

Propuesta inicial

David Kutugata A01138865

Humberto Acosta A01195234

19 de Marzo, 2015

ITESM



# Visión

El objetivo del lenguaje es proporcionar una herramienta de diseño y simulación de circuitos digitales para educación y revisión de calidad y funcionamiento de una arquitectura digital.

# Objetivo

El lenguaje es orientado a la educación, servirá para que un alumno o profesor pueda diseñar y simular circuitos digitales con una herramienta fácil de usar, ligera para el CPU y que no cree archivos.

**Justificación:**

La herramienta usada típicamente es el ISE Designe Suite de Xilinx, que aunque es muy completa, pesa más de 17 Gigabytes, tarda alrededor de un minuto para compilar cualquier código y cree muchos archivos para ejecución que no son analizados.

**Ejemplo:**

Éstos son los archivos creados por el siguiente código:

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity ADDER is

port(

A: in std\_logic\_vector(n-1 downto 0);

B: in std\_logic\_vector(n-1 downto 0);

carry: out std\_logic;

sum: out std\_logic\_vector(n-1 downto 0)

);

end ADDER;

architecture behv of ADDER is

signal result: std\_logic\_vector(n downto 0);

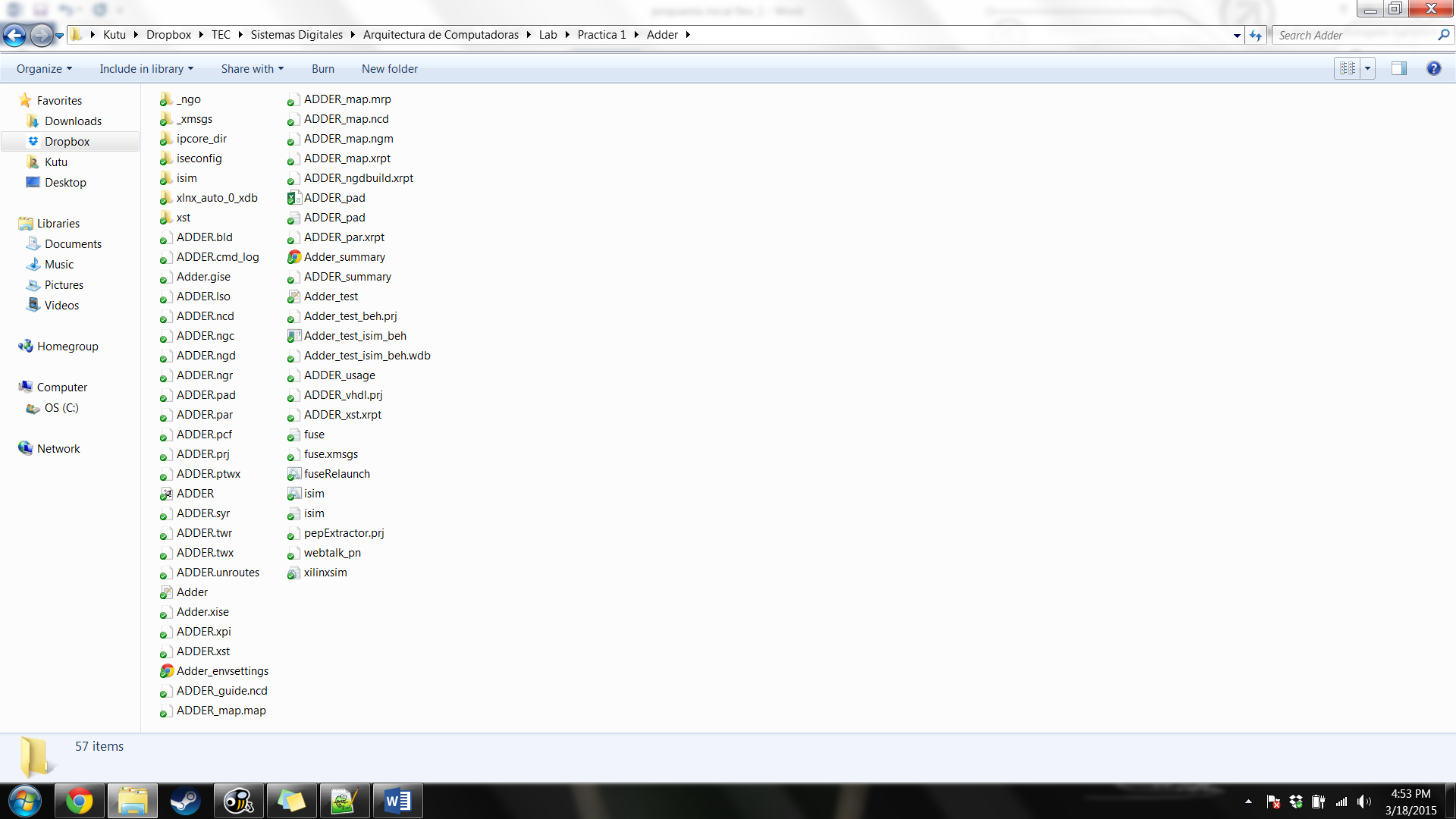
begin

result <= ('0' & A)+('0' & B);

sum <= result(n-1 downto 0);

carry <= result(n);

end behv;



# Requerimientos

## Léxico

Ejemplo del archivo lex:

%%

"entity" return entity;

"is" return is;

"port" return port;

"in" return in;

"out" return out;

"signal" return signal;

"bit" return bit;

"vector" return vector;

"to" return to;

"end" return end;

"architecture" return architecture;

"of" return of;

"begin" return begin;

"if" return IF;

"then" return THEN;

"else" return ELSE;

"case" return CASE;

"when" return WHEN;

"break" return BREAK;

"other" return OTHER;

"(" return yytext[0];

")" return yytext[0];

":" return yytext[0];

";" return yytext[0];

"==" return yytext[0];

"<<" return yytext[0];

">>" return yytext[0];

"!=" return yytext[0];

"=" return yytext[0];

"<" return yytext[0];

">" return yytext[0];

"&" return yytext[0];

"|" return yytext[0];

"!" return yytext[0];

"^" return yytext[0];

"+" return yytext[0];

"-" return yytext[0];

[a-zA-Z][\_a-zA-Z0-9]\* {yylval.id = yytext[0]; return ID;}

[0-1]+ {yylval.integer = atoi(yytext); return BYTE;}

-?[0-9]+ {yylval.integer = atoi(yytext); return CTE\_INT;}

[ \t\n] ;

. {ECHO; yyerror("ERROR\n");}

%%

## Sintaxis

<PROGRAMA>

VARS

BLOQUE

<VAR>

INOUT

TIPO

<VARS>

VAR

<TIPO>

<INOUT>

<ESTATUTO>

ASGINACION

CONDIF

CONDCASE

<ASIGNACION>

EXP

<BLOQUE>

ESTATUTO

<CONDIF>

ESTATUTO

EXPRESION

ESTATUTO

<CONDCASE>

ESTATUTO

<FACTOR>

EXP

VAR\_CTE

<VAR\_CTE>

ID\_VECTOR

<EXPRESION>

EXP

<EXP>

LOGIC

<LOGIC>

TERMINO

<TERMINO>

FACTOR

<ID\_VECTOR>

## Código

PROGRAM :VARS BLOQUE {;}

;

VARS :entity ID is port '(' VAR1 ')' end ID ';' {;}

;

VAR1 :VAR {;}

|VAR VAR1 {;}

;

VAR :ID ':' INOUT TIPO ';' {;}

;

INOUT :in {;}

|out {;}

|signal {;}

;

TIPO :vector '(' CTE\_INT ')' {;}

|bit {;}

;

BLOQUE :architecture ID of ID is begin ESTATUTO1 end ID ';' {;}

;

ESTATUTO1 :ESTATUTO {;}

|ESTATUTO ESTATUTO1 {;}

;

ESTATUTO :ASIGNACION {;}

|CONDIF {;}

|CONDCASE {;}

;

ASIGNACION :ID '=' EXP ';' {updateSymbolVal($1,$3);}

;

CONDIF :IF '(' EXPRESION ')' THEN ESTATUTO1 end IF ';' {;}

|IF '(' EXPRESION ')' THEN ESTATUTO1 ELSE ESTATUTO1 end IF ';' {;}

;

CONDCASE :CASE ID is CONDCASE1 end CASE ';' {;}

;

CONDCASE1 :WHEN bit ':' ESTATUTO1 BREAK ';' {;}

|WHEN vector ':' ESTATUTO1 BREAK ';' {;}

|WHEN OTHER ':' ESTATUTO1 BREAK ';' {;}

|WHEN bit ':' ESTATUTO1 BREAK ';' CONDCASE1 {;}

|WHEN vector ':' ESTATUTO1 BREAK ';' CONDCASE1 {;}

|WHEN OTHER ':' ESTATUTO1 BREAK ';' CONDCASE1 {;}

;

EXPRESION :EXP {;}

|EXP '==' EXP {$$ = $1 == $3;}

|EXP '!=' EXP {$$ = $1 != $3;}

|EXP '<' EXP {$$ = $1 < $3;}

|EXP '>' EXP {$$ = $1 > $3;}

;

EXP :LOGIC1 {;}

;

LOGIC1 :LOGIC {;}

|'!' LOGIC {$$ = ~$2; printf("negar\n");}

|LOGIC '&' LOGIC1 {$$ = $1 & $3; printf("and\n");}

|LOGIC '|' LOGIC1 {$$ = $1 | $3;}

|LOGIC '^' LOGIC1 {$$ = $1 ^ $3;}

;

LOGIC :TERMINO1 {;}

;

TERMINO1 :TERMINO {;}

|TERMINO '+' TERMINO1 {$$ = $1 + $3;printf("suma\n");}

|TERMINO '-' TERMINO1 {$$ = $1 - $3;}

;

TERMINO :FACTOR1 {;}

;

FACTOR1 :FACTOR {;}

|FACTOR '<<' FACTOR1 {$$ = $1 << $3;}

|FACTOR '>>' FACTOR1 {$$ = $1 >> $3;}

;

FACTOR :'(' EXP ')' {$$ = $2;}

|VAR\_CTE {;}

;

VAR\_CTE :ID {;}

|bit {;}

|vector {;}

;

## semántica

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | N/T | | entity | | is | port | | in | out | | signal | | | | bit | | vector | | | to | | end | | architecture | | | | | of | | begin | | | if | then | | | else | | case |
| 1 | | PROGRAM | | 2 | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | 7 | | | | | x | | x | | | x | x | | | x | | x |
| 2 | | VARS | | 2 | | 2 | 2 | | x | x | | x | | | | x | | x | | | x | | 2 | | x | | | | | x | | x | | | x | x | | | x | | x |
| 3 | | VAR | | x | | x | x | | 4 | 4 | | 4 | | | | 5 | | 5 | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 4 | | INOUT | | x | | x | x | | 3 | 3 | | 3 | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 5 | | TIPO | | x | | x | x | | x | x | | x | | | | 3 | | 5 | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 6 | | BLOQUE | | x | | 6 | x | | x | x | | x | | | | x | | x | | | x | | 6 | | 6 | | | | | 6 | | 7 | | | x | x | | | x | | x |
| 7 | | ESTATUTO | | x | | x | x | | x | x | | x | | | | x | | 8 | | | x | | x | | x | | | | | x | | x | | | 9 | x | | | x | | 10 |
| 8 | | ASIGNACION | | x | | x | x | | x | x | | x | | | | x | | 8 | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 9 | | CONDIF | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | 9 | | x | | | | | x | | x | | | 9 | 7 | | | 7 | | x |
| 10 | | CONDCASE | | x | | 10 | x | | x | x | | x | | | | x | | 10 | | | x | | 10 | | x | | | | | x | | x | | | x | x | | | x | | 10 |
| 11 | | EXPRESION | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 12 | | EXP | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 13 | | LOGIC | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 14 | | TERMINO | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 15 | | FACTOR | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
| 16 | | VAR\_CTE | | x | | x | x | | x | x | | x | | | | x | | x | | | x | | x | | x | | | | | x | | x | | | x | x | | | x | | x |
|  | N/T | | when | | break | | | other | | ( | ) | | : | ; | == | | << | | >> | != | | = | | < | > | & | | | ! | ^ | | + | | - | EoF | | | id | byte | | cte\_int | | |
| 1 | PROGRAM | | x | | x | | | x | | x | x | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | x | x | | x | | |
| 2 | VARS | | x | | x | | | x | | 3 | 2 | |  | 1 | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 2 | x | | x | | |
| 3 | VAR | | x | | x | | | x | | x | x | | 3 | 2 | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 3 | x | | x | | |
| 4 | INOUT | | x | | x | | | x | | x | x | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | x | x | | x | | |
| 5 | TIPO | | x | | x | | | x | | 5 | 3 | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | x | x | | 5 | | |
| 6 | BLOQUE | | x | | x | | | x | | x | x | | x | 1 | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 6 | x | | x | | |
| 7 | ESTATUTO | | x | | x | | | x | | x | x | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 8 | x | | x | | |
| 8 | ASIGNACION | | x | | x | | | x | | x | x | | x | 6 | x | | x | | x | x | | 12 | | x | x | x | x | x | x | | x | | x | x | | | 8 | x | | x | | |
| 9 | CONDIF | | x | | x | | | x | | 11 | 9 | |  | 6 | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | |  | x | | x | | |
| 10 | CONDCASE | | 10 | | 10 | | | 10 | | x | x | | 7 | 6 | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 10 | x | | x | | |
| 11 | EXPRESION | | x | | x | | | x | | 12 | x | | x | x | 11 | | x | | x | 11 | | x | | 11 | 11 | x | x | x | x | | x | | x | x | | | 12 | 12 | | x | | |
| 12 | EXP | | x | | x | | | x | | 13 | x | | x | x | x | | x | | x | x | | x | | x | x | 12 | 12 | 12 | 12 | | x | | x | x | | | 13 | 13 | | x | | |
| 13 | LOGIC | | x | | x | | | x | | 14 | x | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | 13 | | 13 | x | | | 14 | 14 | | x | | |
| 14 | TERMINO | | x | | x | | | x | | 15 | x | | x | x | x | | 14 | | 14 | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 15 | 15 | | x | | |
| 15 | FACTOR | | x | | x | | | x | | 12 | 14 | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 16 | 16 | | x | | |
| 16 | VAR\_CTE | | x | | x | | | x | | x | x | | x | x | x | | x | | x | x | | x | | x | x | x | x | x | x | | x | | x | x | | | 15 | 15 | | x | | |
|  |  | |  | |  | | |  | |  |  | |  |  |  | |  | |  |  | |  | |  |  |  |  |  |  | |  | |  |  | | |  |  | |  | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | = | == | << | >> | != | < | > | & | | | ! | ^ | + | - |
| bit | bit | bit | bit | bit | bit | bit | bit | bit | bit | bit | X | bit | vector | bit |
| bit | vector | x | bit | vector | vector | bit | bit | bit | vector | vector | X | vector | vector | vector |
| vector | bit | vector | bit | vector | vector | bit | bit | bit | vector | vector | X | vector | vector | vector |
| vector | vector | vector | bit | vector | vector | bit | bit | bit | vector | vector | X | vector | vector | vector |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | = | == | << | >> | != | < | > | & | | | ! | ^ | + | - |
| in | in | x | in | in | in | in | in | in | in | in | in | in | in | in |
| in | out | x | x | x | x | x | x | x | x | x | x | x | x | x |
| in | signal | x | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal |
| out | in | out | x | x | x | x | x | x | x | x | x | x | x | x |
| out | out | out | x | x | x | x | x | x | x | x | x | x | x | x |
| out | signal | out | x | x | x | x | x | x | x | x | x | x | x | x |
| signal | in | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal |
| signal | out | x | x | x | x | x | x | x | x | x | x | x | x | x |
| signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal | signal |

## Funciones especiales

No se manejan funciones especiales, sin embargo, aparte de compilar el código, se tendrá la opción de agregar valores de entrada para simulaciones.

## Tipos de datos

-bit: elemento binario

-vector: cadena de elementos binarios

-in: dato de entrada o control

-out: dato de salida o de despliegue

-signal: dato para señales intermedias

# Plataforma de desarrollo

Será desarrollado en C++, utilizando las herramientas flex para léxico y bison para sintáxis.

# Bibliografía

<http://www.cplusplus.com/>

<http://www.xilinx.com/>