DYNAMIC - A Java Based Toolset For Integrating Dynamic Logic Circuits Into A Standard VLSI Design FLow

A.Wassatsch, D.Timmermann

University of Rostock
Department of Electrical Engineering and Information Technology
Institute of Applied Microelectronics and Computer Science
Richard-Wagner-Str. 31, D-18119 Rostock, Germany
Tel./Fax.: ++49+381 498 3534/3601
wassatsch@e-technik.uni-rostock.de

INTERNATIONAL CADENCE USER GROUP CONFERENCE
September 10-13, 2000
San Jose, CA, USA

Outline

- Motivation
- Basics of dynamic circuit technologies
- Design requirements
- The Toolset
- Performance analysis
- Applications
- Conclusion

Motivation

- Why Java for EDA applications?
  - portability:
    - write once run everywhere
    - same version for all systems
    - reduction of the overall maintenance costs
  - and the runtime ?:
    - just in time (jit) compiler technologies delivers the power
    - careful implementation and optimization can further accelerate execution time
    - for the minority of high-end applications like logic synthesis of large designs the performance of conventional systems is by no way sufficient

Java is not the solution for everything, but already well for a majority of the given tasks in EDA !

Motivation (2.)

- Why replace CMOS with dynamic circuits ?
  - speed:
    - some of the fastest microprocessor designs like Compaq Alpha and the 1GHz Prototype from IBM utilize dynamic circuit technology
    - halved fan-in
  - area:
    - count of transistors for a n-input gate with register: $11 + n < 16 + 2n$ (dynamic circuit:CMOS)
    - but increased expense in the clock-tree network
  - power consumption:
    - reduced dependency on clock frequency compared to CMOS
    - current consumption is determined by the signal value and not by the rate of signal value changes
  - True Single Phase Clock (TSPC) as circuit technology for standard cells :
    - solid behavior of the cell signals
    - digital specification of the cell behavior possible

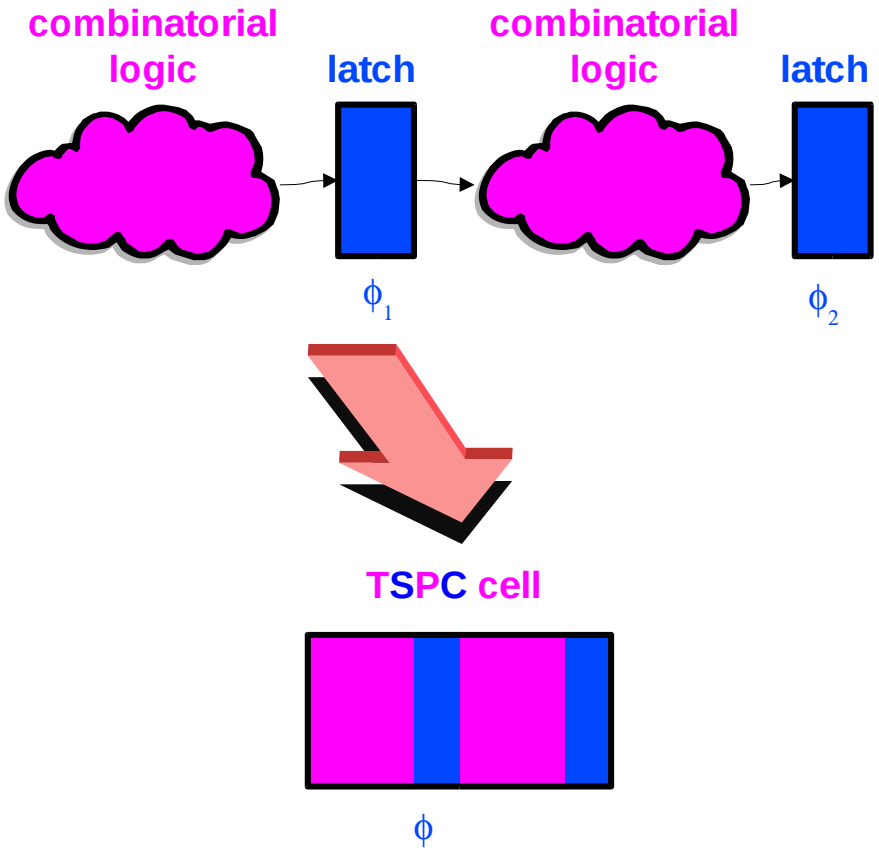
TSPC isn’t an universal remedy for every design !

Motivation (3.)

- Why do we have to expand the design flow?
  - there is at this time, no support through commercial tools for logic synthesis with dynamic circuits style standard cell libraries
  - differentiation of combinatorial and sequential cells, a syn lib must have (N)OR/(N)AND, INV and a register element
  - “Schematic entry” of the design $\rightarrow$ manual work, isn’t appropriate for a modern design flow
  - Workaround 1: structural HDL-description with respect to the demands of dynamic logic $\rightarrow$ like schematic-entry
  - Workaround 2: separate instantiation of logic and pipeline register file through the HDL-description, reordering of the pipeline after the logic synthesis through “balance-registers” $\rightarrow$ resulting netlist not applicable for TSPC circuit style

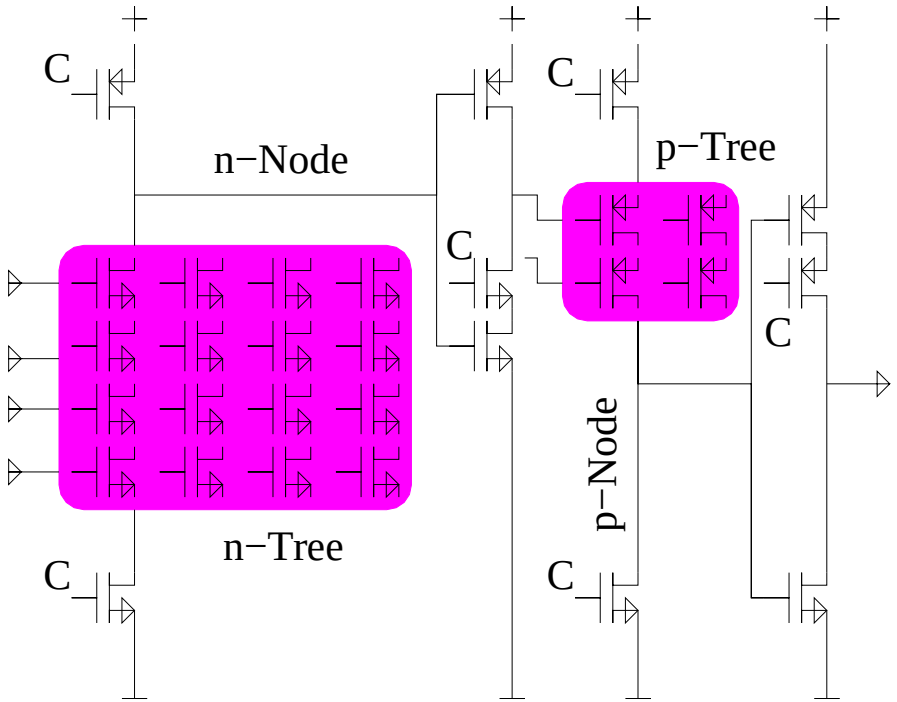
development of a design flow extension necessary !

Basics of dynamic circuits



- conflation of combinatorial and sequential elements into one cell
- no pure combinatorial blocks possible
- processing depends on the clock
- operational principle based on capacitive carrier storage
- logic function is implemented in only one tree of transistors
- examples: C^2MOS TSPC, Domino-Logic, CVSL, DCVSL, DCSL

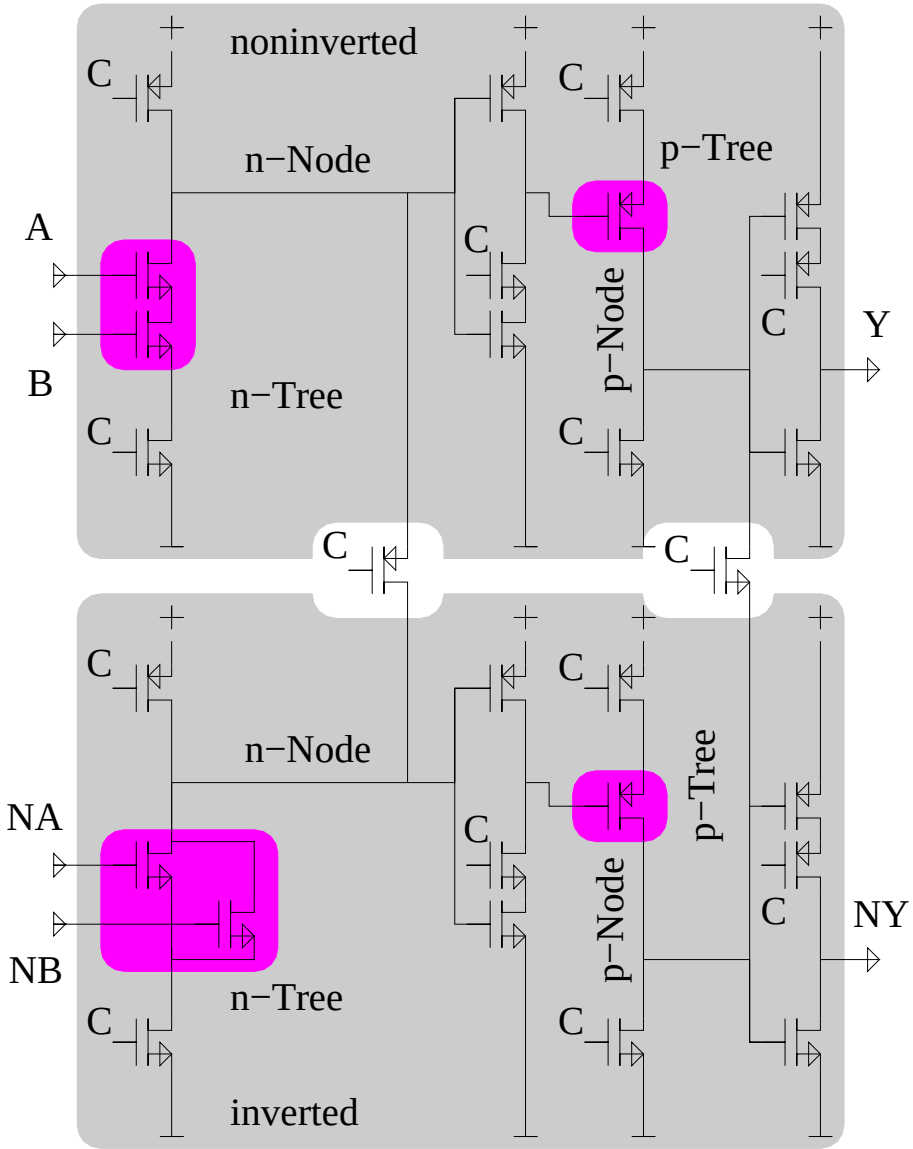
True Single Phase Clock - Logic



- advantages:
  - only one clock signal necessary
  - minor fan-in stress with only one transistor tree
  - utilization of both clock signal phases through alternating activation
- disadvantages:
  - increased load on the clocking signal net

- temporally separated operating conditions
  - pre-charge: storage of a small charge quantity on internal node
  - evaluate: logic-dependent discharge

Differential TSPC

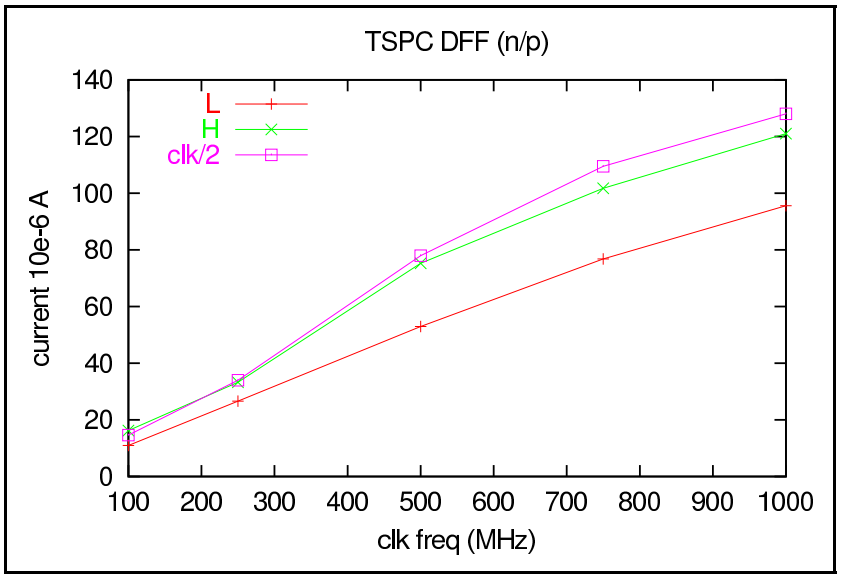


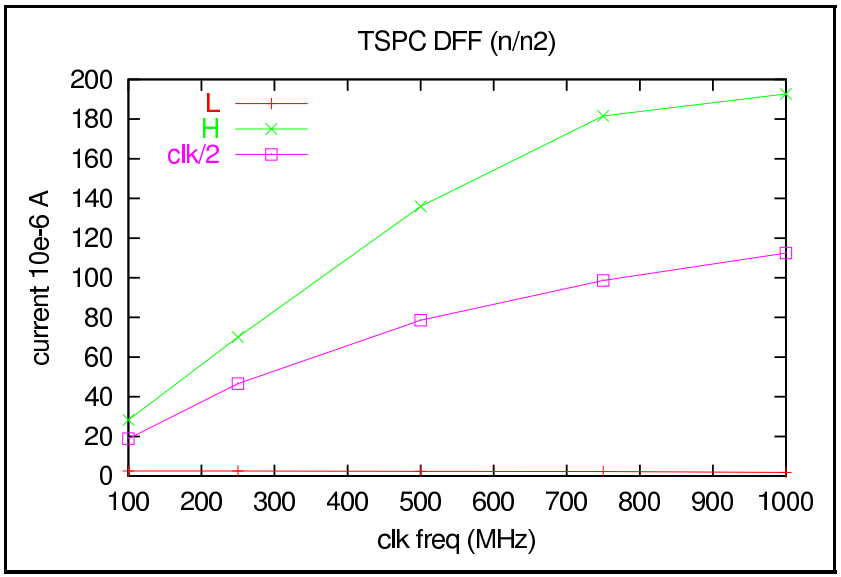
- complementary signal generation
- derived from the TSPC logic circuit
- easily cascade-able due to durable behavior
- advantages:
  - complementary logic function representable
  - abridgement of pipeline depth through parallel calculation
- disadvantages:
  - increased load of clock and signal lines
  - doubled implementation area and current consumption

Design requirements

- concerning latency uncritical and tolerant circuit environment
- particular suitable circuit architectures
  - arithmetic operators with bit-width-independent run time (CS-, SD-adder)
  - pipeline-able signal processing with non-recursive data flow
    - digital filter
    - spacial developed iterative algorithms (CORDIC, DES)
- unfavorable circuit architectures
  - runtime based elements (mono-flop, RS-flipflop)
  - architectures with strongly pronounced internal serial dependencies (Ripple Carry Adder (RCA))
  - feedbacks in data flow over more than one cell level (counter)

Minimization of the power consumption





- not the modification of an internal status consumes current, but the continuous revitalization of this status ⇒ DRAM
- power consumption depends on the distribution of the signal level frequency and not of the signal level change frequency
- possibilities for the reduction of power consumption:
  - by architecture selection
  - by changed optimization target
  - by adapted cell library

Minimization of the power consumption (2.)

influence of the selected architecture

- choosing of consumption-minimal state codings for incomplete code space allocation

  example 4 bit vector

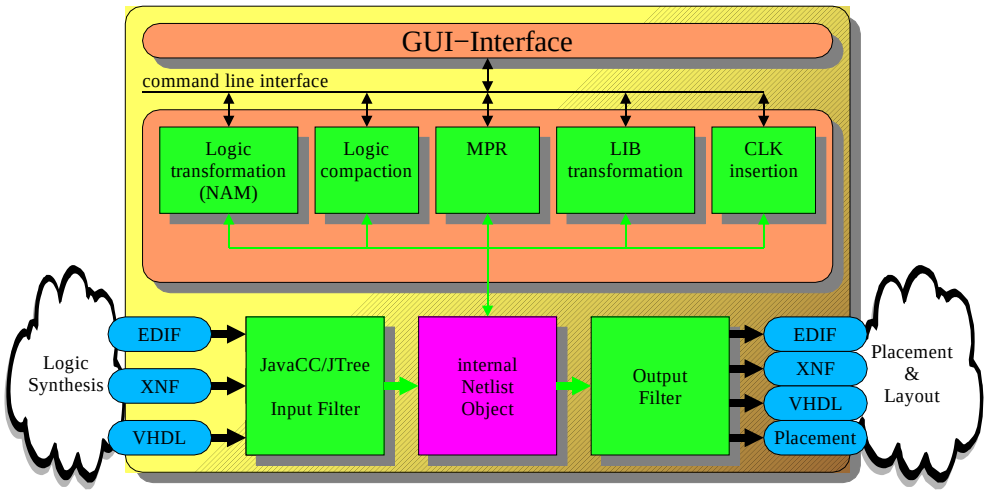
0	1	2	3	4
0000	0001 0010 0100 1000	0011 0101 0110 1001 1010 1100	0111 1011 1101 1110	1111

- one hot encoding for state machines
- ring counters

ring counter $\frac{1}{m}$ active $\Rightarrow 1$

binary counter 0.5 active $\Rightarrow 0.5n$

The DYNAMIC Toolset



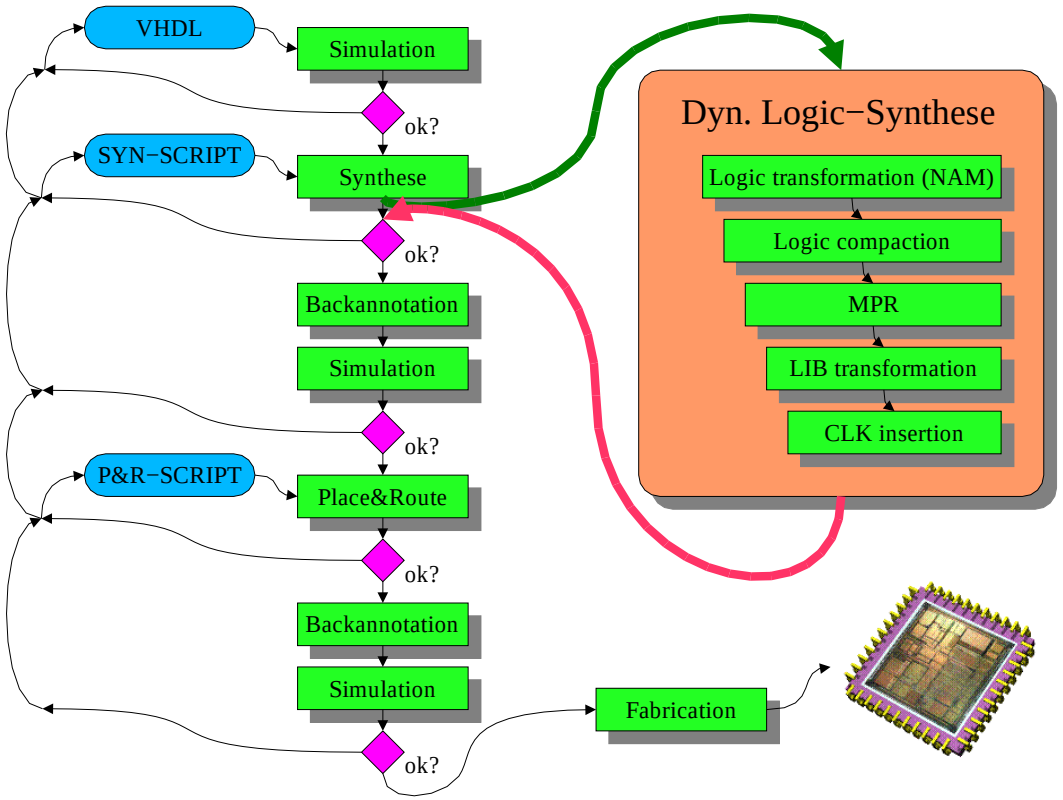
- written in JAVA
  - executable on any system with the necessary java run-time environment
  - web based execution possible → web based design offers
  - with jini/rmi reorganization of the design environment concept
- input filters written with JavaCC/JTree
  - easy creation of input filters for new formats
  - similar to lex/yacc for C
- threaded operation mode, “parallel” execution of sequential tasks
- modularized structure, one block for each task
- command line and gui interface

Performance analysis

Design name	Design-size Cells /Pipes/ CellsP	runtime in seconds Sparc 400MHz/2G Solaris C/Java/JIT	runtime in seconds Athlon 500MHz/256M Linux Java/JIT
add_rpl16	44/29/1115	673/52/19	29/17
add_cla16	216/10/479	90/17/20	9/8
add_clf16	353/11/799	231/35/21	21/15
add_bk16	140/11/444	84/15/19	8/6
add_rpl32	83/ /4029	nn/439/20	262/136
add_cla32	440/16/1591	1248/85/24	50/32
add_clf32	590/13/1429	750/75/31	42/30
add_bk32	336/13/1080	508/48/21	26/19
mult_cs8	288/23/1173	713/57/11	37/25
mult_cs12	628/32/2684	nn/191/26	137/91
mult_w8	574/23/1390	758/68/13	45/31
mult_w12	1141/30/2905	3494/196/28	140/96
des_slice	2046/27/6680	nn/896/132	3186/435
des_pipe	32920/241/119964	nn/22h13m	nn/nn
key_56	1680/9/4802	nn/551/89	416/364

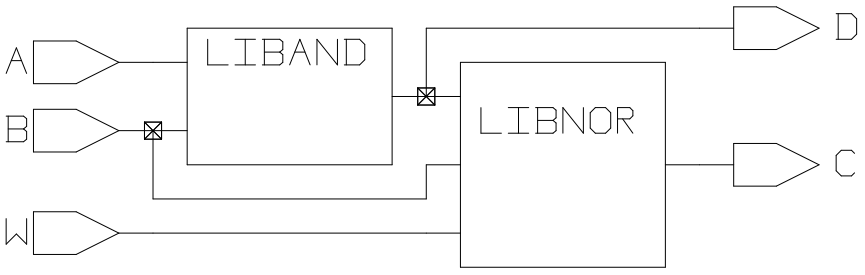
- speed up facts:
  - optimization of the utilized algorithms
  - quality of the runtime environment
- JIT technology brings sufficient run times
- speed scales well with clocking frequency over the different target systems

Design flow

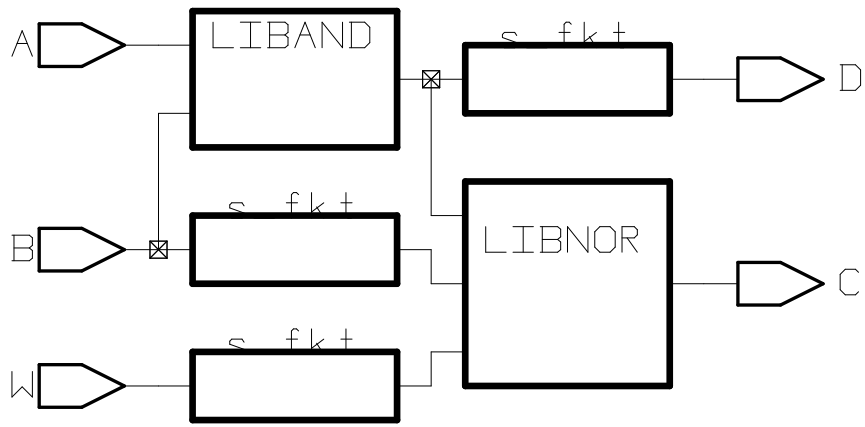


- TSPC design flow
  - integration of the dynamic circuit technology into a standard CMOS design flow
  - encapsulation of the particular dynamic characteristics into simulation and synthesis libraries
  - deployment of standard cell technique

Micro Pipeline Reorganizer (MPR)



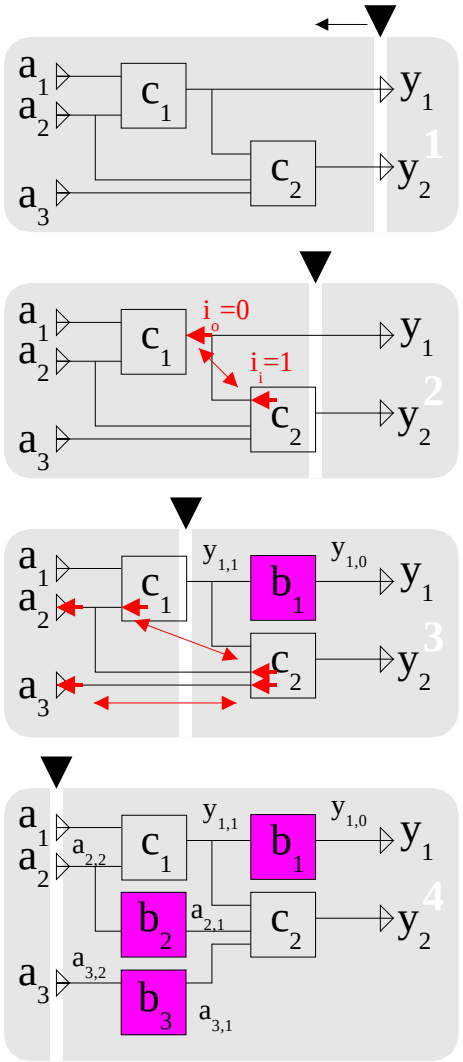
source netlist



processed netlist

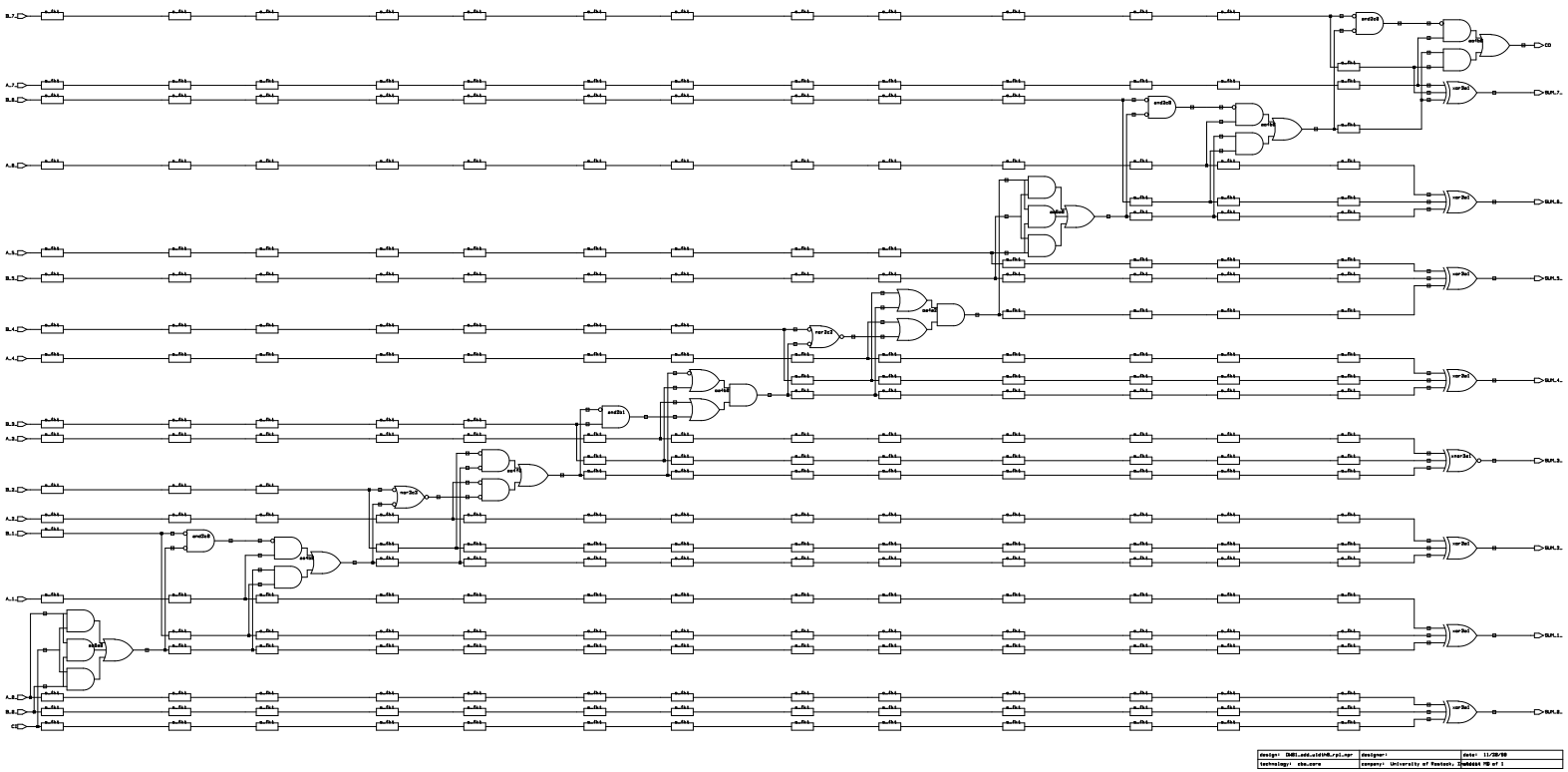
- target: Improvement of the pipeline throughput of a given circuit
- task: en-queuing of a pipeline structure by insertion of additional register cells
- requirements:
  - basic cells with only on output signal
  - no feedbacks in the netlist
- netlist formats: XNF, struct. VHDL, EDIF

MPR Algorithm R2L



- optimization begins at output ports of the network with the labeling of the net-level
- acquisition of the driving cell outputs
- verification of the net-level values of the associated cell inputs
- if necessary, insertion of buffer cells (re-gister)
- increment of the current net-level value
- repeat until all regarded networks are driven only by the input ports of the net-list

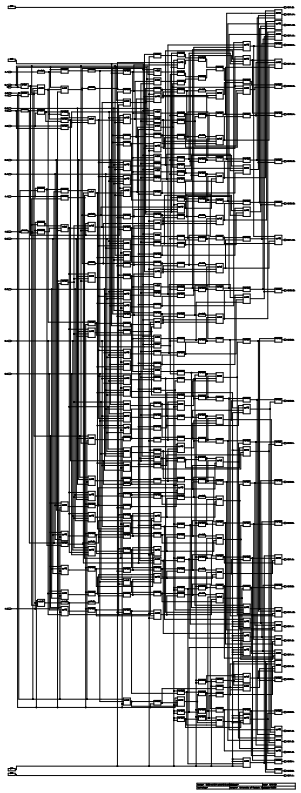
Applications

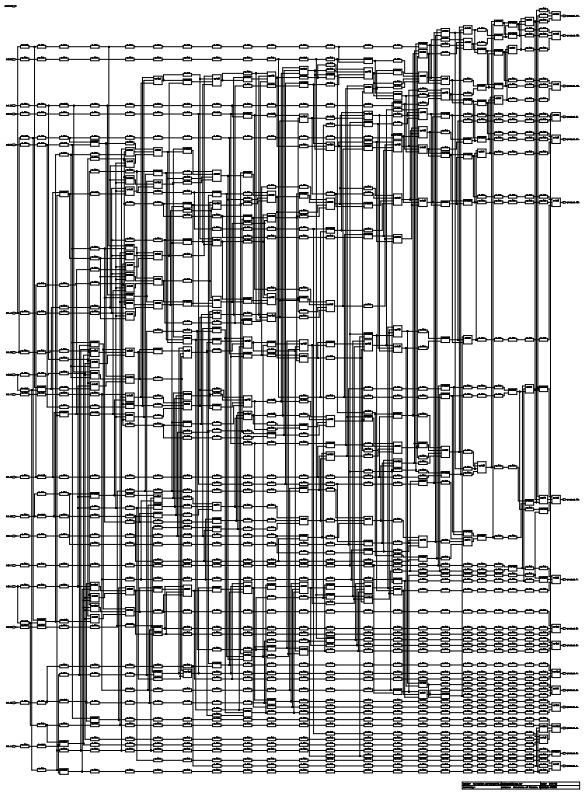


MPR 8 bit Ripple-Carry Adder

- pipeline expansion due to strong internal serial signal dependency $\Rightarrow$ factor $2n-2$
- has a pipeline effectiveness degree $E = \frac{b}{m}\frac{\sum_{i=1}^{m}\frac{N_i}{n_i}}{m} = \frac{8}{14}\frac{9.62}{14} = 0.392$

Design example: MPR 8x8 CSA Multiplier



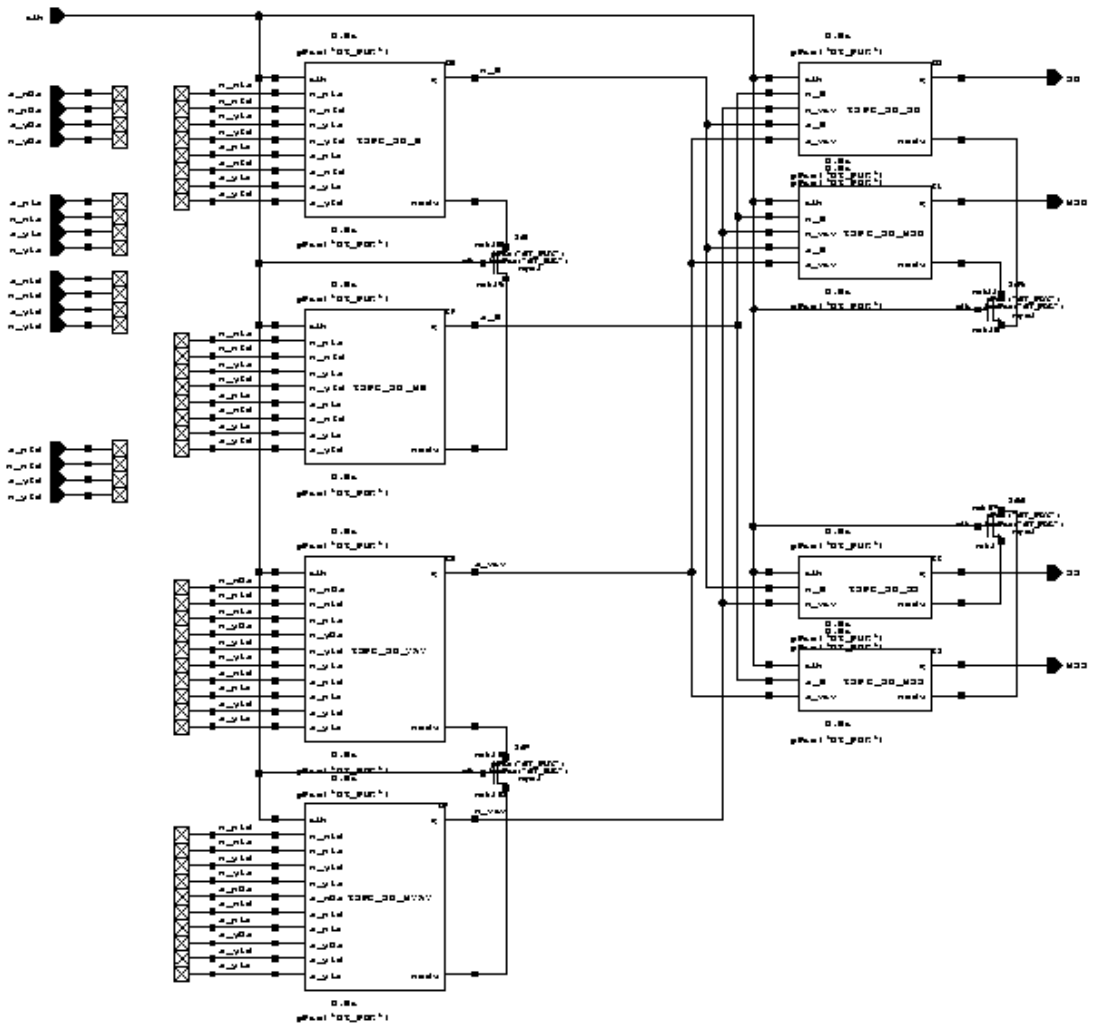


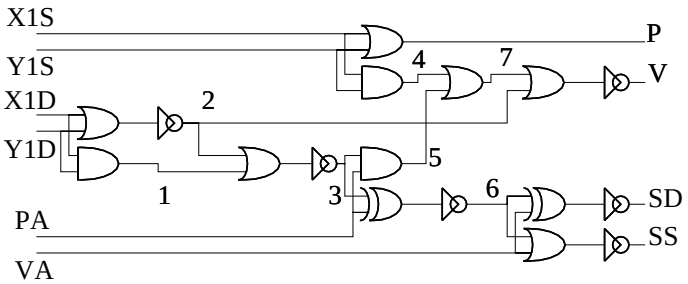
- 8bit x 8bit Carry-Save Multiplier from the Synopsys Design-ware
- increase of the pipeline levels
- pipeline effectiveness degree

$$E = \frac{b}{m}\frac{\sum_{i=1}^{m}\frac{N_i}{n_i}}{m}$$

$$= \frac{16}{23}\frac{20.98}{23} = 0.635$$

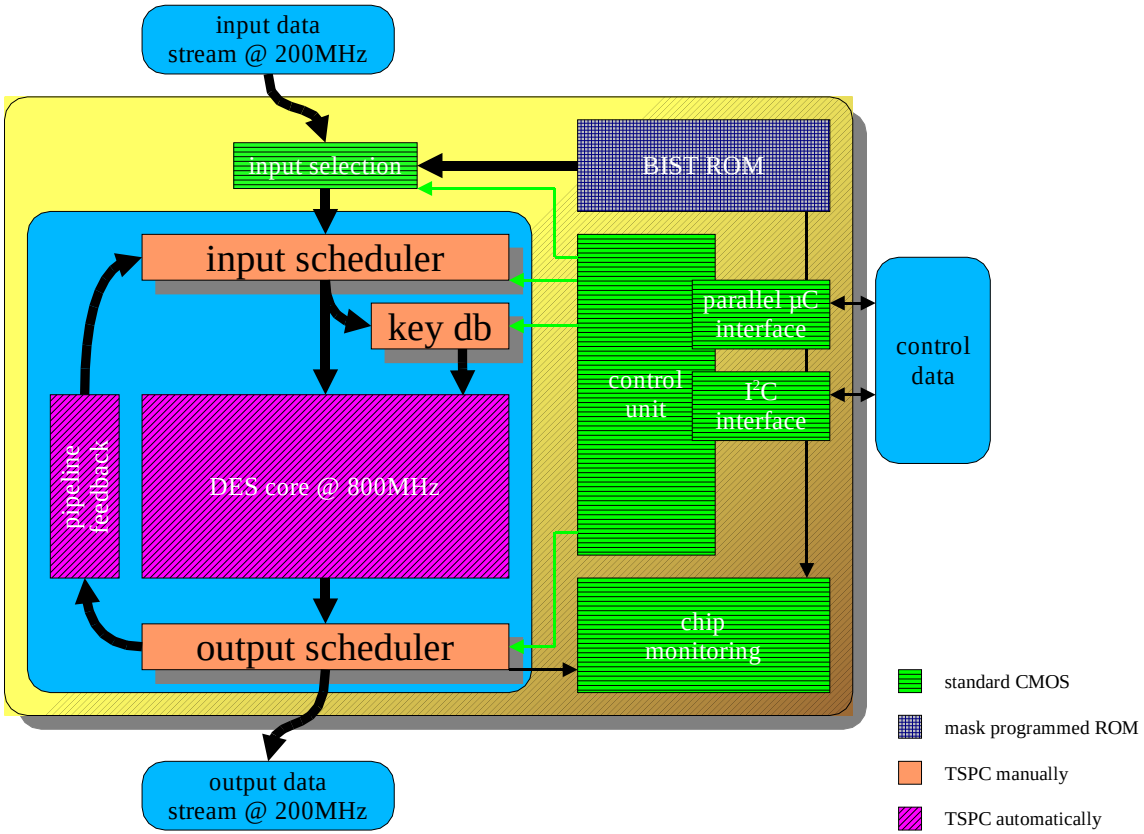
DTSPC example: Signed Digit Adder Cell





- 0.6μ $5V$ AMS-CUB Technology
- SD-Addition in one clock cycle in opposition to 8 clock cycles with single cell implementation

Design example: TDES



- streaming data (de/)encryption with Triple DES
- 0.6μ $5V$ AMS-CUB Technology @ 200/800MHz
- tree different design styles utilized
  - standard CMOS for control logic
  - circuits with manual arranged TSPC cells
  - automatically synthesized DES-pipeline constructed by TSPC cells

- core pipeline implements the 16 DES stages with 241 rows, approx. 30000 gates

Alternative application types

- acceleration of standard CMOS Designs
  - increase of the performance by automaticly generated pipeline structures, pipeline depth is determined by the netlist, no specification from the designer needed/possible as with the “balance-register”-approach
  - deployment for FPGA-development: one pipeline stage per clb
- implementation of wave pipelines
  - approach: exchange of the register functionality by simple delay elements (buffers)
  - for FPGA development: specification of the necessary routing delays simplified by finer granularity of the netlist

Conclusion

- application of dynamic circuit technology requires utilization of adapted circuit architectures
- integration in standard CMOS design flow possible by partial extension
- reference implementation of a TSPC standard cell library based on AMS $0.35\mu m$ or $0.6\mu m$ Technologies
- implementation of reference designs (CORDIC, Digit-Online-Neuro, DES)
- evaluation of alternative application type for the toolset
- development of a standard cell generator for TSPC