A Versatile Netlist Generator for Fast Dataprocessing Slices Basing on Tree Representations

Gerd Karl Heinz
SICAN GmbH, Hannover
Germany

Dr. Ing. G. Heinz, SICAN GmbH, Garbsener Landstr. 10, W-3000 Hannover 21, Germany. Tel. +049 (511) 2795 242, Fax. 280

A VERSATILE NETLIST GENERATOR FOR FAST DATAPROCESSING SLICES BASING ON TREE REPRESENTATIONS

Gerd Karl Heinz
SICAN GmbH, Hannover
Germany

Introduction

Modern ASIC- CAD Tools - like CADENCE-, VTI- or LSI Logic- software - include powerful tools for ASIC- schematics and layout generation. So it is possible, to reduce the problem of module generation for datapath slices to that of netlist generation for variable types of netlists. The advantage is a rapidly increasing flexibility and versatility for datapath generation tools. For example VTI shows the efficiency of tools, generating 'allround' datapath processing elements with easy algorithms.
Our approach to datapath module generation is based on tree representations of standard cells working in order log(n) times. Tree representations in this sense means: tree structures at the algorithmical level and tree representations as a way to solve electrical fanout limitation problems at internal nodes for parametrisable structures.
Tree solutions are - compared with available ripple algorithms going in order const*n times (n represents the buswidth) very fast for larger busses (>12 bit). The resulting speed improvement is significant.
Compared with easy ripple representations, the design of tree representations is extremly difficult. Costs increase rapidly. To introduce datapath tree solutions into the ASIC- market, it is necessary, to design the tools independent of wafer fab technologies. So our tools refer to allround available, hierarchical useable standart cells as basic elements only.
The customer can use the tools like library elements: pick up the library menue, choose a datapath slice element (like adder, comparator, decoder), complete a fill form for slice parameters (buswidth, input-/output-/control- activities, maximum internal node fanout, functional specifications) press return- buttom, and the generated datapath slice symbol drags at the cursor and can be placed in the schematics.
The resulting area overhead compared with ripple solutions is not substancial. Datapath circuits in many chips occupy only small amounts of total chip area.
For the custumer the design costs decreases, because the chip designer gets modules, that work as fast as possible, in shortest periods of time. So the schematic design of a 32 bit incrementer circuit needs approximately 15 CPU seconds (VAX 11/780) , compared with some weeks to design the module by hand.
The algorithms are programmed in 'C' using a VMS environment.

Dr. G. Heinz, SICAN GmbH, Garbsener Landstr. 10, W- 3000 Hannover 21, Germany. Tel. +049 (511) 2795 242, Fax. 280

Increasing the speed of datapath elements for ASIC's demands

1) fast (O log n) algorithms (default)

2) small layout area (default)

3) mapping to different technologies (scaling ability)

4) small delays of *each* internal node (fanout optimum)

How to solve this problems?

Problem 1)
Usage of area and time optimal algorithms *only.*

Problem 2)
Routing areas cost a lot of space. The bus order is kept unchanged.

Problem 3)
Layout dependencies has to be avoided. Problems of algorithmic generation and (automated) schematic/ layout generation are strictly decoupled.

Problem 4)
Optimized fanouts at all internal and external nodes are generated from the algorithm.
A node (input, output, clock, enable...) is changed into a binary or higher order tree representation, if fanout increases.

VIEW AT SOME DETAILS...

1. INTRODUCTION

TASK

- Generation of *fast* and *parallel* computing circuits for ASICs

- *fast*	without ripple carry propagation
	node fanouts not allowed to increase
-*parallel*	unlimited buswidths theoretical possible

- Speed up the design time
- Support for high- level synthesis (?!)

PROBLEMS to solve

global	: unrestricted, variable buswidth
electrical	: delay- minimum = fanout- optimum of all nodes
algorithmic	: choice of fast algorithms

-> medium found: tree structures

MODULES

- Add_Sub	Incr, Decr, CLA, Carry Save Adder*
- Coder	Decoder, Encoder, Multiplexer
- Flags	Parity, Zero, Equal, Compare
- Mult_Div*	Carry_save_tree, Booth_Encoding, div_1/x
- Register	Addressed, Associative*, FIFO, Stack
- Shifter	Left, Right, Left&Right: arithm./logical, Barrel

VOLUME OF GENERATION

- hierarchical netlist
- relative placement
- module- datasheet
- module- help
- behavioral model*

TYPES OF MODULE GENERATORS (digital)

VARIABLE WIDTH

- ROM
- EPROM
- RAM
- PLA (PLD, SLA, D- Matrix)
- SLA
- Adder
- Multiplier (integer)
- Barrel Shifter
- FIFO & Stack
- Counter
- LFSR
- Parity Generation
- ALU
- MMU
- Filter

LIBRARY BASED

- Boolean Gates
- Sliced Datapath (vertical/ horizontal)
- 2901, 8051 CPU
- 2901, 8051 ALU
- Microcontroller
- Control Unit
- Bus Interface

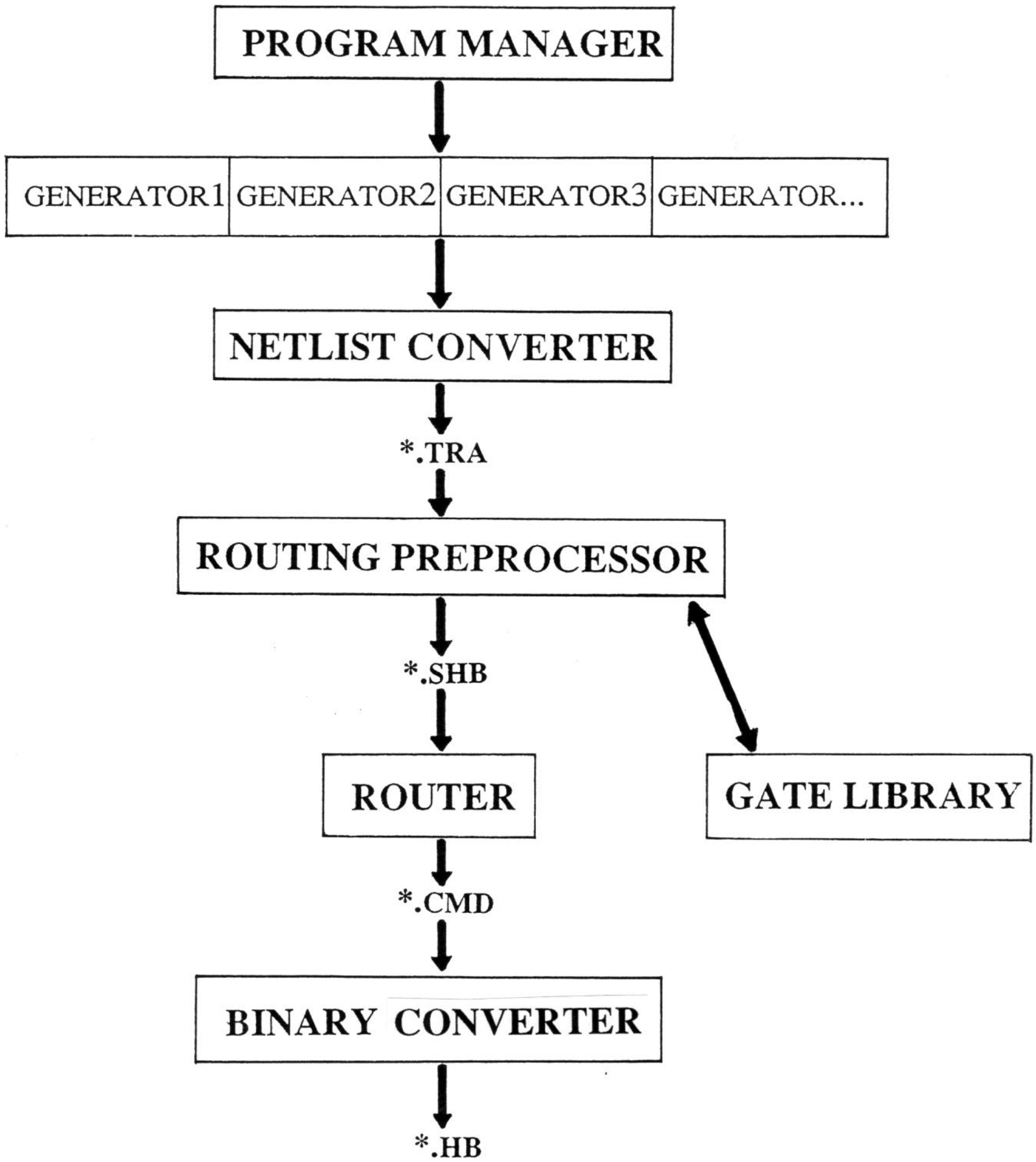
PRODUCTS (assortment weighted[2])

1. Concorde (Seattle Silicon Techn.)
2. Genesil (Silicon Compiler Systems)
3. Chipsmith (Lattice Logic)
4. Ltimate (Silicon Design Labs.)
5. Modular Design Environment (LSI Logic)
6. % (NCR Microelectronics)
7. VLSI Silicon Comp. Syst. (VLSI Technology)

⋮

[2]Source: E.L.Meyer, On Module Generation, VLSI Systems Design, March 1987

PROGRAM STRUCTURE


PROGRAM MANAGER
GENERATOR1
GENERATOR2
GENERATOR3
GENERATOR...
NETLIST CONVERTER
*.TRA
ROUTING PREPROCESSOR
*.SHB
ROUTER
GATE LIBRARY
*.CMD
BINARY CONVERTER
*.HB