# TRADUCTOR DE UN LENGUAJE ESTRUCTURADO A DESCRIPCION ELEMENTAL DE HARDWARE



H. ANTILLANCA E.*  R. VALLEJOS C**

RESUMEN

Se diseñó un traductor, cuyo código fuente está escrito en un lenguaje de alto nivel, apropiado para describir 'hardware', y el código objeto generado es el listado de los circuitos e interconexiones de una máquina digital, que funcionalmente resuelve el mismo problema del programa fuente.

También se diseñó el lenguaje de programación. Esto se hizo mediante modificaciones al lenguaje Pascal -escrito en castellano- de forma tal que permitiera incorporar las estructuras de datos propias de los circuitos digitales.

La máquina digital, obtenida automáticamente por el traductor, está basada en una unidad de control microprogramada. Los microprogramas necesarios también son generados por el traductor.

ABSTRACT

A translator was designed, such that its source code is written in a high -level language, appropriate for "hardware" description, and its generated object code is the printout, or listing, of the circuits and interconnections of a digital machine which functionally solves the same problem of the source program.

Furthermore, the programming language was designed. This was done by modifying the Pascal language - written in Spanish - in such a form as to permit the incorporation of the digital circuits proper data structures.

The digital machine automatically obtained by the translator, is based upon a microprogrammed control unit. The necesary microprograms are also generated by the translator.

* Ingeniero en Electrónica (UCV, Chile, 1980), egresado de Magister en Sistemas Digitales (UTFSM, Chile, 1983). Universidad Técnica Federico Santa María, Departamento de Electrónica, Casilla 110-V, Valparaíso-CHILE.

** Ingeniero Civil Electrónico (UTFSM, Chile, 1976). Profesor del Departamento de Electrónica, Universidad Técnica Federico Santa María, casilla 110-V, Valparaíso-CHILE.

## INTRODUCCION

El diseño de los sistemas digitales ha evolucionado conforme lo ha hecho la tecnolología de los componentes, desde la integración de las compuertas básicas, donde el diseño implicaba usar ecuaciones booleanas y diagramas lógicos, hasta la integración de circuitos lógicos complejos (circuitos MSI/LSI/VLSI) cuya modularidad permite ahora al diseñador escribir las máquinas digitales mediante abstracciones de componentes y Lenguajes de Descripción de Hardware (HDLs).

Así es como ha nacido una gran variedad de HDLs. Muchas publicaciones muestran las discusiones y los esfuerzos realizados por los autores para lograr una uniformidad en pro de especificar claramente una máquina digital en todos los diferentes niveles de representación posibles [1,2,3,4,5,6,7].

En forma paralela al nacimiento de los HDLs se ha agregado, al diseño de los sistemas digitales, el uso de una herramienta poderosa como lo es el diseño apoyado por computador (CAD), y en consecuencia producir ambientes computacionales de diseño [8], que apuntan sus esfuerzos a la simulación [9] y a la síntesis automática de sistemas digitales [10,11].

El presente trabajo aprovecha la evolución ocurrida en los lenguajes de programación de computadores, con el objeto de diseñar máquinas digitales. La idea es incorporar al diseño de hardware los conceptos de modularidad y las metodologías jerárquicas de diseño que posee el diseño de software. Ejemplo claro es el lenguaje de programación Pascal, el cual tiene la ventaja de permitir al usuario diseñar programas estructurados usando sentencias de control de flujo de alto nivel [12,13].

El objetivo ha sido evaluar una técnica de diseño automático de máquinas digitales apoyado por computador, para lo cual se ha construido un traductor capaz de generar una representación elemental de la máquina (alambrado), a partir de una descripción inicial de ésta en términos de un HDL de alto nivel y a la vez estructurado. En consecuencia, diseñar una máquina digital es equivalente a resolver un problema específico escribiendo un programa en ese lenguaje.

## 1. DESCRIPCION DEL TRABAJO REALIZADO

El trabajo consta de dos partes:

1) Definición de un Lenguaje de Descripción de 'Hardware' de alto nivel con sentencias de control estructuradas, y

2) Construcción de un Traductor.

## 1.1. El Lenguaje de Descripción de Hardware de Alto Nivel

El HDL diseñado, llamado lenguaje DMD (Descripción de Máquinas Digitales), es una variación del lenguaje de programación de alto nivel llamado Pascal, pero orientado a satisfacer las necesidades de describir sistemas digitales. La sintaxis completa del lenguaje DMD se presenta en el Apéndice A.

El nivel del lenguaje es tal que no se preocupa tanto del control de la máquina sino del aspecto funcional. Esto se hace mediante las sentencias de control de flujo estructuradas. De esta forma, el diseñador se desliga del aspecto control de la máquina y puede, usando las facilidades de modularidad de las sentencias, definir el hardware en base a los requerimientos de software. La parte control es un problema que resuelve el traductor.

El lenguaje DMD exige del diseñador especificar:

a) Los tipos y variables a usar, llamada Parte Declaraciones, y

b) La descripción del programa, llamada Parte Bloque.

### 1.1.1. Parte Declaraciones

La parte declaraciones considera las declaraciones de tipo y las declaraciones variables.

La declaración de tipo permite definir, sobre ciertos recursos, atributos que posibilitan al traductor chequear la legalidad y compatibilidad de las construcciones realizadas. Así, por ejemplo, en la mayoría de los lenguajes se pueden definir las variables tipo enteros, y cualquier operación en que el resultado deba ser entero y se opere, por ejemplo, con una variable booleana, es ilegal, sin embargo se aceptan ciertas operaciones con variables de tipo real.

En el lenguaje DMD, se puede definir, por ejemplo, variables tipo contador, pero no se puede aplicar sobre ellas operaciones aplicables solo a variables de tipo registro, como por ejemplo operaciones de corrimientos.

El tipo básico del lenguaje DMD es el tipo booleano, sin embargo, no es

declarable explítamente. Este tipo es el átomo que compone todas las variables declarables y, en consecuencia, todas pueden ser tratadas como variables booleanas en su nivel mínimo.

Los tipos básicos declarables explícitamente son aquellos tipos que permiten identificar el recurso de hardware que se asocia a la variable del tipo dado. Estos tipos son: tipo flip-flop, tipo registro, tipo contador, tipo terminal y tipo memoria.

1.1.2. Parte bloque

Corresponde a la parte en que se describe el funcionamiento de la máquina mediante sentencias. Hay dos tipos de sentencias:

a) Las que efectúan el control del programa
b) Las que indican una acción sobre el recurso

Las sentencias de control utilizables son:

- COMIENZO-FINAL
- SI-ENTONCES-SINO
- MIENTRAS-HAGA
- REPITA-HASTA

Las sentencias que indican acción sobre un recurso corresponden a:

- ASIGNACION (Transferencias entre registros)
- PRIMITIVAS (Operaciones sobre un recurso específico)

1.1.3. Ejemplo Ilustrativo

A continuación se presenta un ejemplo que muestra la filosofía de diseñar con DMD:

```
MAQUINA PARIDAD;

(* Este es un comentario.  El traductor lo ignora *)
(* El siguiente programa es la descripción autocontenida de una máquina
simple que determina, en forma serie, la paridad de un registro *).
```

```
TIPOS BIT       : FLIPFLOP;
      CONT      : CONTADOR DE 4 BITS;
      REG       : REGISTRO DE 8 BITS;

VAR   A  : REG;
      CC:  CONT;
      BP:  BIT;
      FIN: BIT;

COMIENZO
A:='3F';
LIMPIE(CC);
LIMPIE(BP);
LIMPIE(FIN);
REPITA
   BP:=BP % A[8]; (* OR EXCLUSIVO ENTRE BP y A[8]*)
   DESPLZ(DER,A,A[8]); (*Desplazamiento cíclico a la derecha de A*)
   INCR(CC);
HASTA {CC='8'};
FIN:=BP;
FINAL.
```

1.2. El Traductor

El Traductor realiza básicamente las siguientes tareas:

a) Revisar la sintaxis del lenguaje DMD.

b) Analizar legalidad y compatibilidad de las operaciones sobre los recursos declarados.

c) Producir el listado con el alambrado de la máquina a nivel de hardware, esto es, los tipos de circuitos integrados necesarios, la cantidad de cada uno de ellos, y las interconexiones entre ellos.

d) Sintetizar operaciones booleanas en hardware.

e) Analizar situaciones que producen repeticiones de alambrado, a fin de evitarlo.

f) Generar el código del microprograma que controla la secuencia de operaciones

de la máquina. Esto se deduce de las sentencias de control de flujo del lenguaje DMD.

## 1.3. Arquitectura de la Máquina Generada

En el proceso de traducir el lenguaje DMD, es tarea del traductor generar la estructura que produce el control de la máquina descrita. La estructura generada consta de una unidad de control de microprograma (secuenciador), que controla la secuencia de obtención de las microinstrucciones de una memoria, y la memoria que almacena el microprograma que controla la máquina descrita. Esta estructura es lo único que se mantiene igual para todas las máquinas generadas, no así el microprograma que depende de los algoritmos de funcionamiento de las mismas.

La arquitectura de la máquina generada, que se observa en la figura 1, se puede descomponer en tres estructuras de hardware: la estructura de control -ya descrita-, la estructura objeto -sobre la cual se ejecuta el control-, esto es, aquellos recursos dependientes de las variables declaradas por el diseñador, y la estructura de interfaz entre la estructura de control y la estructura objeto -estructura compuesta por elementos de hardware que se asocian a la máquina objeto para que se haga efectivo el control sobre ella.

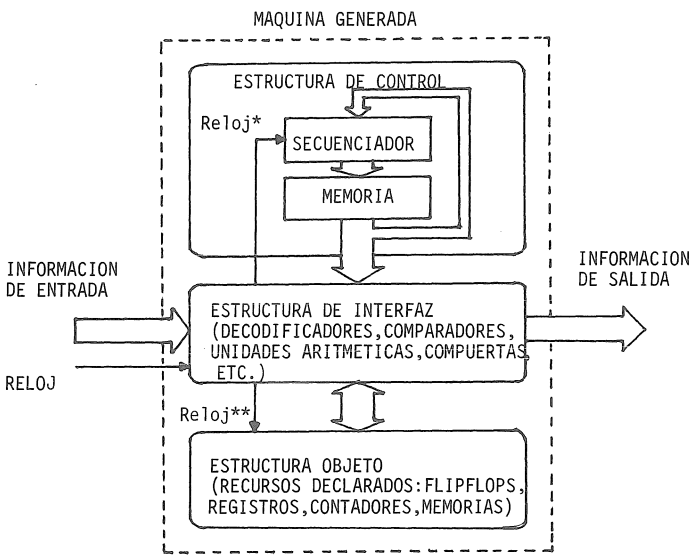


Fig. 1. Arquitectura de la Máquina Generada.

## 2. CONCLUSIONES

El uso del lenguaje DMD ha permitido profundizar, desde el punto de vista del diseño, en una conceptualización de los sistemas digitales. Con este lenguaje es posible enfrentar el diseño de hardware de la misma manera como se enfrenta el diseño de software, esto es, emplear las metodologías de diseño estructurado y 'Top-Down', permitiendo al diseñador desligarse de detalles estructurales de la máquina, y concentrarse fundamentalmente en los aspectos algorítmicos y de almacenamiento de la información.

## 3. REFERENCIAS

[1] M.R. Barbacci: "A Comparison of Register Transfer Languages for Describing Computer and Digital System", IEEE Trans., 1975, c-24, pp.137-150.

[2] G.J. Lipovski: "Hardware Description Languages: Voices from the Tower of Babel", Computer, 1977, 10, pp. 14-17.

[3] Harry F. Jordan and Burton J. Smith: "The Assignment Statement in Hardware Description Languages", Computer, 1977, 10, pp. 43-49.

[4] G.R. Peterson: "Teaching Digital System Design with a Hardware Description Languages", Proc., 1978 Frontier in Education Conference, ASEE/IEEE.

[5] "Instruction Set Processor Specifications (ISPS): The Notation and its Application", IEEE Trans. on Comput., Vol. c-30, Jan. 1981.

[6] C.A.G. Enstone, "HARTRAN: A language for Top-Down Digital System Design", GEC Hirst Research Centre, Wembley, Middlesex, England, 1981.

[7] Ed Thompson, A. Lekkos, Don Ross, R. von Blucher, "TEGAS Design Language (TDL) - A System for Modular Description and Top-Down/Bottom-Up Verification of LSI and VLSE". C. Computing System and Service, Inc., Austin, Texas, 1980.

[8] P.W. Foulk, R.A. Mason, "A Design Enviroment for Digital Hardware", Computer Engineering, Heriot-Watt University, Edinburgh, 1980.

[9] Melvin A. Breuer, Editor: "Digital System Design Automation: Language, Simulation & Data Base", Computer Science Press, Inc., Woodland Hills, California, 1975.

[10] S.W. Director, A.C. Parker, D.P. Siewiorek, and D.E. Thomas: "A Design Methodology and Computer Aids for Digital VLSI System", IEEE Trans. Circuits System, Vol. cas-28, July, 1981.

[11] Donald E. Thomas: "The Automatic Synthesis of Digital System", Proceeding of the IEEE, Vol. 69, October 1981.

[12] Dahl, Dijkstra and Hoare: "Structured Programming", Academic Press, 1978.

[13] Niklaus Wirth: "Algorithms+Data Structure=Program", Prentice-Hall, Inc.

[14] G.M. Baudet, M. Cutler, M. Davio, A.M. Peskin, F.J. Ramming: "The Relationship between HDLs and Programming Languages". Silver Spring, MD, USA: IEEE Comput. Soc. Press. Inc., 1983.

APENDICE A

SINTAXIS DEL LENGUAJE DMD (DESCRIPCION DE MAQUINAS DIGITALES)

El formalismo BNF (Bankus Naur Form) se usa a continuación para describir la sintaxis del presente lenguaje.

<Descripción de Máquina Digital>::=<encabezado de máquina> <bloque>

<encabezado de máquina>::=MAQUINA <identificador de máquina> ;
<identificador de máquina>::=<identificador>
<identificador>::=<letra>{<letra>|<dígito>}
<letra>::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<dígito>::=0|1|2|3|4|5|6|7|8|9

<bloque>::=<declaraciones><sentencia compuesta>

<declaraciones>::={<declaración de const>}
{<declaración de tipos>}
{<declaración de variables>}
{<declaración de submáquinas>}

<declaración de const>::=CONST <definición de const>{;<definición de const>};
<definición de const>::=<identificador de const>=<hexadecimal>
<identificador de const>::=<identificador>
<hexadecimal>::='<hexdígito>{<hexdígito>}'
<hexdígito>::=<dígito>|A|B|C|D|E|F

<declaración de tipos>::=TIPOS<definición de tipo>{;<definición de tipo>};
<definición de tipo>::=<identificador de tipo>=<tipo>
<identificador de tipo>::=<identificador>
<tipo>::=<tipo simple>|<tipo estructurado>
<tipo simple>::=<tipo básico>|<identificador de tipo>
<tipo básico>::=<flipflop>|<registro>|<contador>|<terminal>|<memoria>
<flipflop>::=FLIPFLOP
<registro>::=REGISTRO DE <dimensión del ancho> BITS
<contador>::=CONTADOR DE <dimensión del ancho> BITS
<terminal>::=TERMINAL DE <dimensión del ancho> BITS
<memoria>::=MEMORIA DE <dimensión del ancho>*<dimensión del largo> BITS
<tipo estructurado>::=<tipo arreglo>|<tipo record>
<tipo arreglo>::=ARREGLO [<entero>..<entero>] DE <tipo>

<entero>::=<dígito>{<dígito>}
<tipo record>::=RECORD <lista de campos> FINAL
<lista de campos>::=<definición de campo>{;<definición de campo>}
<definición de campo>::=<identificador de campo>:<tipo>

<declaración de variables>::=VAR <definición de var>{;<definición de var>};
<definición de var>::=<identificador de var>{,<identificador de var>}:<tipo>

<declaración de submáquinas>::=<definición de submáq>{;<definición de submáq>};
<definición de submáq>::=<encabezado de submáquina><bloque>
<encabezado de submáquina>::=SUBMAQ <identificador de submáquina>;
<identificador de submáquina>::=<identificador>

<sentencia compuesta>::=COMIENZO <sentencia>{;<sentencia> FINAL
<sentencia>::=<asignación>|<identificador de submáquina>|<primitiva>|

<sentencia si-entonces-sino>|<sentencia mientras-haga>|
<sentencia repita-hasta>|<sentencia con-haga>|
<sentencia desde-hasta-haga>|
<sentencia compuesta>

<asignación>::=<variable>:=<expresión>
<variable>::=<identificador de var><designador>
<designador>::={.<identificador de campo> [<dirección>]}
<dirección>::=<variable>:<entero>
<expresión>::=<constante>|<expresión simple>|<expresión booleana>
<constante>::=<identificador de constante>:<hexadecimal>
<expresión simple>::=<primer operando><operador><segundo operando>
<primer operando>::=<variable>|~<variable>
<segundo operando>::=<variable>|~<variable>
<operador>::=+|-|#|%|&
<expresión booleana>::=<función booleana>|<parizq><comparación><parder>
<función booleana>::=<término booleano>{#<término booleano>}
<término booleano>::=<subtérmino booelano>{%<subtérmino booleano>}
<subtérmino booleano>::=<factor booleano>{&<factor booleano>}
<factor booleano>::=<variable>|(<expresión booleana>)|~<factor booleano>
<parizq>::={
<parder>::=}
<comparación>::=<primer comparando><relación><segundo comparando>
<primer comparando>::=<variable>
<segundo comparando>::=<variable>:<constante>
<relación>::=>|<|=

<primitiva>::=<cuenta>|<desplaz>|<colocación>
<cuenta>::=<ident.primitiva cuenta>(<variable>)
<ident.primitiva cuenta>::=INCR | DECR
<desplaz>::=<ident.primitiva desplaz>(<sentido>,<variable>,<desde>)
<ident.primitiva desplaz>::=DESPLZ
<sentido>::=DER | IZQ
<desde>::=<dato simple>
<dato simple>::= Ø | 1 | <expresión booleana>
<colocación>::=<id. prim.coloc.>(<variable>)
<ident.primitiva coloc>::=LIMPIE PONGA

<sentencia si-entonces-sino>::=SI <expresión booleana>
ENTONCES <sentencia>
{SINO <sentencia>}
<sentencia mientras-haga>::=MIENTRAS <expresión booleana>
HAGA <sentencia>
<sentencia repita-hasta>::=REPITA <sentencia> HASTA <expresión booleana>

<sentencia con-haga>::=CON <variable> HAGA <sentencia>

<sentencia desde-hasta-haga>::=DESDE <asignación> HASTA <constante>
HAGA <sentencia>