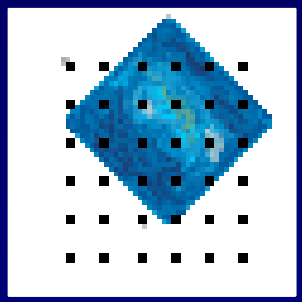
Taylor Expansion Diagrams: A Compact Canonical Representation for Symbolic Verification



M. Ciesielski, P. Kalla, Z. Zeng

Electrical & Computer Engineering
University of Massachusetts, Amherst, USA
ciesiel@ecs.umass.edu

B. Rouzeyre

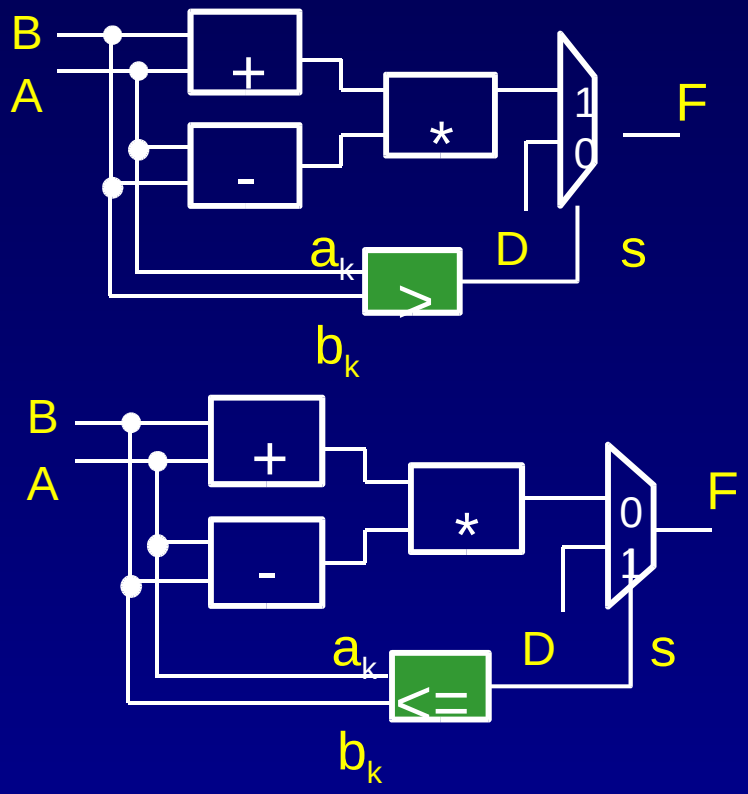
LIRMM
Montpellier, France
rouzeyre@lirmm.fr

Outline

- Motivation – RTL verification
- Background
  - BDD (binary), BMD (word-level)
- New canonical representation: TED
  - Construction and manipulation
  - Properties
  - Applications
- Conclusions
  - Limitations, future work

Motivation – RTL Verification

- Complex RTL designs
  - Data flow and control
  - Arithmetic and Boolean
- Equivalence verification
  - Need representation that can handle arithmetic/Boolean
    - Efficient, compact
    - Canonical



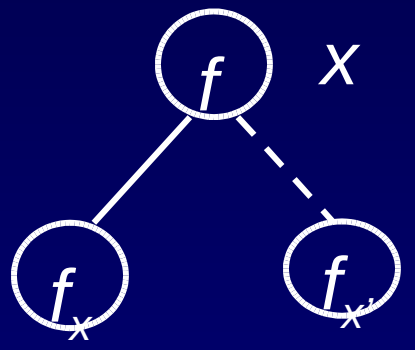
Common Representations

- Boolean functions ($f : B \rightarrow B$)
  - Truth table, Karnaugh map
  - SoP, PoS, ESoP, Reed-Muller, etc.
  - Binary Decision diagrams (BDD, ZDD, KFDD, …)
- Arithmetic functions ($f : B \rightarrow Int$)
  - Binary Moment Diagrams (*BMD, K*BMD, …)
  - Multi-terminal, Algebraic Decision Diagrams (ADD)
- Need more abstract representation for arithmetic functions ($f : Int \rightarrow Int$)

Binary Decision Diagrams (BDD)

- Based on recursive Shannon expansion: $f = x f_x + x' f_{x'}$

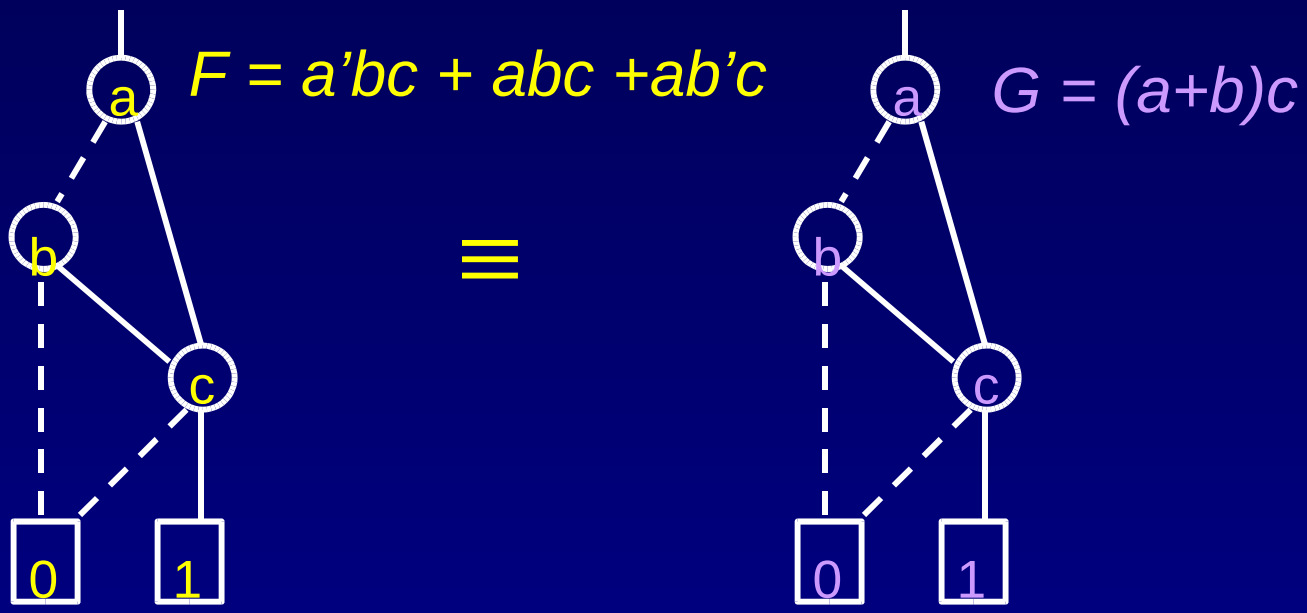
  where: $f_x = f(x=1)$, $f_{x'}=f(x=0)$



- Compact data structure for Boolean logic
- Canonical representation
  - reduced *ordered* BDDs (ROBDD)
- Essential for verification
  - equivalence checking, satisfiability (SAT)

Application to Verification

- Canonicity: equivalence checking of logic circuits



- Limitations
  - Require bit-level expansion of word-level variables
  - Size explosion for some functions (arithmetic)

Binary Moment Diagrams (BMD)

- Devised for word-level, arithmetic operations
- Based on modified Shannon expansion (*pos. Davio*)

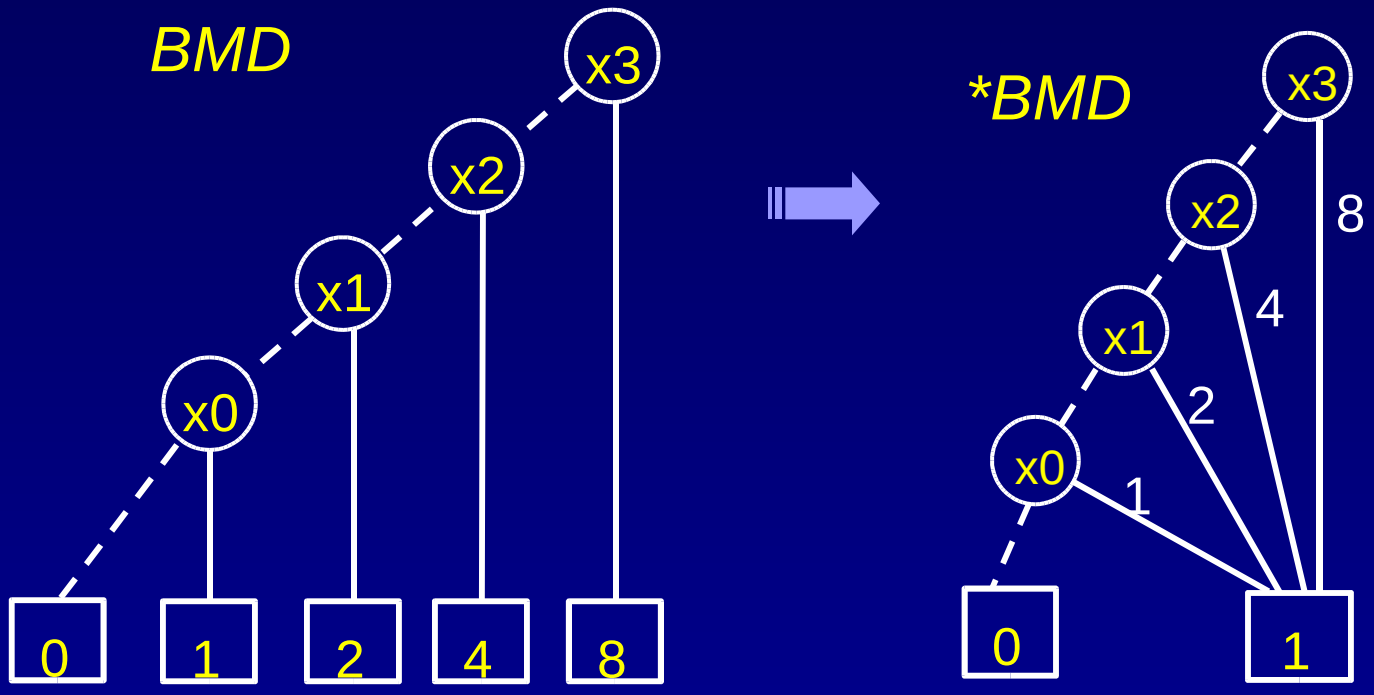
$$f = x f_x + x' f_{x'} = x f_x + (1-x) f_{x'}$$
$$= f_{x'} + x (f_x - f_{x'}) = f_{x'} + x f_{\Delta x}$$

where $f_{x'} = f_{x=0}$ is *zero moment*

$f_{\Delta x} = (f_x - f_{x'})$ is *first moment (*derivative)

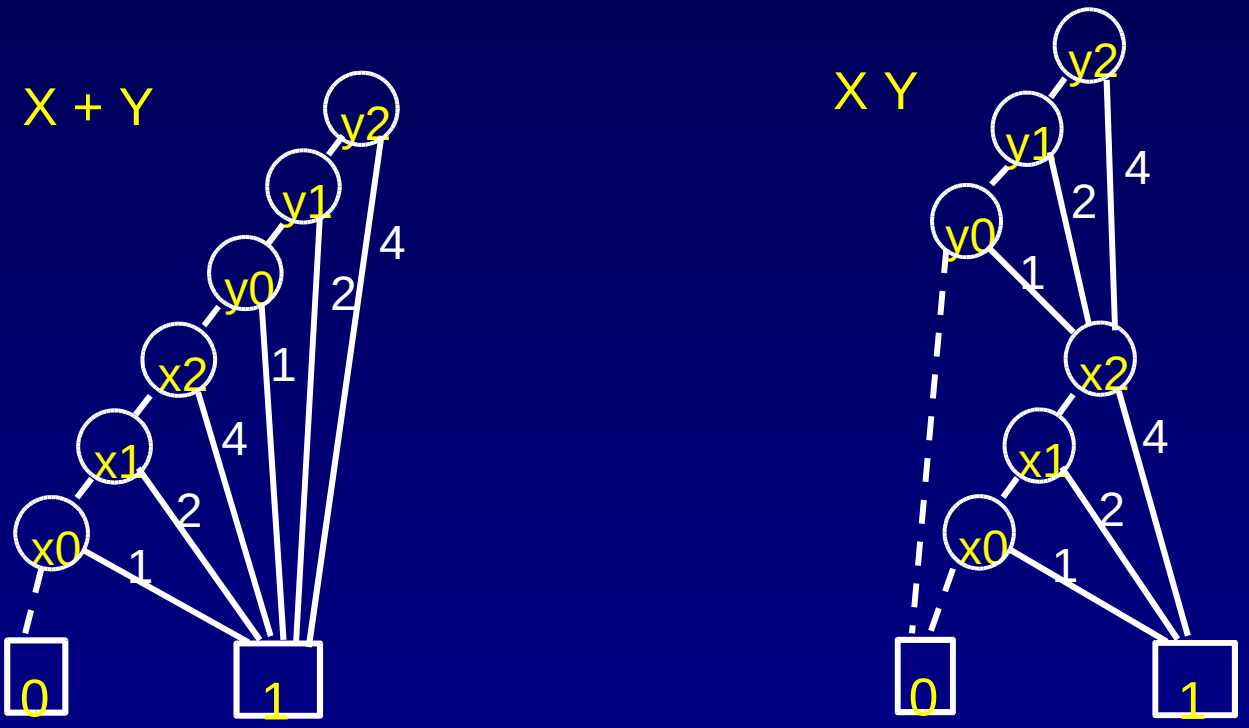
BMD – Example

- Unsigned integer: $X = 8x_3 + 4x_2 + 2x_1 + x_0$



*BMD

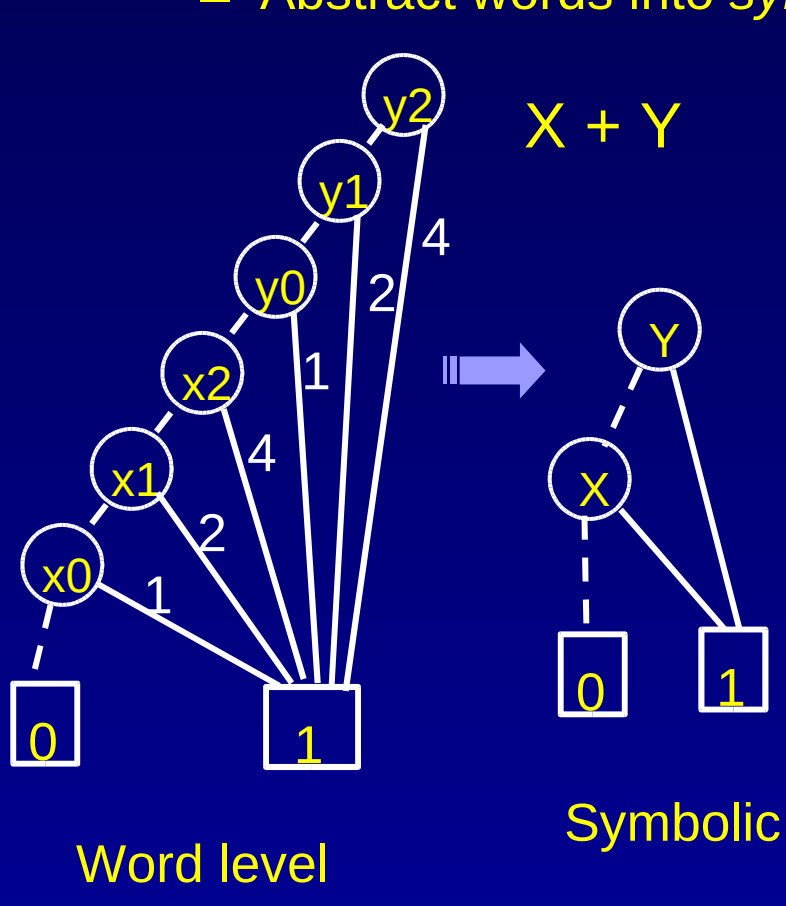
- Efficiently models word-level operators

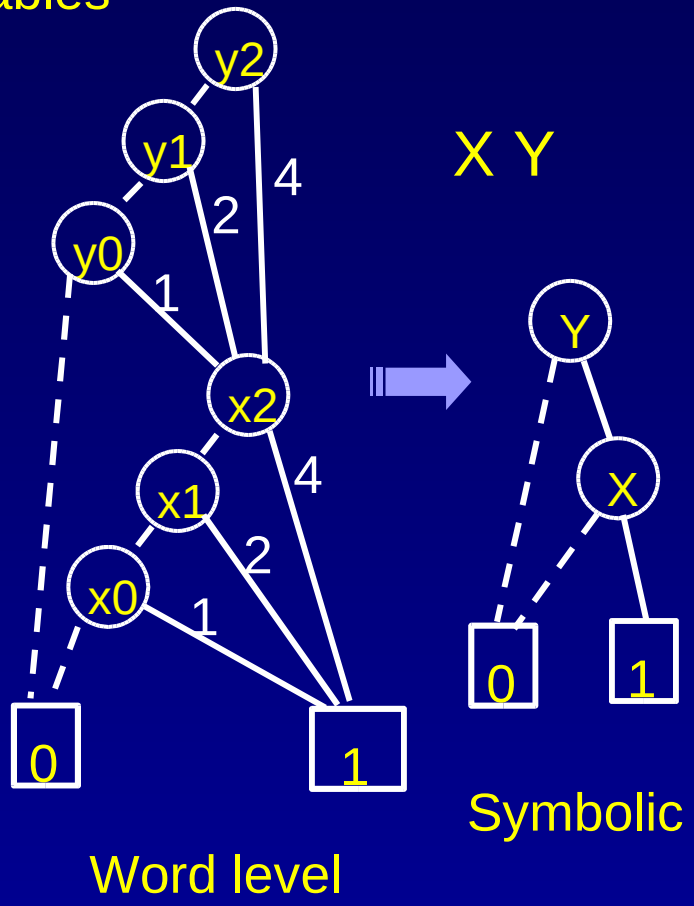


- Limitation: requires bit-level representation of a word

Symbolic Representation

- Why expand words into bits? Can we do better?
  - Abstract words into *symbolic* variables



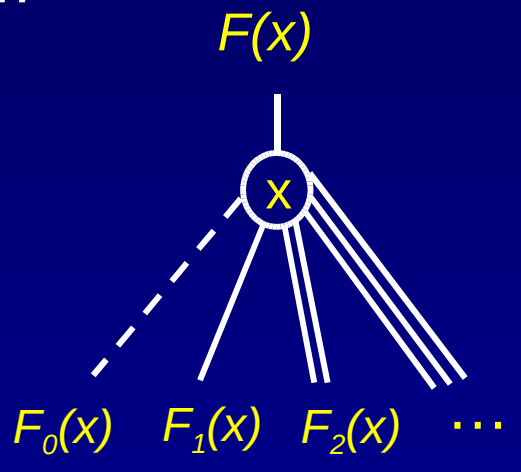


Taylor Expansion Diagram (TED)

- *F = arithmetic* function (*F : Int → Int*)
  - Treat *F* as a continuous function
  - *Taylor Expansion* (around *x=0*)*:*

$$F(x) = F(0) + x\,F'(0) + \tfrac{1}{2}\,x^2\,F''(0) + \ldots$$

- Notation
  - $F_0(x) = F(x=0)$ — *0-child* - - - - - -
  - $F_1(x) = F'(x=0)$ — *1-child* ----------
  - $F_2(x) = \tfrac{1}{2}\,F''(x=0)$ — *2-child* ======
  - *etc.*



$$F(x) = F_0(x) + x\,F_1(x) + x^2\,F_2(x) + \ldots$$

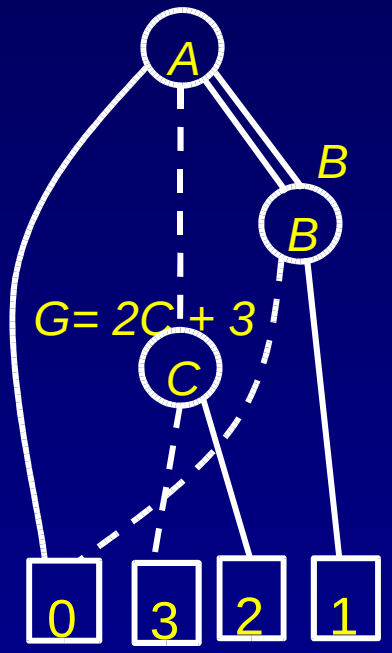
Construction - Your First TED

$F = A^2B + 2C + 3$

A
- $F_0(A) = F_{|A=0} = 2C + 3$
- $F_1(A) = F'_{|A=0} = 2AB_{|A=0} = 0$
- $F_2(A) = \frac{1}{2}\, F''_{|A=0} = B$

B
- $B_0 = B(0) = 0$
- $B_1 = B' = 1$

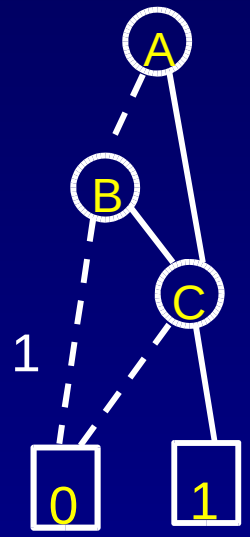
C
- $G_0(C) = (2C+3)_{|C=0} = 3$
- $G_1(C) = (2C+3)' = 2$



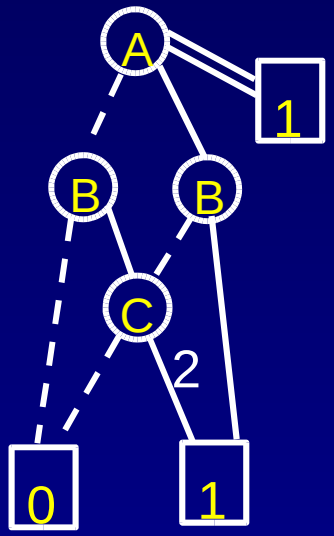
(without normalization)

TED – a few Examples

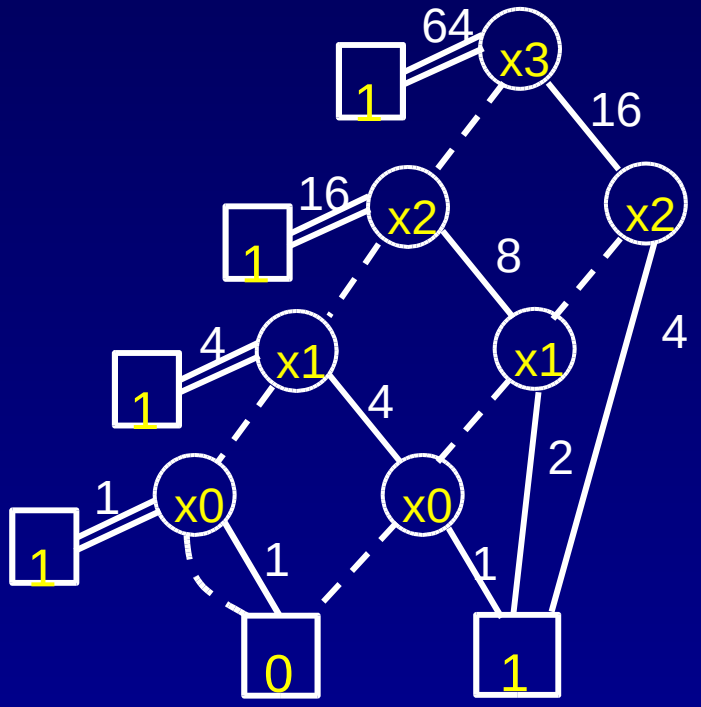
(A+B)C +1



(A+B)(A+2C)



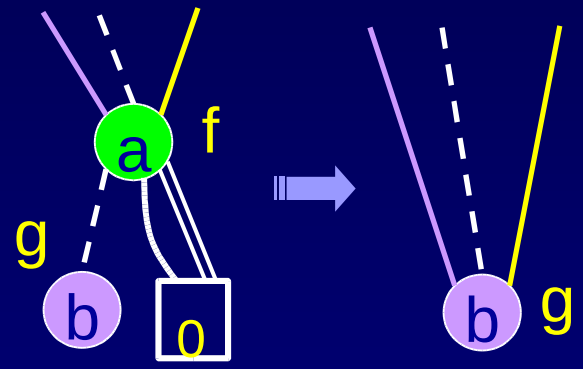
$X^2 = (8x_3 + 4x_2 + 2x_1 + x_0)^2$



TED - Reduction Rules

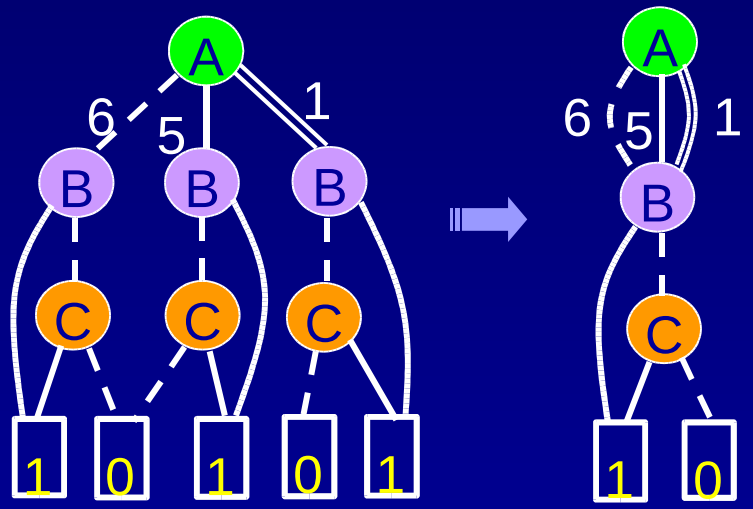
1. Eliminate *redundant* nodes:
   - Nodes with *all* edges $\rightarrow$ 0
   - Nodes with only constant term

$$f = 0\,a^2 + 0\,a + g(b) = g(b)$$



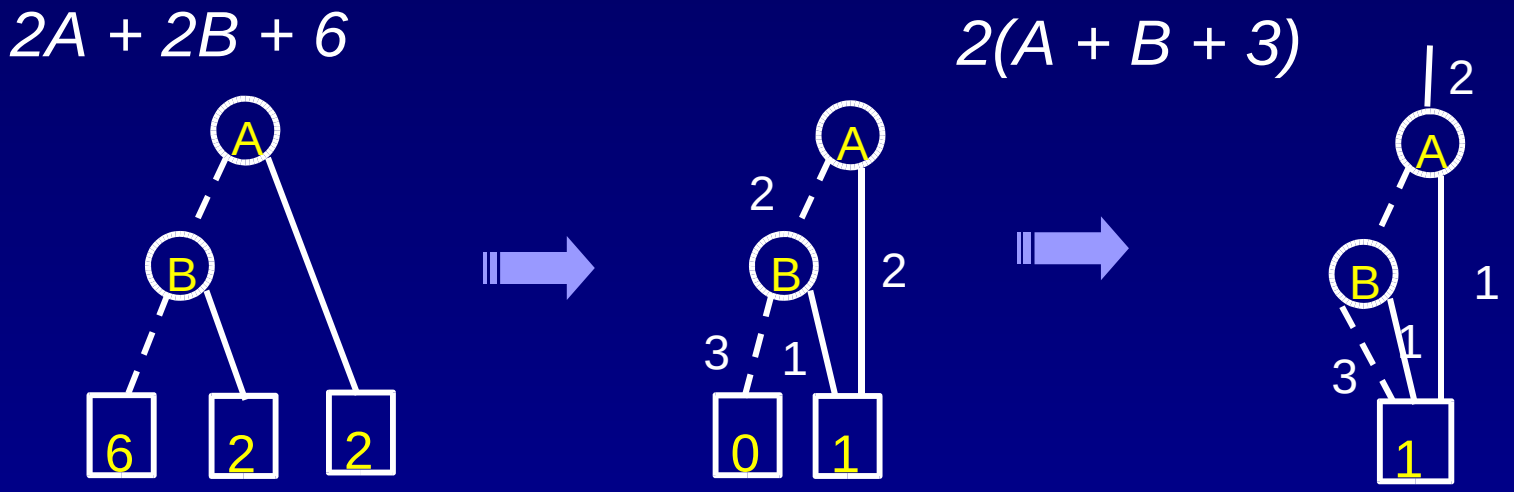
2. Merge *isomorphic* subgraphs

$$(A^2 + 5A + 6)(B + C)$$



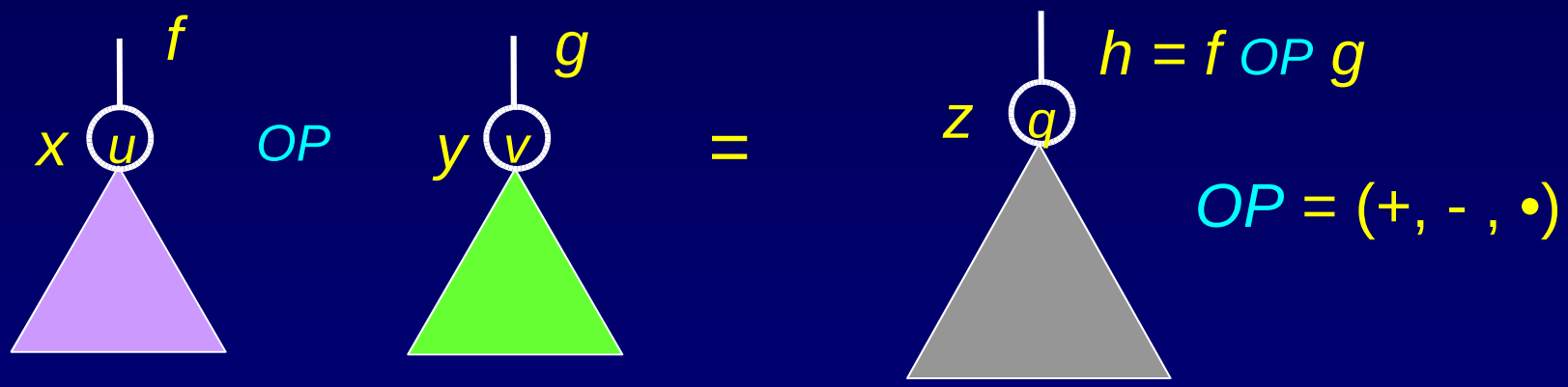
TED Normalization

- TED can be *normalized*
  - weights of edges of a given node must be *relatively prime* (to allow sharing isomorphic graphs)



TED: Composition

- Recursive *composition* of nodes, starting at the top



- Operation depends on relative order of variables *x*, *y*
  - if $x = y$, then $z = x$, and

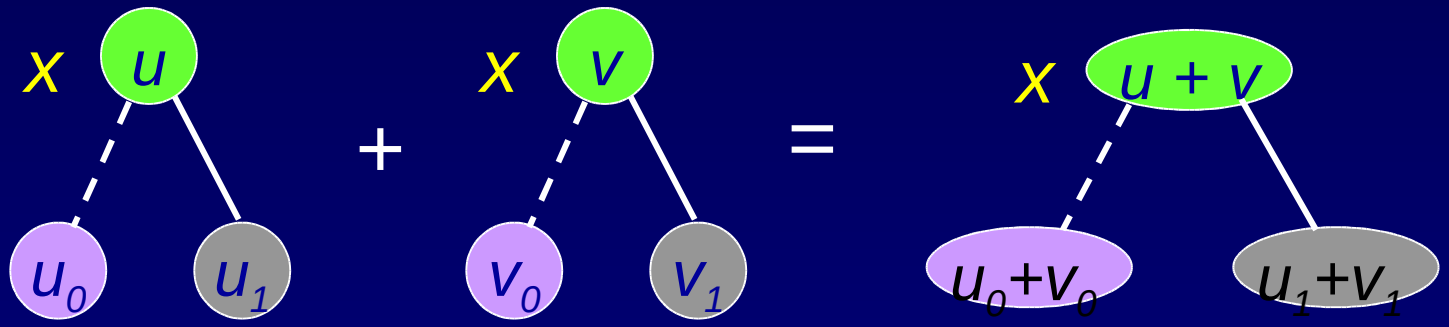
    $h(x) = f(x)\ OP\ g(x)$

    $= f_0(x)\ OP\ g_0(y) + x\,[f_1(x)\ OP\ g_1(y)] + x^2\,[f_2(x)\ OP\ g_2], \ldots$
  - if $x > y$, then $z = x$, and

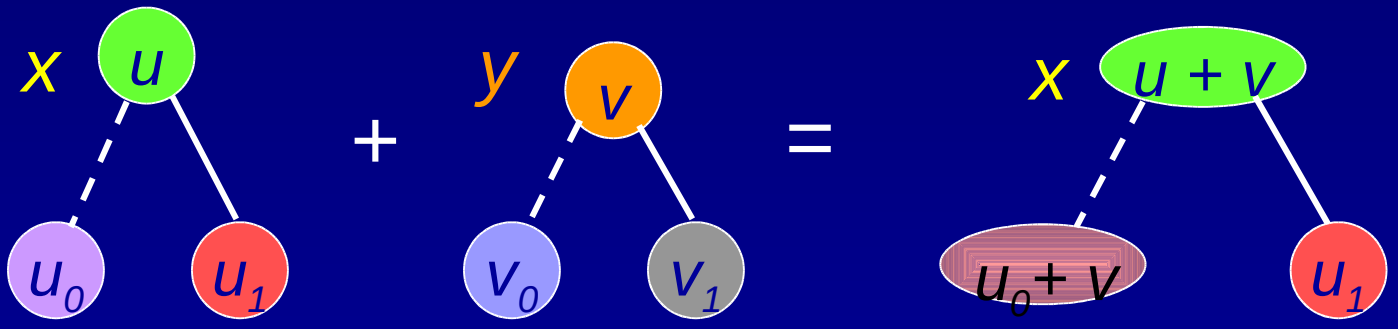
    $h(x) = f_0(x)\ OP\ g(y) + x\,[f_1(x)\ OP\ g(y)] + x^2\,[f_2(x)\ OP\ g], \ldots$
  - else ….

COMPOSE Operator – ADD/SUB

- Nodes indexed by *same* variable

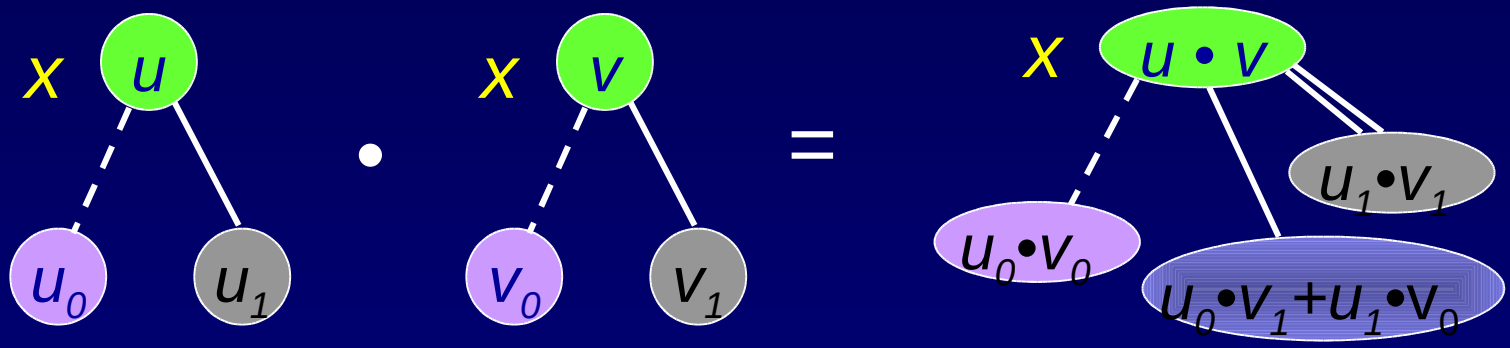


- Nodes indexed by *different* variable ($x > y$)

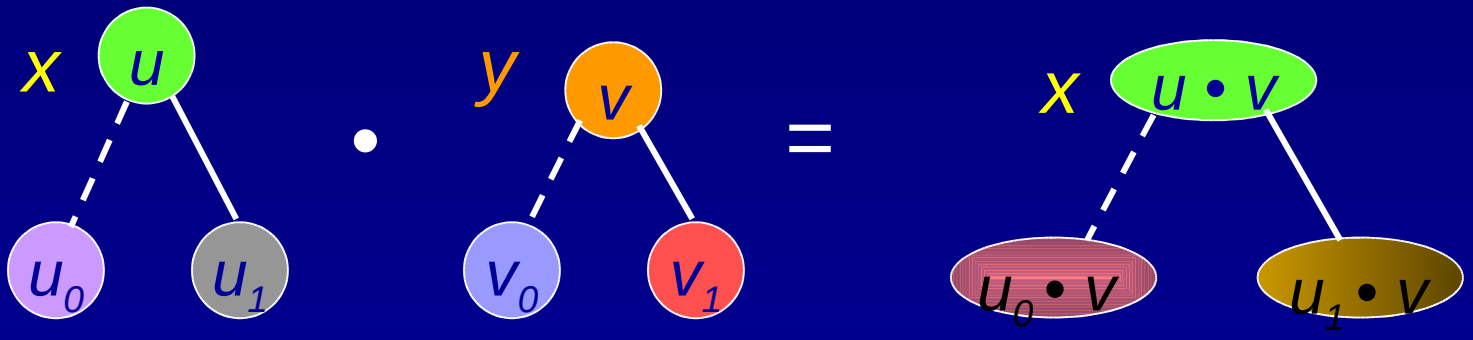


COMPOSE Operator – MULT

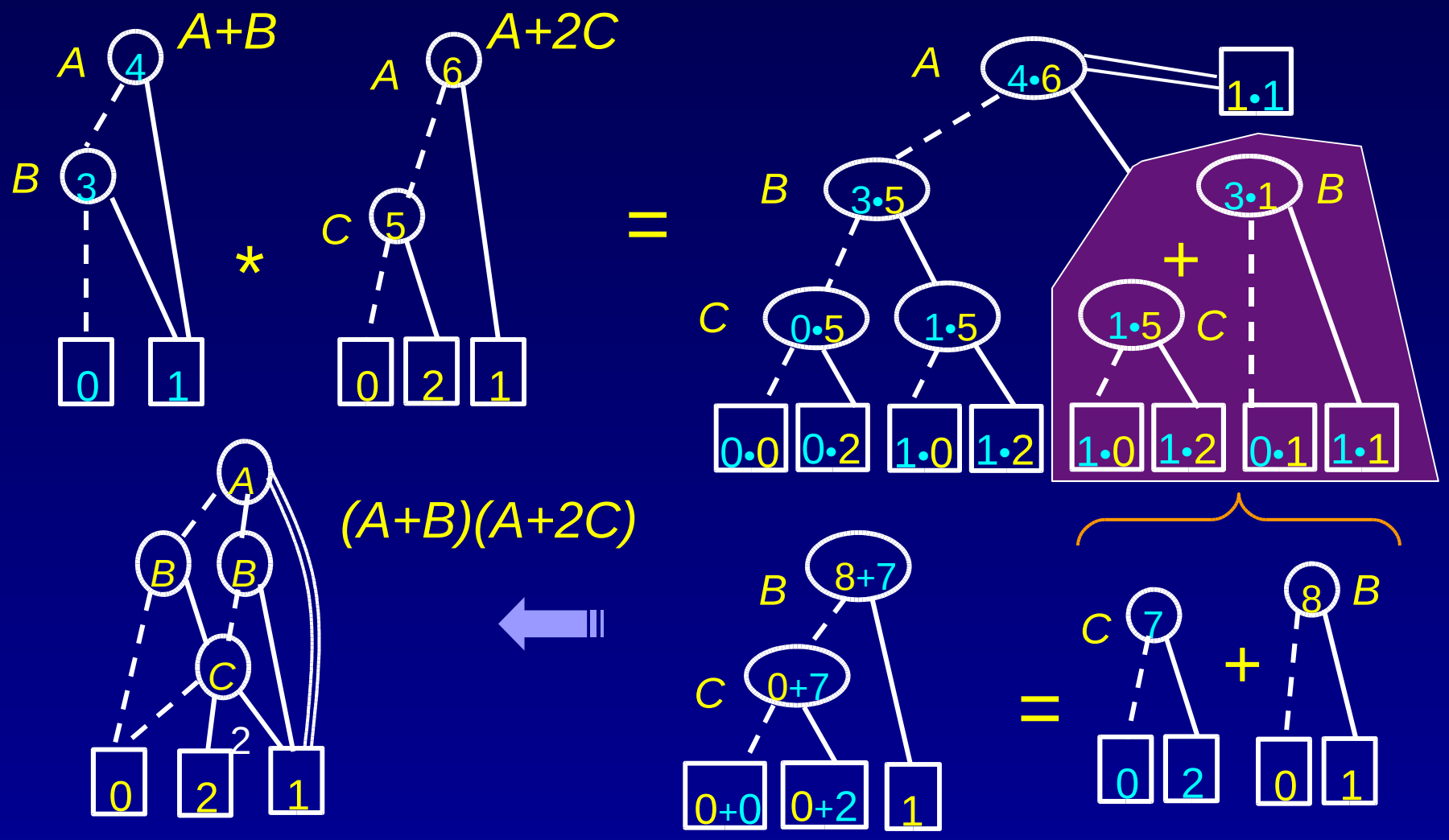
- Nodes indexed by *same* variable



- Nodes indexed by *different* variable ($x > y$)



COMPOSE Operation - Example

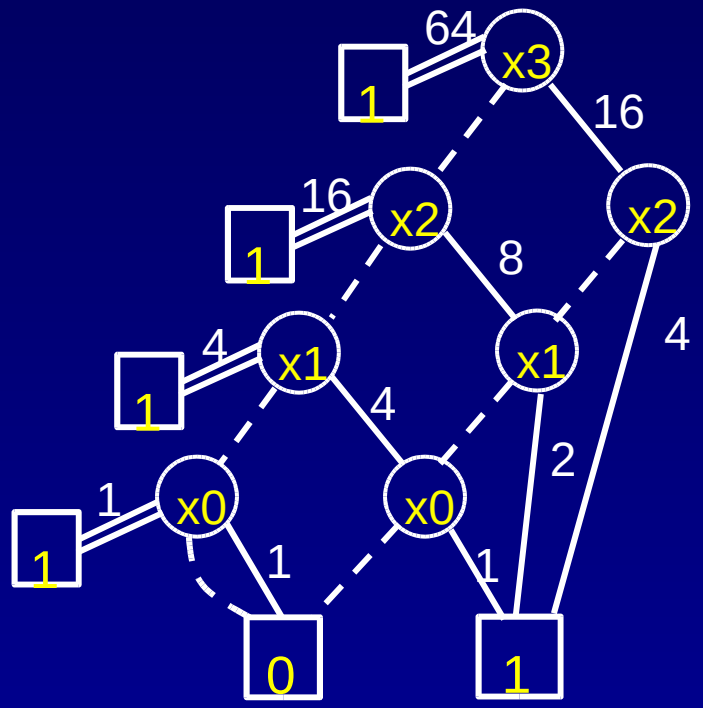


Properties of TED

- Canonical
- Compact
- Linear for polynomials of arbitrary degree
  - TED for X^k, *k* = const, with *n* bits, has *k(n-1)+1* nodes.
- Can contain symbolic, word-level, and Boolean variables
- It is *not* a Decision Diagram

n = 4, k = 2

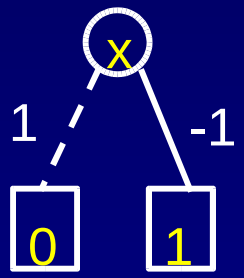
$X^2=(8x_3+4x_2+2x_1+x_0)^2$



TED for Boolean logic

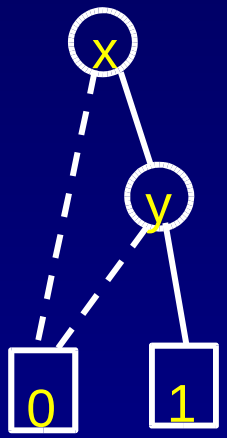
- Needed to model arithmetic-Boolean interface

$x' = (1-x)$



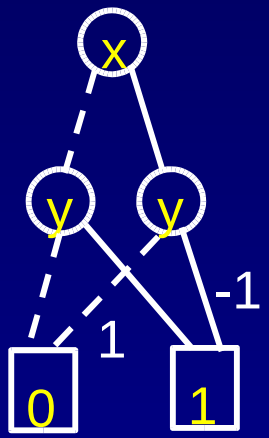
NOT

$x \wedge y = x\,y$



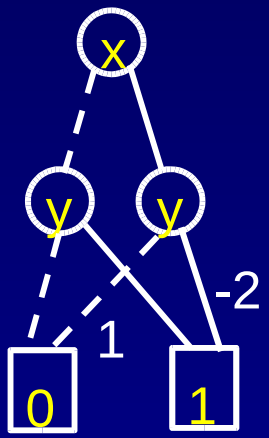
AND

$x \vee y = (x + y - x\,y)$



OR

$x \oplus y = (x + y - 2\,x\,y)$

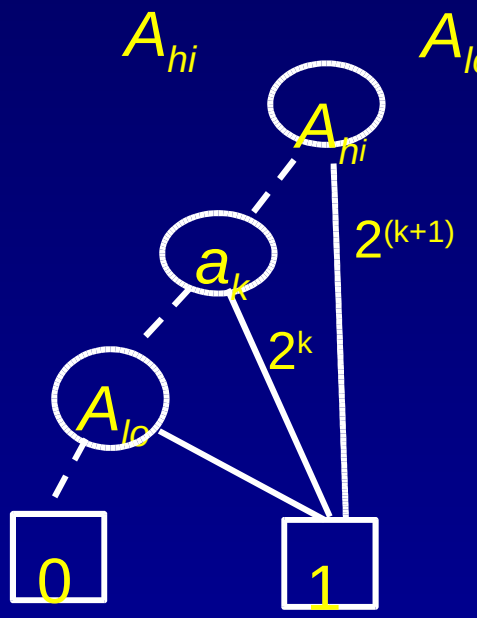


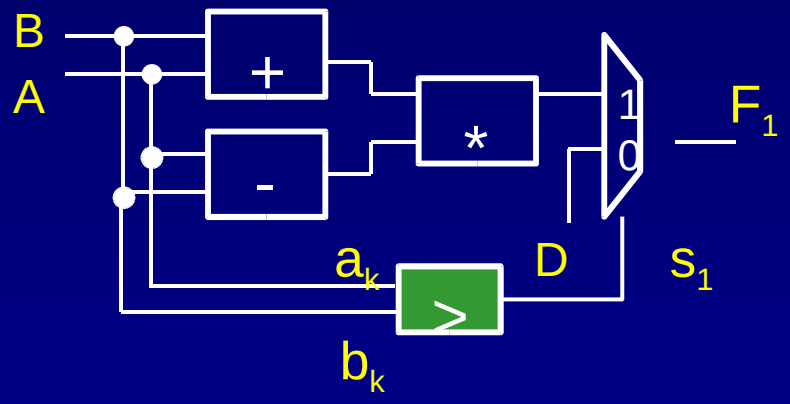
XOR

TED for Arithmetic Circuits

- Arithmetic circuits contain related word-level (*A, B*) and Boolean (a_k, b_k) variables

$$A = [\, a_{n-1}, \ldots, a_k, \ldots, a_0 \,] = 2^{(k+1)} A_{hi} + 2^k a_k + A_{lo}$$

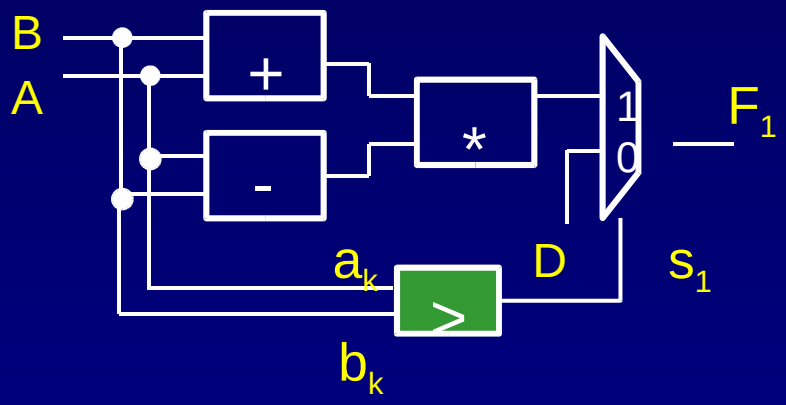


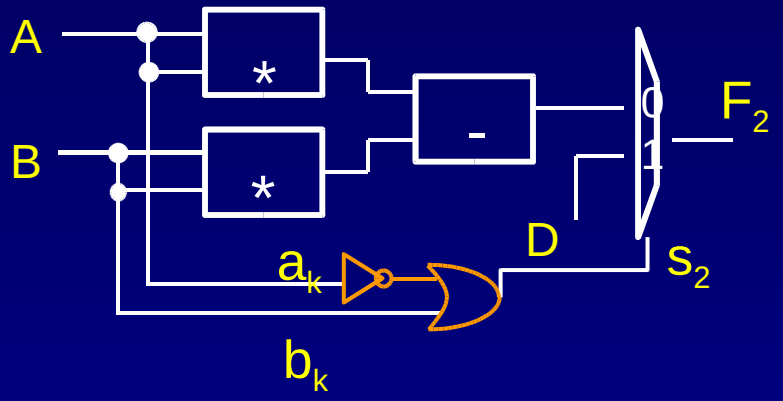


$$s_1 = a_k (1 - b_k)$$

Applications to RTL Verification

- Equivalence checking with TEDs
  - interacting word-level and Boolean variables





$F_1 = s_1(A+B)(A-B) + (1-s_1)D$

$s_1 = (a_k > b_k) = a_k\,(1-b_k)$

$F_2 = (1-s_2)\,(A^2-B^2) + s_2\,D$

$s_2 = a_k' \vee b_k = 1 - a_k + a_k\,b_k$

$A = [A_{hi}, a_k, A_{lo}], \quad B = [B_{hi}, b_k, B_{lo}]$

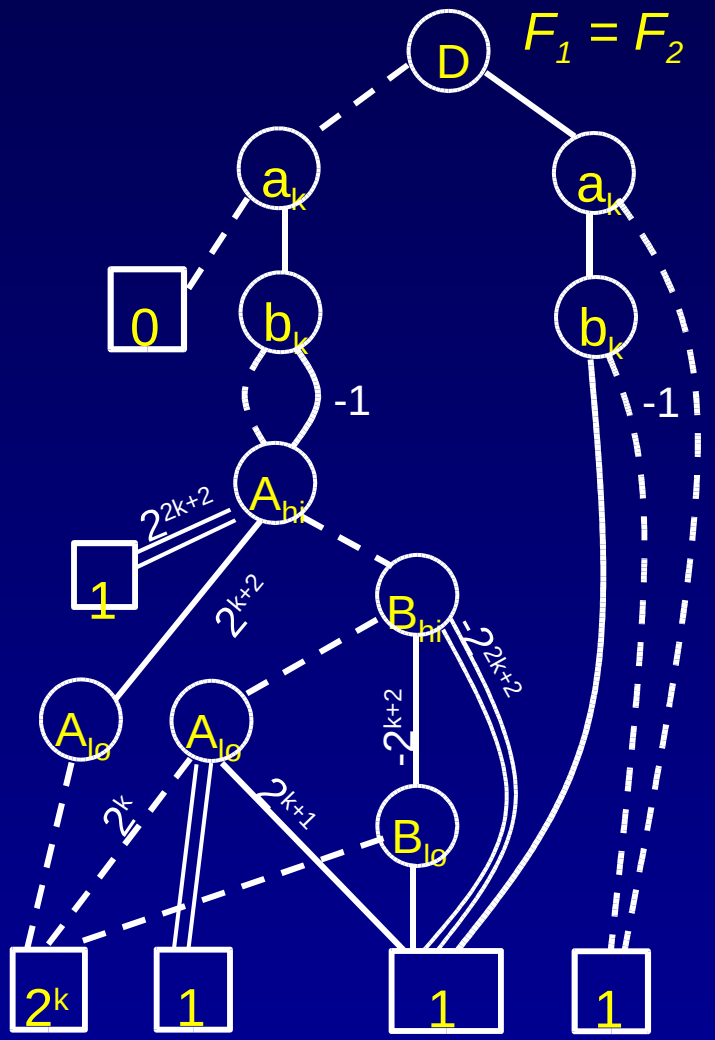
RTL Verification – Result

- Related word-level and Boolean variables

$F_1 = s_1(A+B)(A-B) + (1-s_1)D$

$s_1 = (a_k > b_k) = a_k\,(1-b_k)$

$F_1 = (a_k - a_k b_k)\{ (2^{k+1}A_{hi} + 2^k a_k + A_{lo})^2 - (2^{k+1}B_{hi} + 2^k\, b_k + B_{lo})^2 \} + (1 - a_k + a_k b_k)\, D$



Conclusions

- TED representation, features
  - Canonical
  - Compact (linear for polynomials)
  - Represents arithmetic (word-level) operators + Boolean logic
- Applications
  - Equivalence checking, symbolic simulation
- Limitations
  - TED still needs to be implemented: efficiency, robustness?
  - How to represent relations ($X < Y$) without bit expansion?
  - Normalization: more complicated than in *BMD
  - Infinite series for some functions (a^x)
- Open problems
  - Application to satisfiability (SAT), functional test generation