KUNGL TEKNISKA HÖGSKOLAN

International Master Program in System-on-Chip Design

L9: Galois Fields

Reading material

- Muzio & Wesselkamper "Multiple-valued switching theory", p. 3 - 5, 101 - 104
- Sasao, "Switching theory for logic synthesis", pp. 43 - 44

Motivation

- There are alternative to Boolean algebras for repsenting Boolean functions, e.g. Galois Field (GF(2))
- Some functions have much shorter expression in GF(2) as compared to SOP expression in Boolean algebra
  - e.g. n-variable parity function has 2^{n-1} products in SOP and just n in GF(2)

Fields

- Informally, a field is an algebra over a set upon which the operations of addition, subtraction, multiplication and division are all defined
- More formally, an algebra $\langle E; +, \cdot; \mathbf{0}, \mathbf{1} \rangle$ is a field is the following set of axioms holds for binary operations "+", "·" and some distinct elements **0** and **1** of a set E

Axioms of a field

A1: $a,b \in E \Rightarrow a+b, a \cdot b \in E$

A2: $\forall a,b \in E, a \cdot b = b \cdot a, a + b = b + a$

A3: $\forall a,b,c \in E, a \cdot (b+c) = a \cdot b + a \cdot c$

A4: $\forall a \in E, a \cdot \mathbf{1} = a, a + \mathbf{0} = \mathbf{a}$

A5: $\forall a \in E \; \exists b \in E$ such that $a+b = \mathbf{0}$,
$\forall a \in E - \{0\} \; \exists c \in E$ such that $a \cdot c = \mathbf{1}$

A6: $\forall a,b,c \in E, a \cdot (b \cdot c) = (a \cdot b) \cdot c, (a+b)+c = a+(b+c)$

Axioms of a field (cont.)

- A5 means that all elements in E have an inverse w.r.t. addition and all non-zero elements have an inverse w.r.t. multiplication
- If E is finite, we have a finite field

Galois Field GF(2)

- Galois Field GF(2) is defined as GF(2) := {B; ⊕, ·; 0, 1} where
  - B = {0,1}
  - "⊕" is the binary operation XOR
  - "·" is the binary operation AND
  - GF(2) is functionally complete with constants

Example 1 of a field - GF(p)

- If
  - E = {0,1,…, p}, p - prime
  - "+" = addition mod p
  - "·" = multiplication mod p
- then all the axioms are satisfied and ⟨{0,1 ,…, p}; +, ·; 0,1⟩ is a field, called Galois field of order p

Example 2 of a field - GF(2)

- If p = 2, the addition mod 2 = XOR ($\oplus$) and the multiplication mod 2 = AND
- {XOR, AND, 1} is functionally complete for for Boolean functions f: $\{0,1\}^n \rightarrow \{0,1\}$

Reed-Muller canonical form (or Algebraic Normal Form)

- Developed in 1954 by Reed and Muller
- Any Boolean function f: $\{0,1\}^n \rightarrow \{0,1\}$ has a canonical form of type:

$$f(x_1,x_2,\ldots,x_n) = \sum_{i=0}^{2^n-1} c_i \cdot x_1^{i_1} \cdot x_1^{i_2} \cdot \ldots \cdot x_n^{i_n}$$

where

- $\sum$ is XOR-sum and $c_i \in \{0,1\}$ is a constant
- $(i_1,i_2,\ldots,i_n)$ is the binary expansion of i
- $x_k^{i_k} = 1$ if $i_k = 0$ and $x_k^{i_k} = x$ if $i_k = 1$

Difference from AND-OR SOP form

- "sum" is XOR
- product-terms have different lengths
- Example:

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

AND-OR form: $f(x_1,x_2) = x'_1 x_2 + x_1 x'_2$

AND-XOR form: $f(x_1,x_2) = x_1 \oplus x_2$

An algorithm for finding Reed-Muller canonical form

- Multiply the truth table of function $f(x_1,x_2, ...,x_n)$ by the transformation matrix T^n, defined by

$$T^1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$T^n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes T^{n-1}$$

where "$\otimes$" denotes the Kroneker product

Examples of transformation matrices for n=2 and n=3

$$T^2 = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}\right] \qquad T^3 = \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

Example of computing RM form

- Let $f(x_1,x_2,x_3) = x'_1x_2 + x_1x_3$

$$\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right] \cdot \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{array}\right] \begin{array}{l} \\ \\ x_2 \\ \\ \\ x_1x_3 \\ x_1x_2 \\ \end{array} \qquad \begin{array}{ccc} x_1 & x_2 & x_3 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}$$

$f = x_1x_2 \oplus x_1x_3 \oplus x_2$

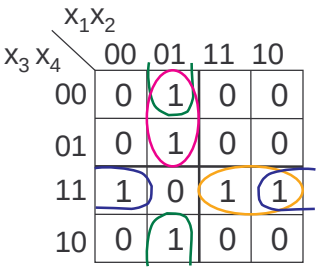
Minimization

- The number of products in the RM canonical form is up to 2^n
- It can be simplified using the axioms and properties of GF(2)
  - e.g. the number of products can be reduced by using the property of XOR

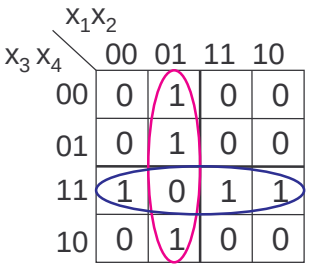
    $x \oplus x = 0$

    so, any minterm from the off-set can be covered by an even number of implicants

Example



AND-OR cover



AND-XOR cover

Why minimizing RM form?

- an RM expression can be directly implemented by a Programmable Logic Array with XOR plane
- the number of products in the RM expression corresponds to the number of columns in PLA

Formulation of the AND-XOR two-level minimization problem

input: a Boolean function $f(x_1,x_2, ...,x_n)$

output: an expression for f of type

$$f(x_1,x_2, ...,x_n) = P_1 \oplus P_2 \oplus ... \oplus P_k$$

with the minimal number k of products P_k

Extension: Fixed/mixed polarity Reed-Muller canonical forms

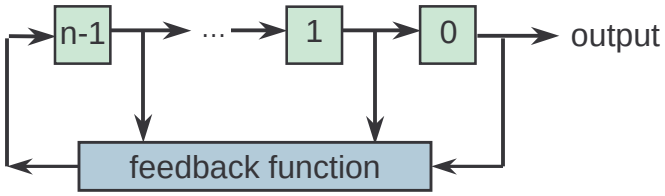
- Fixed polarity: Variables are allowed to appear in either complemented or un-complemented form (but not both), according to the polarity vector
- Mixed polarity: Variables are allowed to appear in both complemented or un-complemented form

Applications of RM canonical form

- Design of easily testable circuits
  - A circuit implementing RM canonical form of an n-variable function needs only 2n+4 tests to detect all single stuck-at fault
    - $O(2^n)$ tests for a random circuit
  - only 4 universal tests are needed if an n-input AND with an observable output is added to the circuit implementing RM canonical form

Applications of RM canonical form

- Design of Feedback Shift Registers (FSRs)

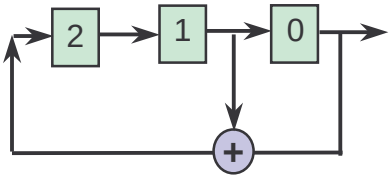


- An FSR consists of n bits with feedback from each to n-1st
- At each clocking instance, the value of the bit i is moved to the bit i-1. The value of the bit 0 becomes the output
- The new value of the bit n-1 is computed as some function of the previous values of other bits

Feedback functions of FSRs

- Feedback functions are usually represented in RM canonical form
- Example: 3-bit linear FSR

  $f = x_0 \oplus x_1$



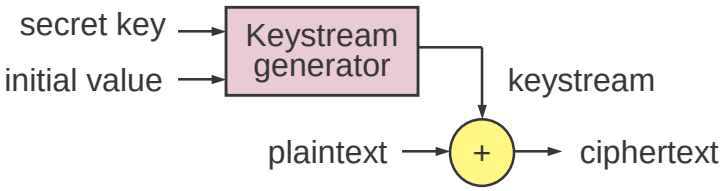
- Example: 32-bit non-linear FSR

$$f = x_0 \oplus x_2 \oplus x_6 \oplus x_7 \oplus x_{12} \oplus x_{17} \oplus x_{20} \oplus x_{27} \oplus x_{30} \oplus x_3 x_9 \oplus x_{12} x_{15} \oplus x_4 x_5 x_{16}$$

Applications of FSRs

- FSRs are used extensively for generating pseudo-random sequences for a number of applications:
  - Pseudo-random testing (LFSRs)
  - Monte Carlo simulation (LFSRs)
  - Stream ciphers (LFSRs and NLFSRs)

Binary Additive Stream Cipher

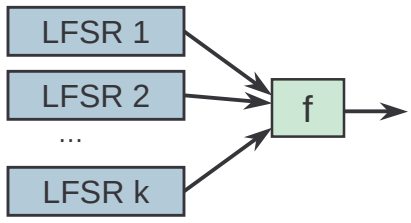


- Generation of truly random sequences is too costly
- Desirable properties of pseudo-random sequence generators:
  - Large period length
  - Good statistical properties

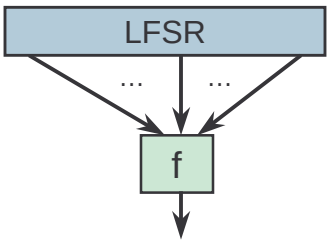
LFSRs

- Linear Feedback Shift Registers (LFSRs):
  - + Easy to implement in both, software and hardware, fast
  - + Period grows exponentially with the size
  - + It is known how to construct an LFSR with the maximal period
  - + Sequences have good statistical properties
  - – Sequences are easy to predict
    - LFSRs are combined with a non-linear function

Combining LFSRs with non-linear functions



Non-linear combining function



Non-linear filter

NLFSRs

- Non-linear Feedback Shift Registers (NLFSRs):
  - + Easy to implement, fast
  - + Period grows exponentially with the size
  - + Sequences have good statistical properties
  - + Sequences are hard to predict
  - – It is not known how to construct an NLFSR with the maximal period

Optimization of feedback functions

- The number of terms in the RM canonical form of a feedback function translates into the number of gates in the FSR
- The depth of the circuit implementing the feedback function translates into the throughput of the FSR
- Current implementations of FSR-based stream ciphers do not satisfy requirements of some constrained environments applications, e.g. RFID