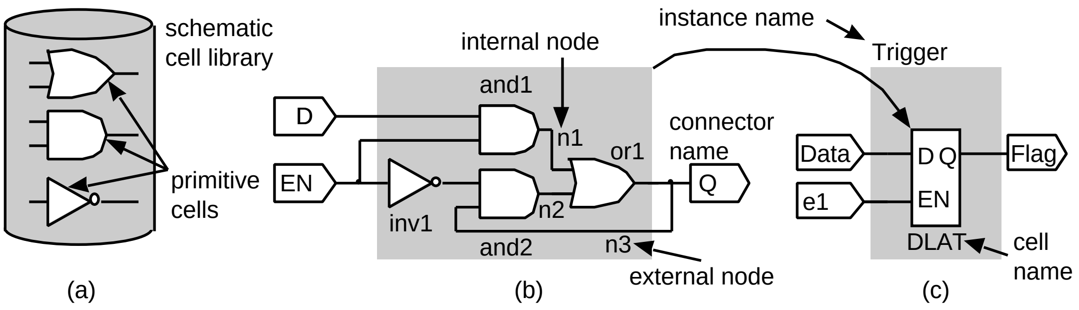
Key terms: modules (cells, gates, macros, books) • schematic library (vendor-dependent) • retargeting • porting a design • primitive cells or cells (flip-flops or transistors?) • hard macro (placement) • soft macro (connection)

9.1.3 Names

Key terms: cell name • cell instance • instance name • icon (picture) • symbol • name spaces • case sensistivity • hierarchical names

9.1.4 Schematic Icons and Symbols

Key terms: derived icon • derived symbol • subcell • vectored instance • cardinality

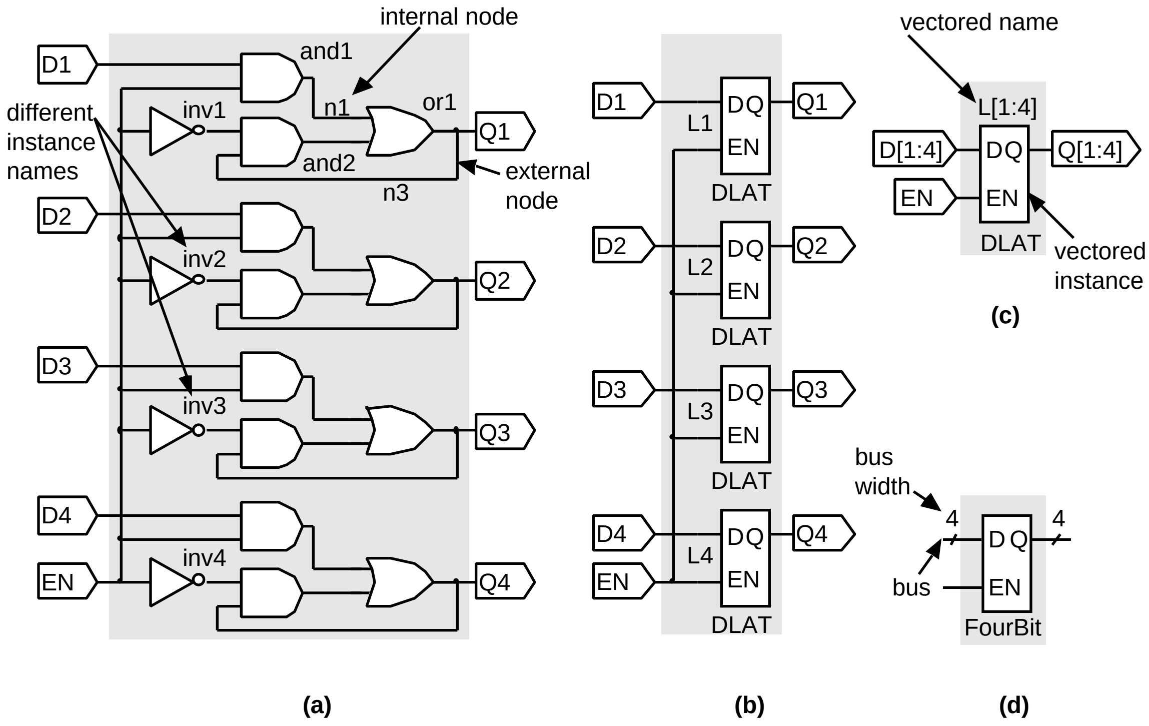


A cell and its subschematic

(a) A schematic library containing icons for the primitive cells

(b) A subschematic for a cell, DLAT, showing the instance names for the primitive cells

(c) A symbol for cell DLAT



A 4-bit latch:

(a) drawn as a flat schematic from gate-level primitives

(b) drawn as four instances of the cell symbol DLAT

(c) drawn using a vectored instance of the DLAT cell symbol with cardinality of 4

(d) drawn using a new cell symbol with cell name FourBit

9.1.5 Nets

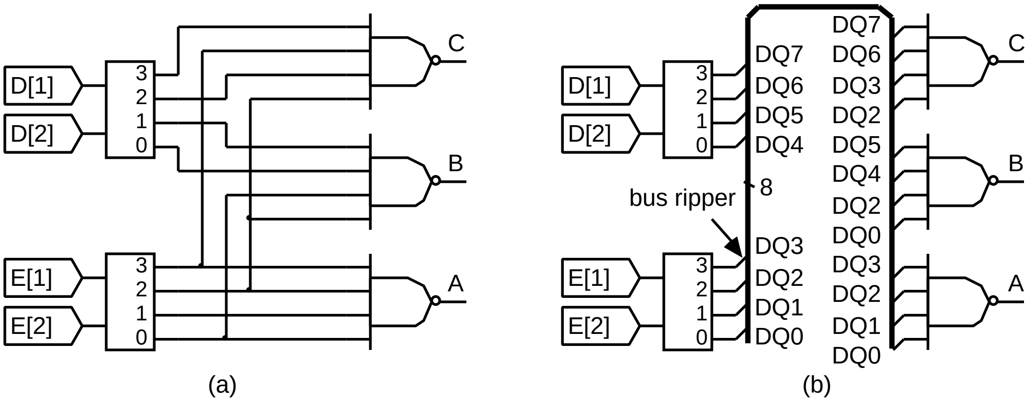
Key terms: local nets • external nets • delimiter • Verilog and VHDL naming

9.1.6 Schematic Entry for ASICs and PCBs

Key terms: component • TTL SN74LS00N • Quad 2-input NAND • component parts • reference designator • R99 • pin number • part assignment

9.1.7 Connections

Key terms: terminals • pins, connectors, or signals • wire segments or nets • bus or buses (not busses) • bundle or array • breakout • ripper (EDIF) • extractor • swizzle (Compass datapath)

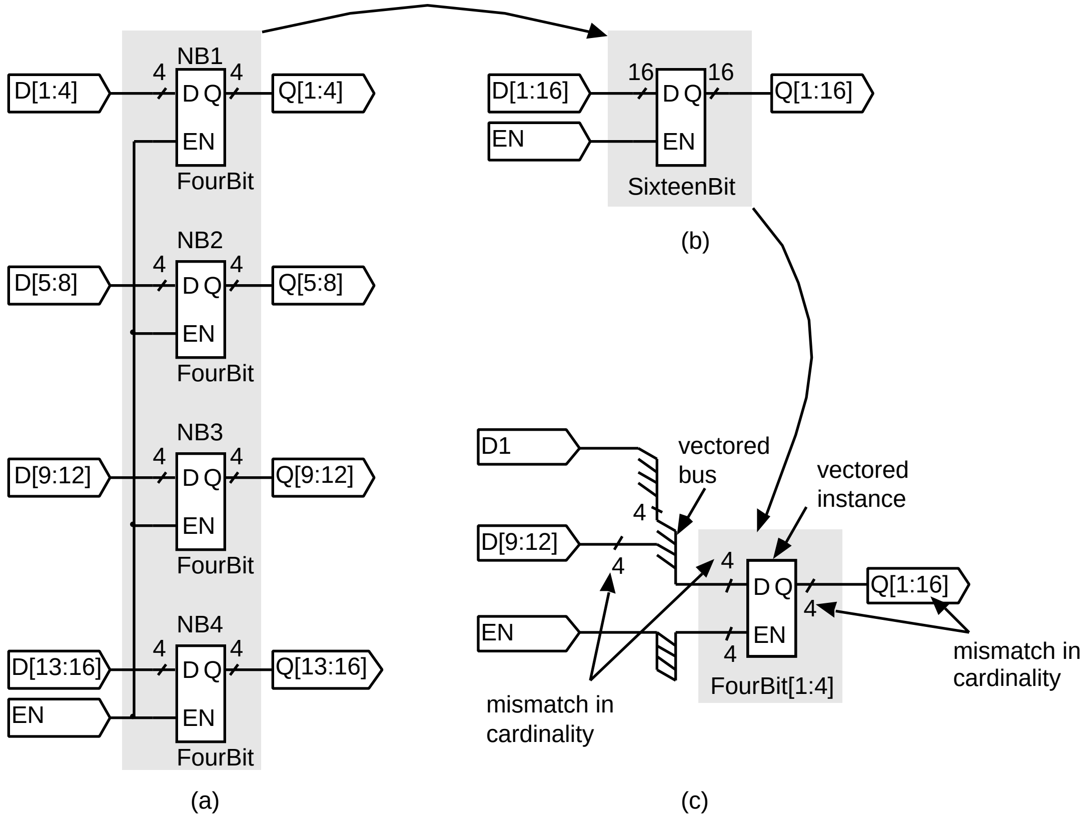


An example of the use of a bus to simplify a schematic

(a) An address decoder without using a bus

(b) A bus with bus rippers simplifies the schematic and reduces the possibility of making a mistake in creating and reading the schematic

9.1.8 Vectored Instances and Buses



A 16-bit latch:

(a) drawn as four instances of cell FourBit

(b) drawn as a cell named SixteenBit

(c) drawn as four multiple instances of cell FourBit

9.1.9 Edit-in-Place

Key terms: edit-in-place • alias • dictionary of names

9.1.10 Attributes

Key terms: name • identifier • label • attribute • property • NFS filenames (28 characters)

9.1.11 Netlist Screener

Key terms: schematic or netlist screener catches errors at an early stage • handle (to find components) • snap to grid • wildcard matching • automatic naming • datapath (multiple instances) • vectored cell instance • vectored instance • cell cardinality • cardinality • terminal polarity • terminal direction • fanout • fanin • standard load

9.1.12 Schematic-Entry Tools

Key terms: icon edit-in-place • timestamp or datestamp • versions • version number • design manager or library manager • version history • check-out • undo • rubber banding • global nets • connectors • off-page connector • multipage connector • fanout • fanin • standard load

9.1.13 Back-Annotation

Key terms: logical design • prelayout simulation• physical design • parasitic capacitance • interconnect delay • back-annotation • postlayout simulation

9.2 Low-Level Design Languages

Key terms and concepts: changes to a schematic are tedious • no standards for schematics • PLD design entry • a design language is better than schematic entry • a low-level design language is not as powerful as logic synthesis • legacy code

9.2.1 ABEL

ABEL

Statement	Example	Comment
Module	`module MyModule`	You can have multiple modules.
Title	`title 'Title in a String'`	A string is a character series between quotes.
Device	`MYDEV device '22V10' ;`	`MYDEV` is Device ID for documentation. `22V10` is checked by the compiler.

Comment	`"comments go between double quotes"`
	`"end of line is end of comment`
		The end of a line signifies the end of a comment; there is no need for an end quote.

@ALTERNATE	`@ALTERNATE "use alternate symbols`	operator alternate default AND * & OR + # NOT / ! XOR :+: $ XNOR :*: !$

Pin declaration	`MYINPUT pin 2; I3, I4 pin 3, 4 ;`
	`/MYOUTPUT pin 22; IO3,IO4 pin 21,20 ;`
		Pin 22 is the IO for input on pin 2 for a 22V10.
		`MYOUTPUT` is active-low at the chip pin.
		Signal names must start with a letter.

Equations	`equations`
	`IO4 = HELPER ; HELPER = /I4 ;`
		Defines combinational logic.
		Two-pass logic

Assignments	`MYOUTPUT = /MYINPUT ;`	Equals `'='` is unlocked assignment.

	IO3 := I4 ;	Clocked assignment operator (registered IO)

Signal sets	D = [D0, D1, D2, D3] ;
	Q = [Q0, Q1, Q2, Q3];
		A signal set, an ABEL bus

	Q := D ;	4-bit-wide register
Suffix	MYOUTPUT.RE = CLR ;	Register reset
	MYOUTPUT.PR = PRE ;	Register preset

Addition	COUNT = [D0, D1, D2];
	COUNT := COUNT + 1;
		Can’t use @ALTERNATE
		if you use '+' to add.

Enable	ENABLE IO3 = IO2;
	IO3 = MYINPUT;
		Three-state enable (ENABLE is a keyword).
		IO3 must be a three-state pin.

Constants	K = [1, 0, 1] ;	K is 5.
Relational	IO# = D == K5 ;	Operators: == != < > <= >=
End	end MyModule	Last statement in module

Example:

```
module MUX4
title '4:1 MUX'
MyDevice device 'P16L8' ;
@ALTERNATE
"inputs
A, B, /P1G1, /P1G2 pin 17,18,1,6 "LS153 pins 14,2,1,15
P1C0, P1C1, P1C2, P1C3 pin 2,3,4,5 "LS153 pins 6,5,4,3
P2C0, P2C1, P2C2, P2C3 pin 7,8,9,11 "LS153 pins 10,11,12,13
"outputs
P1Y, P2Y pin 19, 12 "LS153 pins 7,9
equations
  P1Y = P1G*(/B*/A*P1C0 + /B*A*P1C1 + B*/A*P1C2 + B*A*P1C3);
  P1Y = P1G*(/B*/A*P1C0 + /B*A*P1C1 + B*/A*P1C2 + B*A*P1C3);
end MUX4
```

9.2.2 CUPL

Key terms and concepts: CUPL is a PLD design language from Logical Devices • CUPL 4.0 • extension • fitter • Atmel ATV2500B • complex PLD • “buried” features • pin-number tables • skeleton headers and pin declarations

```
SEQUENCE BayBridgeTollPlaza {
 PRESENT red
   IF car NEXT green OUT go; /* conditional synchronous output */
   DEFAULT NEXT red;   /* default next state */
 PRESENT green
   NEXT red; }         /* unconditional next state */
```

CUPL statements for state-machine entry

Statement			Description
`IF`	`NEXT`		Conditional next state transition
`IF`	`NEXT`	`OUT`	Conditional next state transition with synchronous output
	`NEXT`		Unconditional next state transition
	`NEXT`	`OUT`	Unconditional next state transition with asynchronous output
		`OUT`	Unconditional asynchronous output
`IF`		`OUT`	Conditional asynchronous output
`DEFAULT`	`NEXT`		Default next state transition
`DEFAULT`		`OUT`	Default asynchronous output
`DEFAULT`	`NEXT`	`OUT`	Default next state transition with synchronous output

You may encode state machines as truth tables in CUPL:

```
FIELD input = [in1..0];
FIELD output = [out3..0];
TABLE input => output {00 => 01; 01 => 02; 10 => 04; 11 => 08; }
```

CUPL file for a 4-bit counter (for an ATMEL PLD) that illustrates extensions:

```
Name 4BIT; Device V2500B;
/* inputs */
```

```
pin 1 = CLK; pin 3 = LD_; pin 17 = RST_;
pin [18,19,20,21] = [I0,I1,I2,I3];
/*  outputs  */
pin [4,5,6,7] = [Q0,Q1,Q2,Q3];
field CNT = [Q3,Q2,Q1,Q0];
/* equations */
Q3.T = (!Q2 & !Q1 & !Q0) & LD_ & RST_ /* count down */
     # Q3 & !RST_ /* ReSeT */
     # (Q3 $ I3) & !LD_; /* LoaD*/
Q2.T = (!Q1 & !Q0) & LD_ & RST_ # Q2 & !RST_ # (Q2 $ I2) & !LD_;
Q1.T = !Q0 & LD_ & RST_ # Q1 & !RST_ # (Q1 $ I1) & !LD_;
Q0.T = LD_ & RST_ # Q0 & !RST_ # (Q0 $ I0) & !LD_;
CNT.CK = CLK; CNT.OE = 'h'F; CNT.AR = 'h'0; CNT.SP = 'h'0;
```

CUPL extensions guide the **logic fitter**, for example:

```
output.ext = (Boolean expression);
```

.OE is output enable

.CK marks the clock

.T configures sequential logic as T flip-flops

.OE (wired high) is an output enable

.AR (wired low) is an asynchronous reset

.SP (wired low) is an synchronous preset

CUPL 4.0 extensions

Extension		Explanation	Extension		Explanation
D	L	D input to a D register	DFB	R	D register feedback of combinational output
L	L	L input to a latch	LFB	R	Latched feedback of combinational output
J, K	L	J-K-input to a J-K register	TFB	R	T register feedback of combinational output
S, R	L	S-R input to an S-R register	INT	R	Internal feedback
T	L	T input to a T register	IO	R	Pin feedback of registered output
DQ	R	D output of an input D register	IOD/T	R	D/T register on pin feedback path selection
LQ	R	Q output of an input latch	IOL	R	Latch on pin feedback path selection
AP, AR	L	Asynchronous preset/reset	IOAP, IOAR	L	Asynchronous preset/reset of register on feedback path
SP, SR	L	Synchronous preset/reset	IOSP, IOSR	L	Synchronous preset/reset of register on feedback path
CK	L	Product clock term (async.)	IOCK	L	Clock for pin feedback register
OE	L	Product-term output enable	APMUX, ARMUX	L	Asynchronous preset/reset multiplexor selection
CA	L	Complement array	CKMUX	L	Clock multiplexor selector
PR	L	Programmable preload	LEMUX	L	Latch enable multiplexor selector
CE	L	CE input of a D-CE register	OEMUX	L	Output enable multiplexor selector
LE	L	Product-term latch enable	IMUX	L	Input multiplexor selector of two pins
OBS	L	Programmable observability of buried nodes	TEC	L	Technology-dependent fuse selection
BYP	L	Programmable register bypass	T1	L	T1 input of 2-T register

ABEL and CUPL pin declarations for an ATMEL ATV2500B

ABEL	CUPL
device_id device 'P2500B';	
"device_id used for JEDEC	
filename	device V2500B;
I1,I2,I3,I17,I18 pin 1,2,3,17,18;	pin [1,2,3,17,18] =
O4,O5 pin 4,5 istype	[I1,I2,I3,I17,I18];
'reg_d,buffer';	pin [7,6,5,4] = [O7,O6,O5,O4];
O6,O7 pin 6,7 istype 'com';	pinnode [41,65,44] =
O4Q2,O7Q2 node 41,44 istype	[O4Q2,O4Q1,O7Q2];
'reg_d';	pinnode [43,68] = [O6Q2,O7Q1];
O6F2 node 43 istype 'com';	
O7Q1 node 220 istype 'reg_d';	

9.2.3 PALASM

Key terms and concepts: PALASM is a PLD design language from AMD/MMI • PALASM 2 • ordering of the pin numbers is important • DEVICE • often need manufacturer's data sheet

PALASM 2

Statement	Example	Comment
Chip	`CHIP abc 22V10`	Specific PAL type
	`CHIP xyz USER`	Free-form equation entry
Pinlist	`CLK /LD D0 D1 D2 D3 D4 GND NC Q4 Q3 Q2 Q1 Q0 /RST VCC`	Part of `CHIP` statement; PAL pins in numerical order starting with pin 1
String	`STRING string_name 'text'`	Before `EQUATIONS` statement
Equations	`EQUATIONS`	After `CHIP` statement
	`A = /B`	Logical negation
	`A = B * C`	Logical AND
	`A = B + C`	Logical OR
	`A = B :+: C`	Logical exclusive-OR
	`A = B :*: C`	Logical exclusive-NOR
Polarity inversion	`/A = /(B + C)`	Same as `A = B + C`
Assignment	`A = B + C`	Combinational assignment
	`A := B + C`	Registered assignment
Comment	`A = B + C ; comment`	Comment
Functional equation	`name.TRST`	Output enable control
	`name.CLKF`	Register clock control
	`name.RSTF`	Register reset control
	`name.SETF`	Register set control

Example:

```
TITLE video ; shift register
CHIP video PAL20X8
CK /LD D0 D1 D2 D3 D4 D5 D6 D7 CURS GND NC REV Q7 Q6 Q5 Q4 Q3 Q2 Q1
Q0 /RST VCC
STRING Load 'LD*/REV*/CURS*RST' ; load data
STRING LoadInv 'LD*REV*/CURS*RST' ; load inverted of data
STRING Shift '/LD*/CURS*/RST' ; shift data from MSB to LSB
EQUATIONS
/Q0 := /D0*Load+D0*LoadInv:+:/Q1*Shift+RST
/Q1 := /D1*Load+D1*LoadInv:+:/Q2*Shift+RST
/Q2 := /D2*Load+D2*LoadInv:+:/Q3*Shift+RST
/Q3 := /D3*Load+D3*LoadInv:+:/Q4*Shift+RST
/Q4 := /D4*Load+D4*LoadInv:+:/Q5*Shift+RST
/Q5 := /D5*Load+D5*LoadInv:+:/Q6*Shift+RST
/Q6 := /D6*Load+D6*LoadInv:+:/Q7*Shift+RST
/Q7 := /D7*Load+D7*LoadInv:+:Shift+RST;
```

9.3 PLA Tools

Key terms and concepts: developed at UC Berkeley • `eqntott` input format • `espresso` logic-minimization program • widely used tools in the 1980s • important stepping stones to modern logic synthesis software

A PLA tools example

Input (6 minterms): `F1 = A|B|!C; F2 = !B&C; F3 = A&B|C;`

A	B	C	F1	F2	F3
0	0	0	1	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	1

`eqntott` output

```
.i 3
.o 3
.p 6
--0 100
--1 001
-01 010
-1- 100
1-- 100
11- 001
.e
```

`espresso` output

```
.i 3
.o 3
.p 6
1-- 100
11- 001
--0 100
-01 011
-11 101
.e
```

Output (5 minterms): `F1 = A|!C|(B&C); F2 = !B&C; F3 = A&B|(!B&C)|(B&C);`

The format of the input and output files used by the PLA design tool `espresso`

Expression	Explanation
`# comment`	# must be first character on a line
`[d]`	Decimal number
`[s]`	Character string
`.i [d]`	Number of input variables
`.o [d]`	Number of output variables
`.p [d]`	Number of product terms
`.ilb [s1] [s2]... [sn]`	Names of the binary-valued variables must be after `.i` and `.o`
`.ob [s1] [s2]... [sn]`	Names of the output functions must be after `.i` and `.o`
`.type f`	Following table describes the ON set; DC set is empty
`.type fd`	Following table describes the ON set and DC set
`.type fr`	Following table describes the ON set and OFF set
`.type fdr`	Following table describes the ON set, OFF set, and DC set.
`.e`	Optional, marks the end of the PLA description.

The format of the plane part of the input and output files for `espresso`

Plane	Character	Explanation
I	`1`	The input literal appears in the product term
I	`0`	The input literal appears complemented in the product term
I	`-`	The input literal does not appear in the product term
O	`1` or `4`	This product term appears in the ON set
O	`0`	This product term appears in the OFF set
O	`2` or `-`	This product term appears in the don’t care set
O	`3` or `~`	No meaning for the value of this function

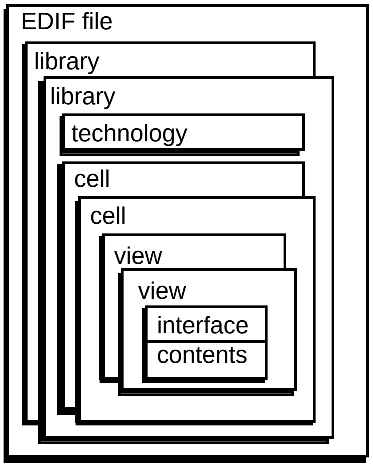
9.4 EDIF

Key terms: electronic design interchange format (EDIF) • EDIF version 2 0 0 • EDIF 3 0 0 handles buses, bus rippers, and buses across schematic pages • EDIF 4 0 0 includes new extensions for PCB and multichip module (MCM) data • Library of Parameterized Modules (LPM) • Electronic Industries Association (EIA) • ANSI/EIA Standard 548-1988

9.4.1 EDIF Syntax

Key terms: EDIF looks like Lisp or Postscript • a "write-only" language • `(keywordName {form})` • keywords • forms • "define before use" • identifiers • `&clock`, `Clock`, and `clock` are the same • `(e 14 -1)` is 1.4 • scale factor • `technology` section • `numberDefinition` • `scale` • `"A quote is % 34 %"` is a string with an embedded double-quote character

The hierarchical nature of an EDIF file

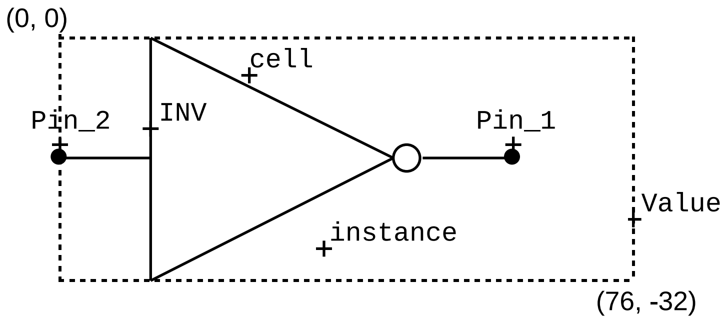


9.4.2 An EDIF Netlist Example

EDIF file for the `halfgate` **netlist**

```
 (edif halfgate_p
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap
  (keywordLevel 0))
 (status
  (written
   (timeStamp 1996 7
10 22
5 10)
   (program "COMPASS
Design Automation --
EDIF Interface"
    (version "v9r1.2
last updated 26-Mar-
96"))
   (author
"mikes")))
 (library xc4000d
  (edifLevel 0)
  (technology
   (numberDefinition
)
   (simulationInfo
    (logicValue  H)
    (logicValue
L)))
  (cell
   (rename INV
"inv")
   (cellType
GENERIC)
   (view
COMPASS_mde_view
    (viewType NETLIST)
    (interface
     (port I
      (direction
INPUT))
     (port O
      (direction
OUTPUT))
     (designator
"@@Label")))))
 (library working
  (edifLevel 0)
  (technology
   (numberDefinition )
   (simulationInfo
    (logicValue  H)
    (logicValue  L)))
  (cell
   (rename HALFGATE_P
"halfgate_p")
   (cellType GENERIC)
   (view
COMPASS_nls_view
    (viewType NETLIST)
    (interface
     (port myInput
      (direction
INPUT))
     (port myOutput
      (direction
OUTPUT))
     (designator
"@@Label"))
    (contents
     (instance B1_i1
      (viewRef
COMPASS_mde_view
       (cellRef INV
        (libraryRef
xc4000d))))
     (net myInput
      (joined
       (portRef
myInput)
       (portRef I
        (instanceRef
B1_i1))))
     (net myOutput
      (joined
       (portRef
myOutput)
       (portRef O
        (instanceRef
B1_i1))))
     (net VDD
      (joined ))
     (net VSS
      (joined ))))))
 (design HALFGATE_P
  (cellRef HALFGATE_P
   (libraryRef
working))))
```

9.4.3 An EDIF Schematic Icon



An EDIF view of an inverter icon

The coordinates shown are in EDIF units. The crosses that show the text location origins and the dotted bounding box do not print as part of the icon.

9.4.4 An EDIF Example

EDIF file for a standard-cell schematic icon

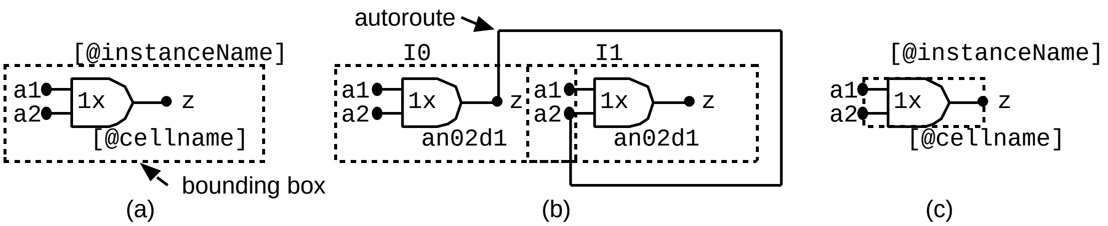
```
(edif pvsc370d
 (edifVersion 2 0 0)
 (edifLevel 0)
 (keywordMap
  (keywordLevel 0))
 (status
  (written
   (timeStamp 1993 2 9 22
38 36)
   (program "COMPASS"
    (version "v8"))
   (author "mikes")))
 (library pvsc370d
  (edifLevel 0)
  (technology
   (numberDefinition )
   (figureGroup
connector_FG
    (color 100 100 100)
    (textHeight 30)
    (visible
     (true )))
   (figureGroup icon_FG
    (color 100 100 100)
    (textHeight 30)
    (visible
     (true )))
   (figureGroup
instance_FG
    (color 100 100 100)
    (textHeight 30)
    (visible
     (true )))
   (figureGroup net_FG
    (color 100 100 100)
    (textHeight 30)
    (visible
     (true )))
   (figureGroup bus_FG
    (color 100 100 100)
    (textHeight 30)
    (visible
     (true ))
    (pathWidth 4)))
  (cell an02d1
```

```
   (cellType GENERIC)
   (view Icon_view
    (viewType SCHEMATIC)
    (interface
     (port A2
      (direction INPUT))
     (port A1
      (direction INPUT))
     (port Z
      (direction OUTPUT))
     (property label
      (string ""))
     (symbol
     (portImplementation
       (name A2
        (display
connector_FG
         (origin
          (pt -5 1))))
       (connectLocation
        (figure
connector_FG
         (dot
          (pt 0 0)))))
     (portImplementation
       (name A1
        (display
connector_FG
         (origin
          (pt -5 21))))
       (connectLocation
        (figure
connector_FG
         (dot
          (pt 0 20)))))
     (portImplementation
       (name Z
        (display
connector_FG
         (origin
          (pt 60 15))))
       (connectLocation
        (figure
connector_FG
         (dot
          (pt 60 10)))))
```

```
      (figure icon_FG
       (path
        (pointList
         (pt 0 20)
         (pt 10 20)))
       (path
        (pointList
         (pt 0 0)
         (pt 10 0)))
       (path
        (pointList
         (pt 10 -5)
         (pt 10 25)))
       (path
        (pointList
         (pt 10 -5)
         (pt 30 -5)))
       (path
        (pointList
         (pt 10 25)
         (pt 30 25)))
       (path
        (pointList
         (pt 45 10)
         (pt 60 10)))
       (openShape
        (curve
         (arc
          (pt 30 -5)
          (pt 45 10)
          (pt 30 25)))))
      (boundingBox
       (rectangle
        (pt -15 -28)
        (pt 134 27)))
      (keywordDisplay
instance
       (display icon_FG
        (origin
         (pt 20 29))))
      (propertyDisplay
label
       (display icon_FG
        (origin
```

Compass and corresponding Cadence `figureGroup` names

Compass name	Cadence name	Compass name	Cadence name
connector_FG	pin	net_FG	wire
icon_FG	device	bus_FG	not used
instance_FG	instance		



The bounding box problem

(a) The original bounding box for the an02d1 icon

(b) Problems in Cadence Composer due to overlapping bounding boxes

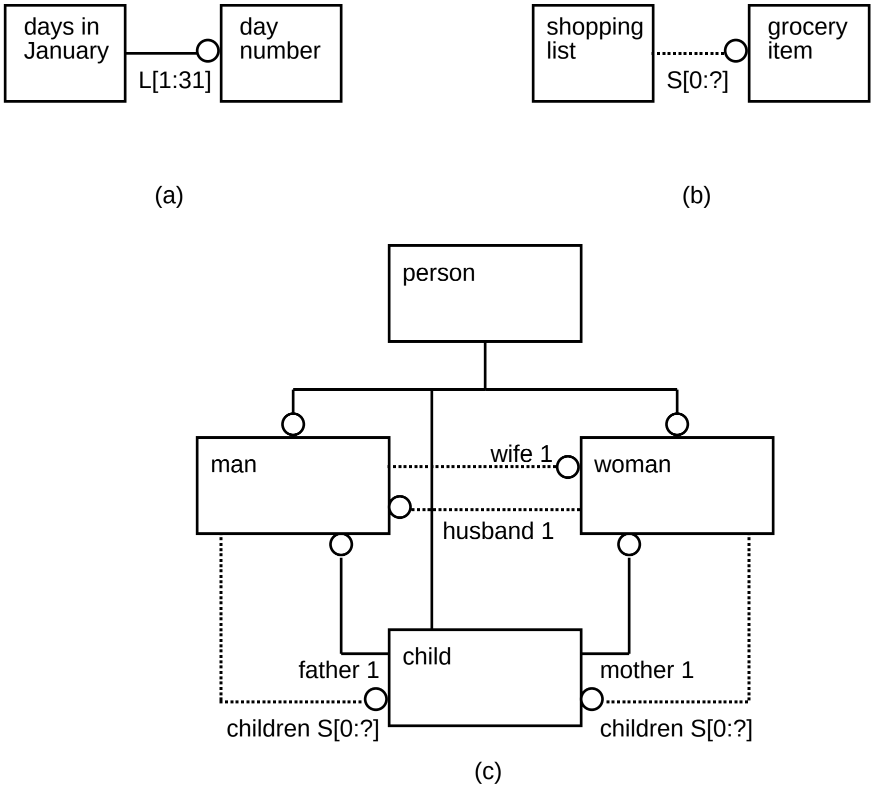
(c) A “shrink-wrapped” bounding box created using Cadence SKILL

9.5 CFI Design Representation

Key terms: CAD Framework Initiative (CFI) • design representation (DR) • information model (IM) • CFI started as an attempt to standardize schematic entry • CFI ended up as an attempt to close the stable door after the horse had bolted

9.5.1 CFI Connectivity Model

Key terms: EXPRESS language • EXPRESS-G • schema • Base Connectivity Model (BCM) • five-box model • an elegant method to represent complex notions



Examples of EXPRESS-G

(a) Each day in January has a number from 1 to 31

(b) A shopping list may contain a list of items

(c) An EXPRESS-G model for a family:

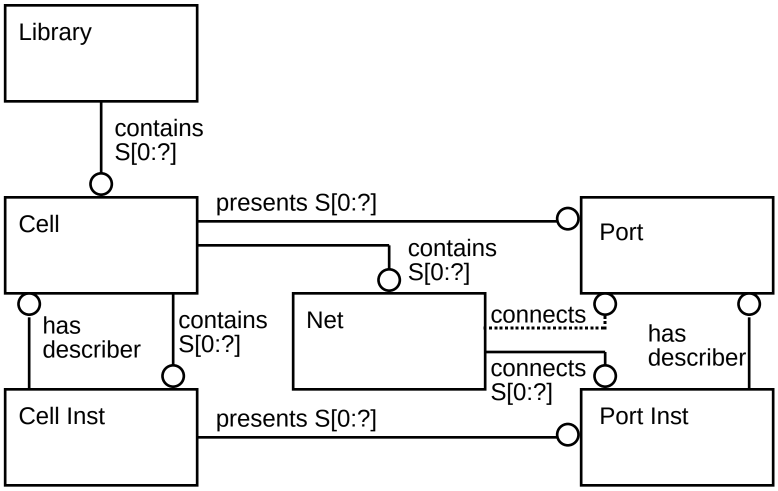
“Men, women, and children are people.”

“A man can have one woman as a wife, but does not have to.”

“A wife can have one man as a husband, but does not have to.”

“A man or a woman can have several children.”

“A child has one father and one mother.”



The original “five-box” model of electrical connectivity. (There are actually six boxes or types in this figure; the Library type was added later.)

“A library contains cells.”

“Cells have ports, contain nets, and can contain other cells.”

“Cell instances are copies of a cell and have port instances.”

“A port instance is a copy of the port in the library cell.”

“You connect to a port using a net.”

“Nets connect port instances together.”

```
SCHEMA family_model;
 ENTITY person
   ABSTRACT SUPERTYPE OF (ONEOF (man, woman, child));
   name: STRING;
   date of birth: STRING;
 END_ENTITY;


 ENTITY man
   SUBTYPE OF (person);
   wife: SET[0:1] OF woman;
   children: SET[0:?] OF child;
 END_ENTITY;


 ENTITY woman
   SUBTYPE OF (person);
   husband: SET[0:1] OF man;
   children: SET[0:?] OF child;
 END_ENTITY;


 ENTITY child
   SUBTYPE OF (person);
   father: man;
   mother: woman;
 END_ENTITY;
END_SCHEMA;
```

9.6 Summary

Key concepts:

Schematic entry using a cell library

Cells and cell instances, nets and ports

Bus naming, vectored instances in datapath

Hierarchy

Editing cells

PLD languages: ABEL, PALASM, and CUPL

Logic minimization

The functions of EDIF

CFI representation of design information