Digital Techniques 2

L4: Combinational Logic Design 2 (Logic Synthesis)

UNIVERSITY OF OULU
CIRCUITS and SYSTEMS

A Closer Look at the Logic Synthesis Process

mydesign.sv

```
always_comb
begin
 logic tmp;
 tmp = a[0];
 for (int i=1; i<=3; ++i)
  if (a[i] == '1)
   tmp = tmp ^ a[i];
  else
   break;
 Z = tmp;
end
```

```
set_max_delay 1 -to { Z }
```

mydesign.sdc

RTL Code

RTL TRANSLATION

GENERIC LOGIC MODEL
(flip-flops and logic functions)

Constraints

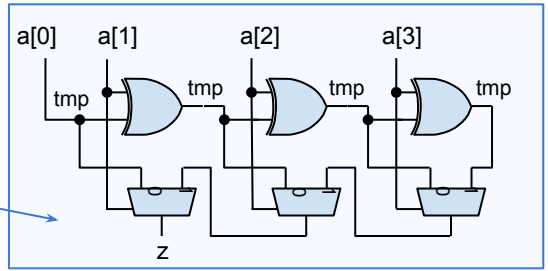
OPTIMIZATION & MAPPING

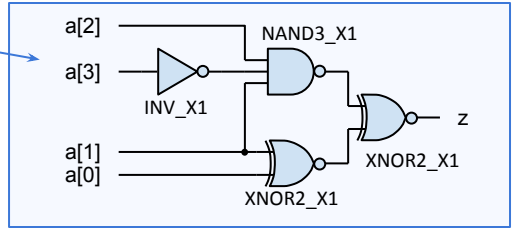
COMPONENT LIBRARY

mylib.db

Gate-Level Model
("netlist")

Tip: Execute **File > Read** in Design Vision and select an RTL code file.Then create a schematic and you will see this intermediate form design.



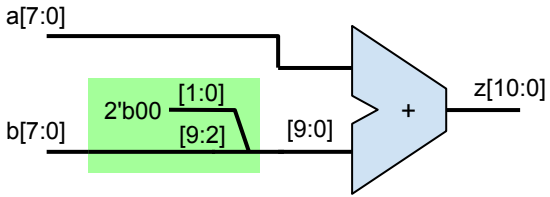


1. RTL code is translated into a generic logic model (ideal Boolean functions)
2. Designer sets constraints and selects a component library for a specific IC or FPGA technology
3. Synthesis tool optimizes logic and maps it to library components according to constraints

RTL Translation: Most Program Constructs Can Be Translated to Logic

Arithmetical/Logical Expressions

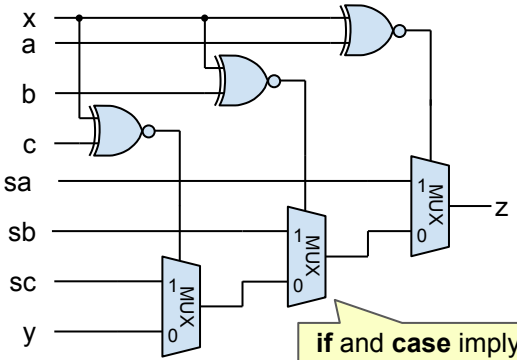
```
input logic [7:0] a, b,
output logic [10:0] z);

always_comb
  z = a + (b << 2);
```



Many combinational implementations for arithmetic operations exist. More in next lecture.

Procedural Controls

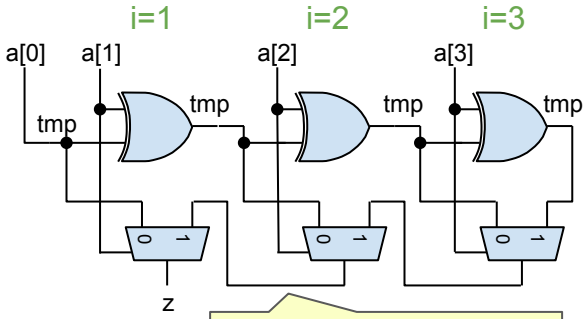
```
if ( x == a )
  z = sa;
else if (x == b)
  z = sb;
else if (x == c)
  z = sc;
else
  z = y;
```



if and **case** imply multiplexers.

Loops (finite)

```
logic tmp;
tmp = a[0];
for (int i=1; i<=3; ++i)
  if (a[i] == '1)
    tmp = tmp ^ a[i];
  else
    break;
Z = tmp;
```



Loops are unrolled which implies replication of logic.

Boolean Logic Optimization 1

Assume a design specified by the logic equations:

X = (A*B + A'*B')*(C + D) // * = AND, + = OR, ' = NOT

Y = (A*B + A'*B')'*(C'*D)

X and Y have a common term: (A*B + A'*B')

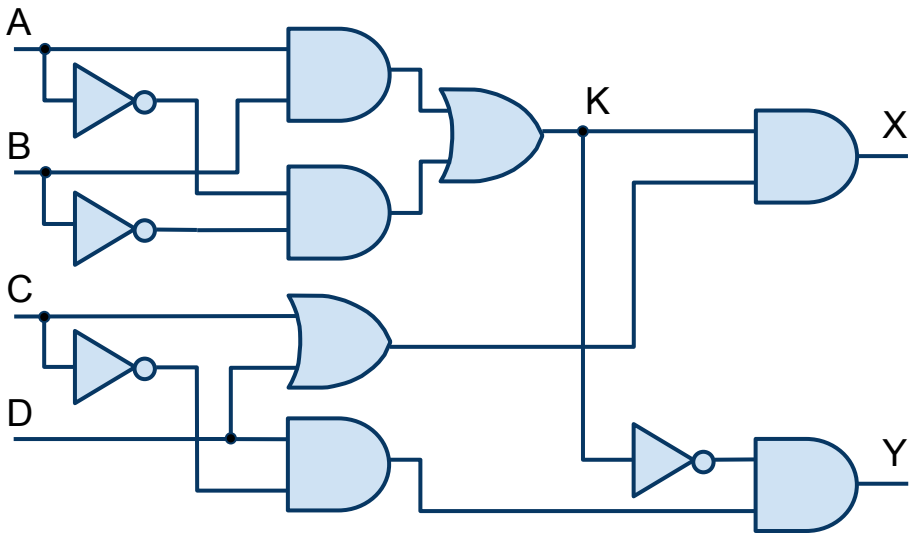
We can factor out the common term

K = (A*B + A'*B')

X = K*(C + D)

Y = K'*(C'*D)

By implementing these **factored forms** of the functions fewer components are needed as logic in K is not replicated.



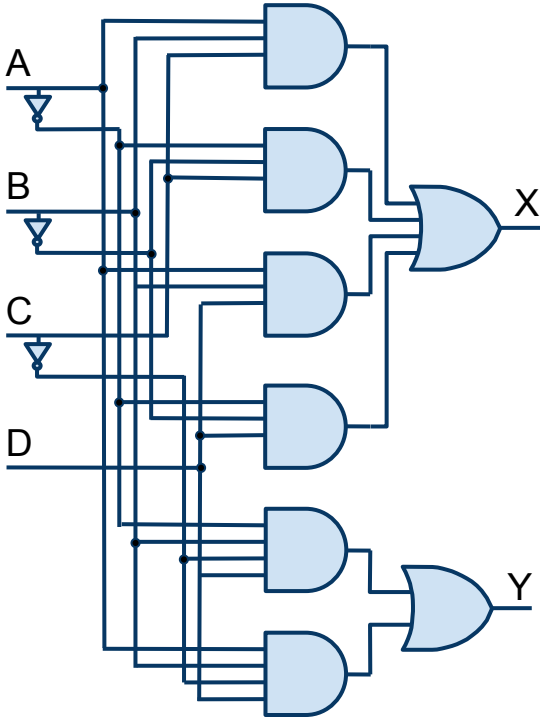
Boolean Logic Optimization 2

Using Boolean algebra, we can try to get rid of the parentheses to see where that might lead:

X = (A*B + A'*B') * (C + D)
  = A*B*C + A'*B'*C + A*B*D + A'*B'*D

Y = (A*B)' * (A'*B')' * (C'* D)
  = (A' + B') * (A + B) * (C'* D)
  = (A'*A + A'*B + B'*A + B'*B) * (C'*D)
  = (A'*B + A*B') * (C'*D)
  = A'*B*C'*D + A*B'*C'*D

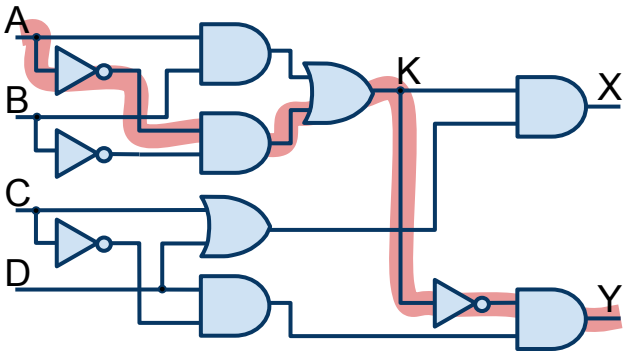
We get a **two-level** (AND-OR) implementation. The logic gate implementation looks different from the multi-level version, but the functions are logically equivalent to those on the previous slide.



Comparison of Factored and Two-Level Functions

FACTORED:

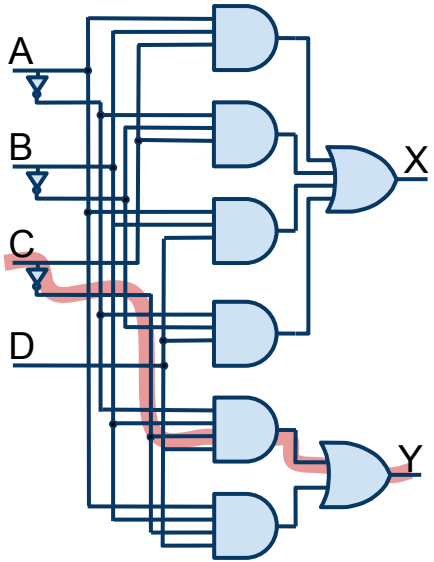
- Longest delay path: 5 gates
- Complexity: 11 gates, 18 gate inputs (= ~36 transistors)
- Conclusion: Small but slow



(many other alternatives exist)

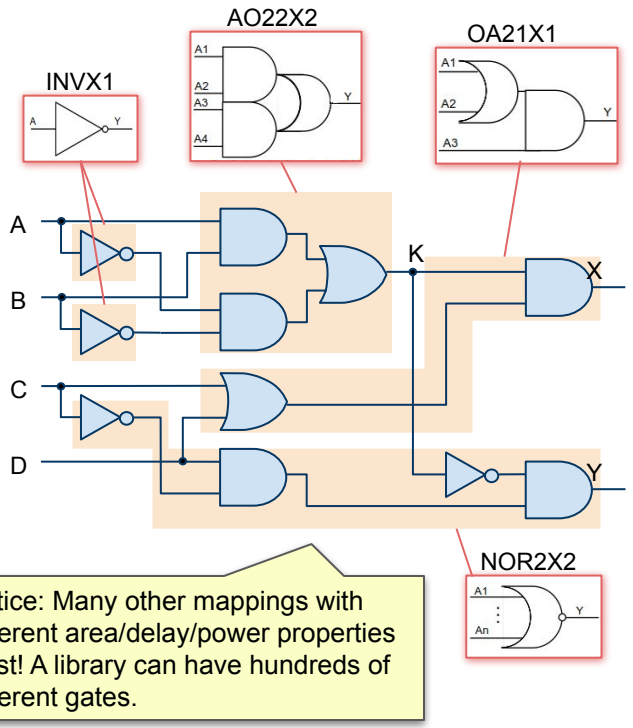
TWO-LEVEL (S-O-P):

- Longest delay path: 3 gates
- Complexity: 11 gates, 28 gate inputs
- Conclusion: Fast but big



Technology Mapping

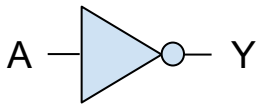
- After logic optimization, "reorganized" generic logic gates are replaced with real logic gates chosen from a component library
- The goal of this "mapping" is to cover the original "tree" with library cells so that some optimization goal is met
- Many kind of mapping optimizations are possible, e.g.
  - Selection of library gates to minimize delay or area
  - Power optimization by selecting high-drive gates ("X2") on critical paths and low-drive ("X1") and low-leakage gates on non-critical paths

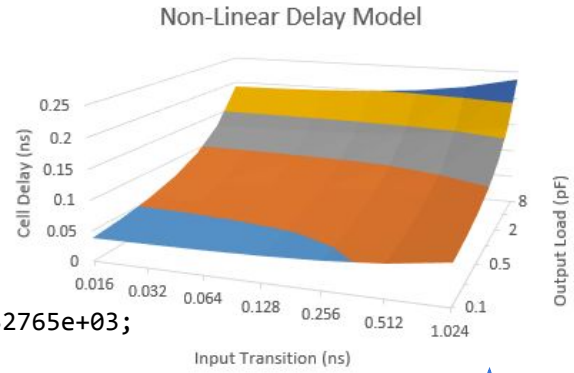


Logic Libraries

- Components available in a target technology are described in a **logic library**
- Component models include:
  - Output pins' logic function
  - Area (corresponds to silicon area)
  - Power consumption
  - Timing
- Timing models are complex **nonlinear delay models** (NLDM) that specify output pins' rise and fall times as a function of input pins' transition times and output pins load capacitance values
- By changing the library, the design can be "mapped" to a different technology

INVX1_HVT





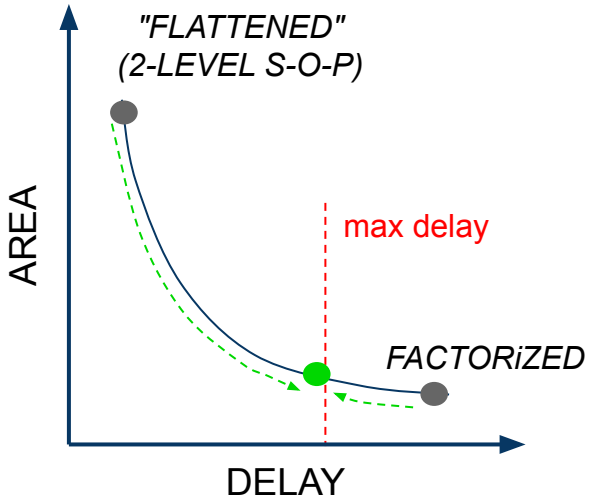

```
cell (INVX1_HVT) {
 area : 1.27072 ;
 cell_leakage_power : 6.832765e+03;
 pin (Y) {
  function : "(A)'";
  timing () {
   related_pin : "A";
   cell_rise ("del_1_7_7") {
    index_1("0.016, 0.032, 0.064, 0.128, 0.256, 0.512, 1.024");
    index_2("0.1, 0.25, 0.5, 1, 2, 4, 8");
    values("0.0115926, 0.0124960, 0.0138466, 0.0162092,  ...", \
           "0.0152338, 0.0165709, 0.0185594, 0.0219617, ...", \
  internal_power () {
   related_pin : "A";
   rise_power ("power_outputs_1") {
    index_1("0.016, 0.032, 0.064, 0.128, 0.256, 0.512, 1.024");
    index_2("0.1, 0.25, 0.5, 1, 2, 4, 8");
    values("0.9903715, 0.9937907, 0.9992987, 1.0122674, ... \
           0.9807714, 0.9871708, 0.9850473, 0.9904108, ...", \
```

Logic Synthesis Usage

- For complex combinational logic functions with many inputs and outputs, a large number of different factorized forms exist
- To optimize a design, the designer sets a maximum delay constraint and lets the optimizer find a minimum-area (component count) solution the meets the constraint
- Minimal synthesis script (Synopsys Design Compiler)

```
set target_library saed32hvt_tt1p05v25c.db
read_file -format sverilog my_design.sv
read_sdc my_constraints.sdc
compile_ultra
```

- Learn more about logic synthesis in 521448S Physical Design of Digital Integrated Circuits



Area-delay trade-off of most combinational designs can be described as a "banana curve"