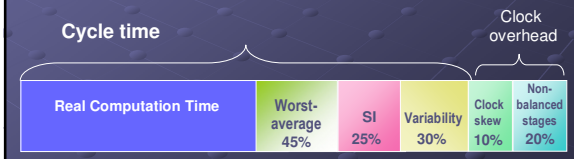
Dual-Phase, Dual-Rail Monotonic Circuits

HY590-20

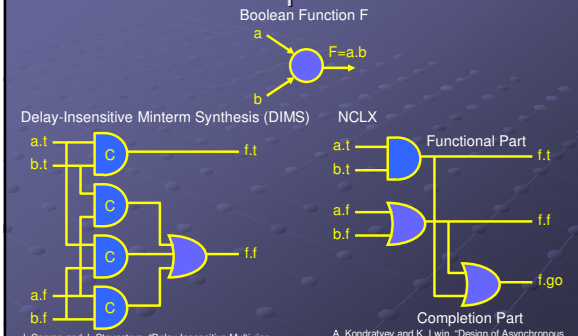
3

Motivation

- EDA technology delay estimation is becoming unrealistic.
  - STA is unrealistic
    - Wc. Vs. typ.
    - IR drop and crosstalk not taken into account.
    - Variability
  - Clocking overhead
    - Skew
    - Pipeline stage balancing



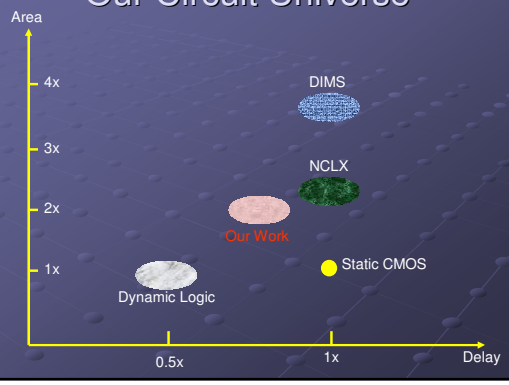
Dual-Rail Implementations



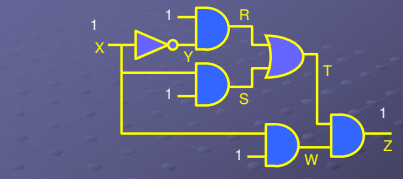
J. Sparsø and J. Staunstrup, "Delay-Insensitive Multi-ring Structures", Integration, the VLSI Journal, vol. 15, no 3. pp. 313-340, Oct. 1993

A. Kondratyev and K. Lwin, "Design of Asynchronous Circuits by synchronous CAD tools", Proceedings of IEEE DAC, June 2002.

Our Circuit Universe



Hazards in Combinational Logic



- single input change at X.
- non-monotonic transitions.
- static-1 hazard at T.
- 1-0 hazard at Z.

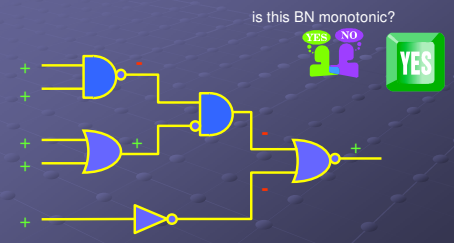
Monotonicity

- Monotonic Input Transition (MIT):
  - Subset of inputs change 0→1 or 1 →0.
- Monotonic Operation Mode (MOM):
  - Divide circuit operation into two MIT phases:
    - (a) subset of inputs changes.
    - (b) subset of outputs changes.
- Dual Rail:
  - (a) and (b) sub-phases where ½ signals rise/fall.

MBNs (Monotonic Boolean Networks)

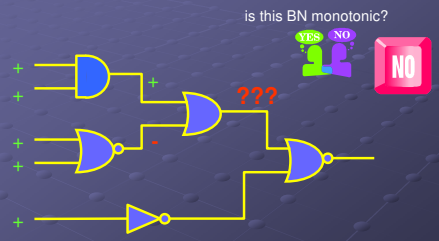
- Unate Functions:
  - f is *increasing (decreasing)* in variable x_i:
    - x_i: 0→1, f cannot change 1 →0 (0 →1).
    - f unate in x_i, iff f is increasing or decreasing in x_i.
- Node n_i with local function f_i is *+ve:*
  - x_i is *+ve* and f_i is *increasing* in x_iOR
  - x_i is *-ve* and f_i is *decreasing* in x_i.
- Node n_i with local function f_i is *-ve:*
  - x_i is *-ve* and f_i is *increasing* in x_iOR
  - x_i is *+ve* and f_i is *decreasing* in x_i.
- A node *is monotonic* if it is *either +ve or –ve.*
- A BN is MBN if all its nodes are monotonic.
- MBN is *hazard-free* under *monotonic input transitions.*

Monotonic Boolean Networks



- Assign polarity to every node to check monotonicity.

Monotonic Boolean Networks



MBN Transformations

- Key Questions:
  - given a BN, how do we transform it to an MBN?
  - given an MBN, which transformations and optimizations can we apply to reduce to another MBN?

MBN Transformations

- Answers:
  - Provide two transformation procedures, *based on the dual-rail code*, for generating an MBN from a generic BN:
    - Technology-Independent (TI) Conversion
    - Technology-Mapped (TM) Conversion
  - Provide a set of hazard-non-increasing transformations on MBNs.

TI Conversion

- Steps
  - For each p.i. *x*, create x^t and x^f representing the true and false evaluations of x.
  - For each node inplementing $y_i = f_i(x_1, \ldots, x_n)$, create two nodes:
    - $y^t_i = DR(f_i(x_1, \ldots, x_n))$ and $y^f_i = DR(\text{'}f_i(x_1, \ldots, x_n))$
    - $y = \text{'}x$, special case, $y^t = x^f$, $y^f = x^t$
- Define DR recursively (Shannon's exp.):
  - DR(0) = 0, DR(1) = 1
  - $DR(x.f_x + \text{'}x.f_{\text{'}x}) = x^t.DR(f_x) + x^f.DR(f_{\text{'}x})$

TI Conversion

- Conversion Theorem:
  - Given:
    - function y = f(x_1, …,x_n)
    - Assumption that $x^i = x^t_i = {'x}^f_i$
  - It holds that:
    - $y = y^t = {'y}^f$
- Proof: by induction on x_i
  - i = 0, DR(0) = 0, i = 1, DR(1) = 0
  - Split y^t, y^f, for i, using S.E.,
  - by induction $DR(f_x) = f_x$, $DR(f'_x) = f_{'x}$.

Shannon's Expansion

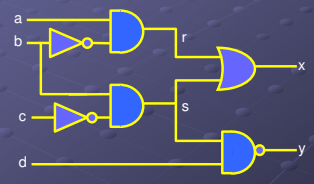
- Boolean Relation, *f:* $B^m \rightarrow B^n$:
- Cofactors of *f:*
  - *for* $f(x_1, x_2, \ldots, x_i, \ldots, x_n)$
    - co-factor w.r.t. x_i : $f(x_1, x_2, \ldots, 1, \ldots, x_n)$
    - co-factor w.r.t. $'x_i$: $f(x_1, x_2, \ldots, 0, \ldots, x_n)$
- Shannon's Expansion:
  - Let $f : B^n \rightarrow B$:
  - => $f(x_1, x_2, \ldots, x_i, \ldots, x_n) =$
    - $(x_i \cdot f_{xi}) + ('x_i \cdot f_{'xi})$ [sum form]
    - $(x_i + f_{'xi}) \cdot ('x_i + f_{xi})$ [product form]

TI Example

- $y = a\ 'b + b(c + 'd)$
  - would be converted into:
  - $y^t = DR(a\ 'b + b(c + 'd)) = a^t b^f + b^t(c^t + d^f)$
  - $y^f = DR('(a\ 'b + b(c + 'd)) = (a^f + b^t)(b^f + c^f d^t)$
- How do I convert?
  - Use S.E.
  - assume that $x^i = x^t_i = {'x}^f_i$, for all x and i.
  - Use the SIS **dr package**!

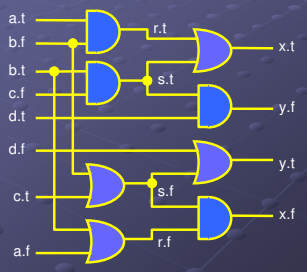
TI Conversion - #1

- Original BN:



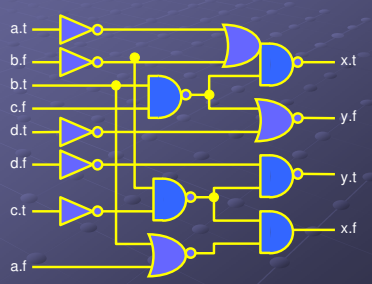
TI Conversion - #2

- Dual-Rail Conversion:



TI Conversion - #3

- Technology Mapping to a library:

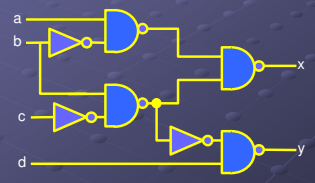


TM Conversion

- Steps
  - For each gate, producing signal y^t_j, from signals y^t_j, …, y^t_k, add a *dual* gate, based on De Morgan's law.
  - Label each node +ve or –ve, starting from p.o.'s, according to gate polarities. In case of multiple paths, select longest.
  - For each gate input or p.i., which is inconsistently labelled, invert and connect to its dual.

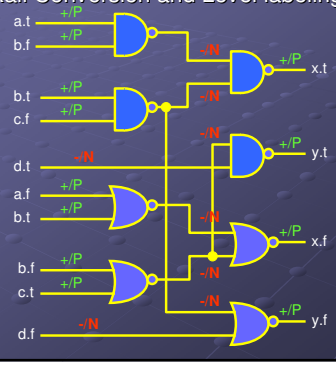
TM Conversion - #1

- Original technology-mapped circuit:



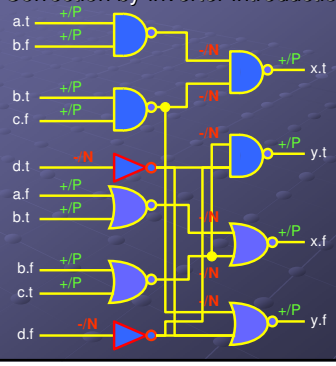
TM Conversion - #2

- Dual-Rail Conversion and Level labeling:



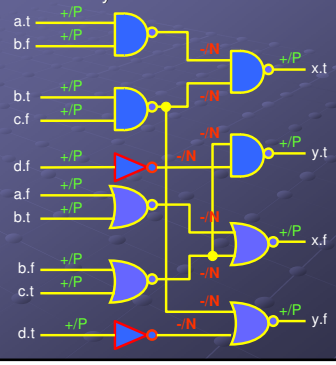
TM Conversion - #2

- Phase Correction by inverter introduction:



TM Conversion - #3

- Phase Correction by inverter introduction:



MBN Conversion Approaches

- Both TI and TM approaches equivalent !
  - TI: early synthesis stages.
    - Based on recursive Shannon Expansion.
  - TM: mapped and analyzed circuits.
    - Based on De Morgan's Law and phase correction.

Hazard-non-Increasing Transformations

- Set of transformations that preserve MBN:
  - De Morgan's laws.
  - Dual global and global flow.
  - Tree decomposition.
  - Gate replication.
  - Collapsing.
  - Kernel-factoring.
  - Cube-factoring.

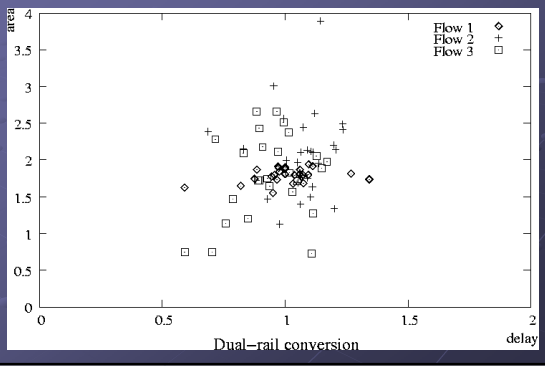
Fast Reset and CD

- MBN circuit operates at ½ speed:
  - WHY?
  - all p.i.'s *must be reset* before the next monotonic phase begin.
    - in D.R. this is the 00 value for every pair.
- **Fast Reset:**
  - Speed-up reset phase
  - Force more gate outputs to their "quiescent" value.
  - Tradeoff: number of gates reset v.s. capacitance of reset signal.

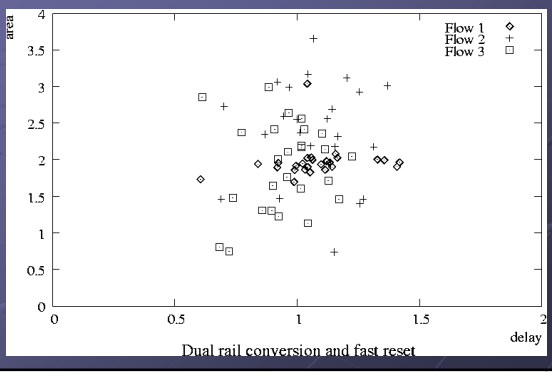
MBN Flows

- **Flow 1**
  - TM-conversion using AWK scripts.
  - Fanout-fixing in Synopsys DC (max_delay).
- **Flow 2**
  - TI-conversion using SIS dr package.
  - Optimization in Synopsys DC (hazards?).
- **Flow 3**
  - TM-conversion using AWK scripts.
  - Optimization in Synopsys DC (hazards?).
- Run them on SIS Benchmarks + RTL Examples (DLX and DES).

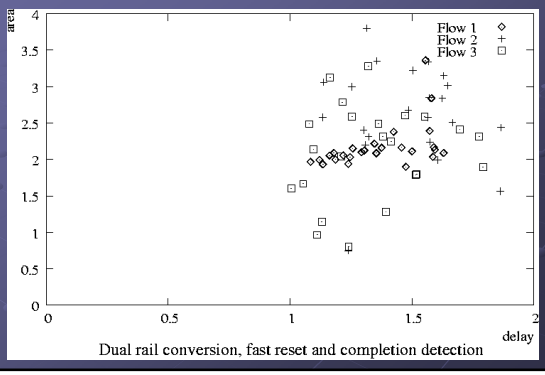
Results – Scatter #1



Results – Scatter #2



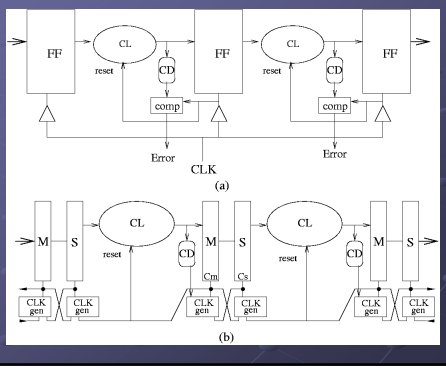
Results – Scatter #3



Use Cases

- C.L. with CD can be used in two environments:
  - Synchronous or Asynchronous.
  - Delay Fault Testing for Free (for MBN).
- Synchronous environment
  - use CD to check completion w.r.t. clock.
  - produce *synchronization error* signal if CD signal transitions after the clock edge.
  - Error signal:
    - Bin chips according to performance
    - provide on-line testing capability
- Asynchronous environment
  - performance

Use Cases



Conclusions

- Using MBNs we can design ***FAST***, ***static CMOS***, Monotonic circuits!
- Fast DR + De-synchronization:
  - COMPLETE design flow for asynchronous circuits!
  - C.L. handled by Fast DR (partially).
  - F.F./Lat. handled by De-synchronization.
- Variability:
  - Asynchronous OR synchronous with error detection (and correction?).
  - Delay Fault Testing!