Computer Architecture

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
⟨csdsp@bristol.ac.uk⟩

October 14, 2024

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
   - the associated notes page may be pre-populated with extra, written explaination of material covered in lecture(s), plus
   - anything with a "grey'ed out" header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- Problem: we have $\neg$, $\wedge$, and $\vee$, so given the specification

x_{n-1}	$\cdots$	x_1	x_0	r
0	$\cdots$	0	0	1
0	$\cdots$	0	1	0
0	$\cdots$	1	0	1
0	$\cdots$	1	1	0
$\vdots$		$\vdots$	$\vdots$	$\vdots$
1	$\cdots$	1	1	0

  for some Boolean function

$$r = f(x_0, x_1, \ldots, x_{n-1}),$$

  design a Boolean expression e which can compute it.
- Solution: ?

Notes:

- Agenda: **combinatorial logic** design, where, crucially,
  - the output is a function of the input only,
  - computation is viewed as being continuous,

  via coverage of

  1. special-purpose design patterns,
  2. special-purpose building blocks, and
  3. general-purpose derivation.

Notes:

- Pattern #1: **decomposition**.
  - Any n-input, m-output Boolean function

$$f : \mathbb{B}^n \rightarrow \mathbb{B}^m$$

can be rewritten as m *separate* n-input, 1-output Boolean functions, say

$$\begin{array}{lcl} f_0 & : & \mathbb{B}^n \rightarrow \mathbb{B} \\ f_1 & : & \mathbb{B}^n \rightarrow \mathbb{B} \\ & \vdots & \\ f_{m-1} & : & \mathbb{B}^n \rightarrow \mathbb{B} \end{array}$$

  - As such, we have

$$f(x) \;\equiv\; f_0(x) \parallel f_1(x) \parallel \ldots \parallel f_{m-1}(x).$$

Notes:

Part 1: special-purpose design patterns

- Pattern #2: **sharing**.
  - Imagine, for example, that we are given a 2-input, 1-bit AND gate.
  - If, within some larger circuit, we compute

$$r = x \wedge y$$

and then, somewhere else,

$$r' = x \wedge y$$

then we can replace the two AND gates with one: clearly

$$r = r',$$

so we can share one definition between two usage points.

Notes:

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

- Pattern #3: **independent replication**.
  - Imagine, for example, that we are given a 2-input, 1-bit AND gate.
  - A 2-input, m-bit AND gate is simply replication of 2-input, 1-bit AND gates, i.e.,

    $$r = x \wedge y$$

    is computed via

    $$r_i = x_i \wedge y_i$$

    for $0 \leq i < m$,
  - for $n = 4$, as an example, this means

    $$\begin{array}{lcl} r_0 & = & x_0 \wedge y_0 \\ r_1 & = & x_1 \wedge y_1 \\ r_2 & = & x_2 \wedge y_2 \\ r_3 & = & x_3 \wedge y_3 \end{array}$$

Notes:

- Pattern #4: **dependent replication**.
  - Imagine, for example, that we are given a 2-input, 1-bit AND gate.
  - An n-input, 1-bit AND gate is simply replication of 2-input, 1-bit AND gates, i.e.,

    $$r = \bigwedge_{i=0}^{n-1} x_i$$

    is computed via

    $$r = x_0 \wedge (x_1 \wedge \cdots (x_{n-1})),$$
  - for $n = 4$, as an example, this means

    $$\begin{array}{lcl} r & = & x_0 \wedge (x_1 \wedge x_2 \wedge (x_3)) \\ & = & x_0 \wedge x_1 \;\; \wedge \;\; x_2 \wedge x_3 \\ & = & (x_0 \wedge x_1) \wedge (x_2 \wedge x_3) \end{array}$$

Notes:

Part 2: special-purpose building blocks (1)
Choice

- Concept: the following building blocks can support most forms of choice
  1. a **multiplexer**
     - has m inputs,
     - has 1 output,
     - uses a $(\lceil \log_2(m) \rceil)$-bit control signal input to choose which input is connected to the output,

     while
  2. a **demultiplexer**
     - has 1 input,
     - has m outputs,
     - uses a $(\lceil \log_2(m) \rceil)$-bit control signal input to choose which output is connected to the input,

  noting that
  - the input(s) and output(s) are n-bit, but clearly must match up,
  - the connection made is continuous, since both components are combinatorial.

Notes:

Part 2: special-purpose building blocks (2)
Choice

- Concept: by analogy,
  1. the C `switch` statement

     Listing

     ```
     switch( c ) {
       case 0 : r = w; break;
       case 1 : r = x; break;
       case 2 : r = y; break;
       case 3 : r = z; break;
     }
     ```

     acts similarly to a 4-input multiplexer,
  2. the C `switch` statement

     Listing

     ```
     switch( c ) {
       case 0 : r0 = x; break;
       case 1 : r1 = x; break;
       case 2 : r2 = x; break;
       case 3 : r3 = x; break;
     }
     ```

     acts similarly to a 4-output demultiplexer.

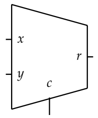
Notes:

Part 2: special-purpose building blocks (3)

Choice

Definition

The behaviour of a 2-**input,** 1-**bit multiplexer** component



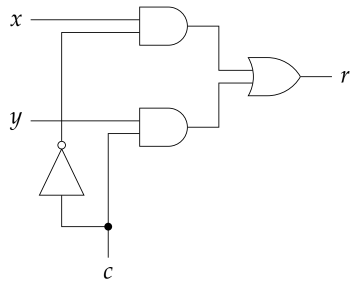
is described by the truth table

c	x	y	r
0	0	?	0
0	1	?	1
1	?	0	0
1	?	1	1

which can be used to derive the following implementation:

$$r = (\neg c \wedge x) \vee (c \wedge y)$$

Circuit



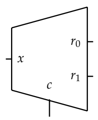
Notes:

Part 2: special-purpose building blocks (4)

Choice

Definition

The behaviour of a 2-**output,** 1-**bit demultiplexer** component



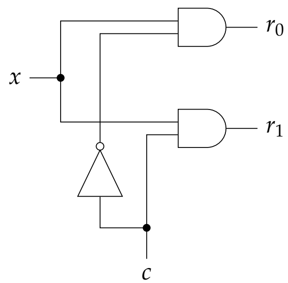
is described by the truth table

c	x	r_1	r_0
0	0	?	0
0	1	?	1
1	0	0	?
1	1	1	?

which can be used to derive the following implementation:

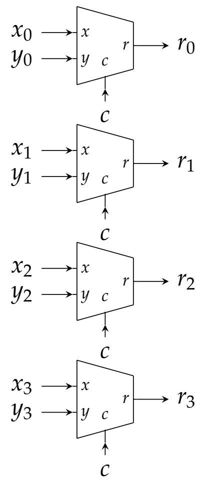
$$\begin{aligned} r_0 &= \neg c \wedge x \\ r_1 &= c \wedge x \end{aligned}$$

Circuit



Notes:

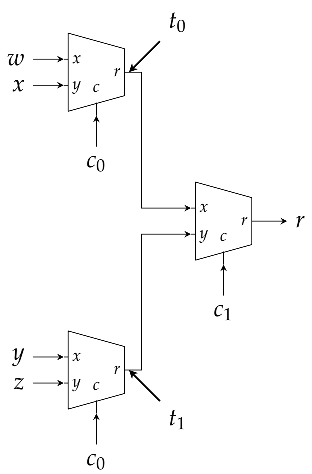
Circuit (2-input, 4-bit multiplexer via independent replication)



Notes:

An Aside: application of design patterns

Circuit (4-input, 1-bit multiplexer via dependent replication)



Notes:

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

- Concept: the following building blocks can support most forms of arithmetic
  1. a **half-adder**
     - has 2 inputs: x and y,
     - computes the 2-bit result $x + y$,
     - has 2 outputs: a sum s, and a carry-out co (which are the LSB and MSB of result),

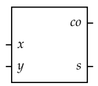
     while
  2. a **full-adder**
     - has 3 inputs: x and y plus a carry-in ci,
     - computes the 2-bit result $x + y + ci$,
     - has 2 outputs: a sum s, and a carry-out co (which are the LSB and MSB of result),

  where all inputs and outputs are 1-bit.

Notes:

Definition

The behaviour of a **half-adder** component



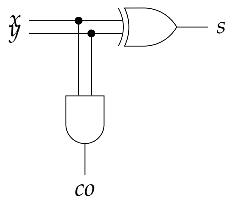
is described by the truth table

x	y	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

which can be used to derive the following implementation:

$$\begin{array}{lcl} co & = & x \wedge y \\ s & = & x \oplus y \end{array}$$

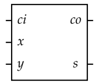
Circuit



Notes:

Definition

The behaviour of a **full-adder** component



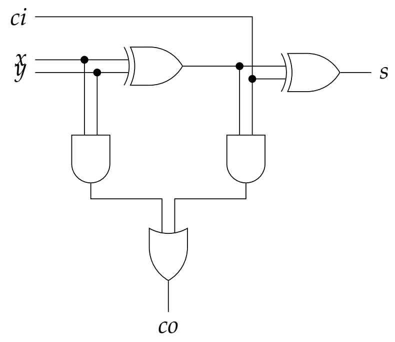
is described by the truth table

ci	x	y	co	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

which can be used to derive the following implementation:

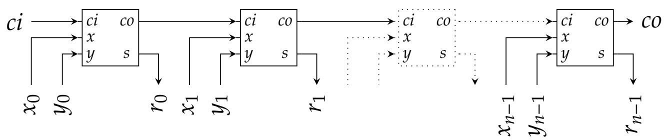
$$
\begin{array}{lcl}
co & = & (x \wedge y) \vee (x \wedge ci) \vee (y \wedge ci) \\
 & = & (x \wedge y) \vee ((x \oplus y) \wedge ci) \\
s & = & x \oplus y \oplus ci
\end{array}
$$

Circuit



Notes:

Circuit (n-bit addition)



Notes:

- **Concept**: the following building blocks can support most forms of comparison
  1. an **equality comparator**
     - has 2 inputs x and y,
     - computes the 1 output as

$$r = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

  while

  2. a **less-than comparator**
     - has 2 inputs x and y,
     - computes the 1 output as

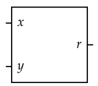
$$r = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{otherwise} \end{cases}$$

  where all inputs and outputs are 1-bit.

Notes:

Definition

The behaviour of an **equality comparator** component



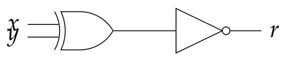
is described by the truth table

x	y	r
0	0	1
0	1	0
1	0	0
1	1	1

which can be used to derive the following implementation:

$$r = \neg(x \oplus y)$$

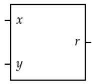
Circuit



Notes:

Definition

The behaviour of a **less-than comparator** component



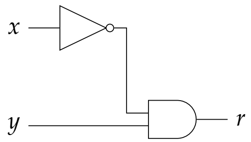
is described by the truth table

x	y	r
0	0	0
0	1	1
1	0	0
1	1	0

which can be used to derive the following implementation:

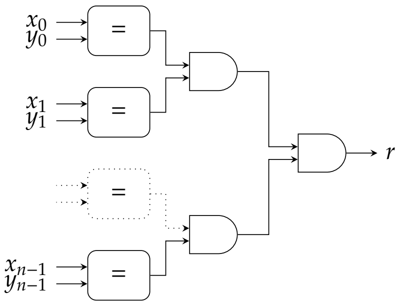
$$r = \neg x \wedge y$$

Circuit



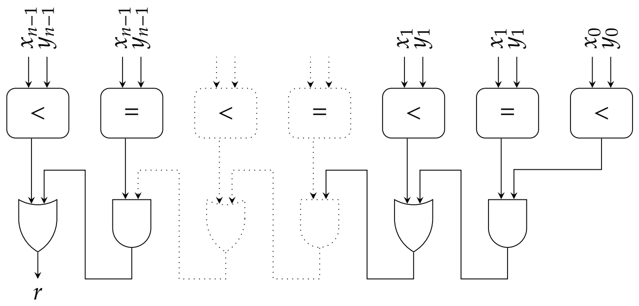
Notes:

Circuit (n-bit equality comparison)



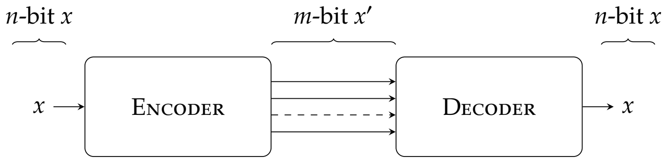
Notes:

Circuit (*n*-bit less-than comparison)



Notes:

- Concept: informally, **encoders** and **decoders** can be viewed as *translators*, i.e.,



  or, more formally,

  1. an n-to-m encoder translates an n-bit input into some m-bit code word, and
  2. an m-to-n decoder translates an m-bit code word back into the same n-bit output

  where if only one output (resp. input) is allowed to be 1 at a time, we call it a **one-of-many** encoder (resp. decoder).

Notes:

Part 2: special-purpose building blocks (16)
Control

- A *general* building block is impossible since it depends on the scheme for encoding/decoding: consider an example such that
  1. to encode, take n inputs, say x_i for $0 \leq i < n$, and produce a unsigned integer x' that determines which $x_i = 1$,
  2. to decode, take x' and set the correct $x_i = 1$

  where for all $j \neq i$, $x'_j = 0$.

Notes:

Part 2: special-purpose building blocks (17)
Control

Definition (example encoder)

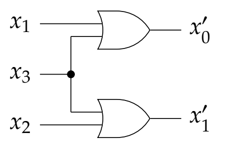
The example encoder is described by the truth table

x_3	x_2	x_1	x_0	x'_1	x'_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

which can be used to derive the following implementation:

$$\begin{array}{lcl} x'_0 & = & x_1 \vee x_3 \\ x'_1 & = & x_2 \vee x_3 \end{array}$$

Circuit (example encoder)



Notes:

Definition (example decoder)

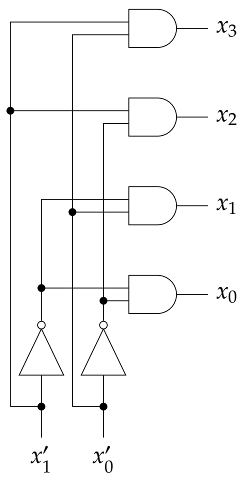
The example decoder is described by the truth table

x'_1	x'_0	x_3	x_2	x_1	x_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

which can be used to derive the following implementation:

$$\begin{array}{lcccc} x_0 & = & \neg x'_0 & \wedge & \neg x'_1 \\ x_1 & = & x'_0 & \wedge & \neg x'_1 \\ x_2 & = & \neg x'_0 & \wedge & x'_1 \\ x_3 & = & x'_0 & \wedge & x'_1 \end{array}$$

Circuit (example decoder)



Notes:

- Problem: if we break the rules and set both $x_1 = 1$ *and* $x_2 = 1$, the encoder fails by producing

$$\begin{array}{lclcl} x'_0 & = & x_1 \vee x_3 & = & 1 \\ x'_1 & = & x_2 \vee x_3 & = & 1 \end{array}$$

  as the result.
- Solution: consider a **priority** encoder, where one input is given priority (or preference) over another.

Notes:

Notes:

Example

Imagine we want to give x_j priority over each x_k for $j > k$, so x_2 over x_1 and x_0 for example:

x_3	x_2	x_1	x_0	x'_1	x'_0
0	0	0	1	0	0
0	0	1	**?**	0	1
0	1	**?**	**?**	1	0
1	**?**	**?**	**?**	1	1

Now, although potentially $x_0 = 1$ or $x_1 = 1$ the output gives priority to x_2: as long as $x_2 = 1$ and $x_3 = 0$, the output will be $x'_0 = 0$ and $x'_1 = 1$ irrespective of x_0 and x_1.

Notes:

Algorithm

Input: A truth table for some Boolean function f, with n inputs and 1 output
Output: A Boolean expression e that implements f

First let $\mathcal{I}_j$ denote the j-th input for $0 \leq j < n$ and $\mathcal{O}$ denote the single output:

1. Find a set T such that $i \in T$ iff. $\mathcal{O} = 1$ in the i-th row of the truth table.
2. For each $i \in T$, form a term t_i by AND'ing together all the variables while following two rules:
   2.1 if $\mathcal{I}_j = 1$ in the i-th row, then we use
   $$\mathcal{I}_j$$
   as is, but
   2.2 if $\mathcal{I}_j = 0$ in the i-th row, then we use
   $$\neg \mathcal{I}_j.$$
3. An expression implementing the function is then formed by OR'ing together all the terms, i.e.,
   $$e = \bigvee_{i \in T} t_i,$$
   which is in SoP form.

Example

Consider the example of deriving an expression for XOR, i.e.,

$$r = f(x, y) = x \oplus y,$$

a function described by the following truth table:

f		
x	y	r
0	0	0
0	1	1
1	0	1
1	1	0

Notes:

Example

Consider the example of deriving an expression for XOR, i.e.,

$$r = f(x, y) = x \oplus y,$$

a function described by the following truth table:

f			
x	y	r	
0	0	0	
0	1	1	$\leadsto$ $i = 1$
1	0	1	$\leadsto$ $i = 2$
1	1	0	

Following the algorithm produces:

1. Looking at the truth table, it is clear there are
   - $n = 2$ inputs that we denote $\mathcal{I}_0 = x$ and $\mathcal{I}_1 = y$, and
   - one output that we denote $\mathcal{O} = r$.

   Clearly $T = \{1, 2\}$ since $\mathcal{O} = 1$ in rows 1 and 2, while $\mathcal{O} = 0$ in rows 0 and 3.

Notes:

Example

Consider the example of deriving an expression for XOR, i.e.,

$$r = f(x, y) = x \oplus y,$$

a function described by the following truth table:

f			
x	y	r	
0	0	0	
0	1	1	$\rightsquigarrow \quad t_1 = \neg x \wedge y$
1	0	1	$\rightsquigarrow \quad t_2 = x \wedge \neg y$
1	1	0	

Following the algorithm produces:

2. Each term t_i for $i \in T = \{1, 2\}$ is formed as follows:
   - For $i = 1$, we find
     - $\mathcal{I}_0 = x = 0$ and so we use $\neg x$,
     - $\mathcal{I}_1 = y = 1$ and so we use y

     and hence form the term $t_1 = \neg x \wedge y$.
   - For $i = 2$, we find
     - $\mathcal{I}_0 = x = 1$ and so we use x,
     - $\mathcal{I}_1 = y = 0$ and so we use $\neg y$

     and hence form the term $t_2 = x \wedge \neg y$.

Notes:

Part 3: general-purpose derivation (2)
Method #1

Example

Consider the example of deriving an expression for XOR, i.e.,

$$r = f(x, y) = x \oplus y,$$

a function described by the following truth table:

f			
x	y	r	
0	0	0	
0	1	1	$\rightsquigarrow \quad t_1 = \neg x \wedge y$
1	0	1	$\rightsquigarrow \quad t_2 = x \wedge \neg y$
1	1	0	

Following the algorithm produces:

3. The expression implementing the function is therefore

$$\begin{array}{rcl} e & = & \bigvee_{i \in T} t_i \\ & = & \bigvee_{i \in \{1,2\}} t_i \\ & = & (\neg x \wedge y) \vee (x \wedge \neg y) \end{array}$$

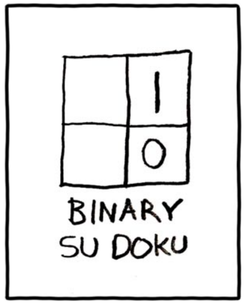
   which is in SoP form.

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

Notes:

Part 3: general-purpose derivation (3)

Method #2: Karnaugh map

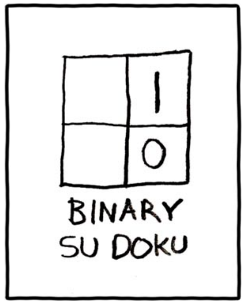


https://xkcd.com/74

Notes:

Part 3: general-purpose derivation (3)

Method #2: Karnaugh map



- Idea:

$$
\begin{array}{lcll}
(x \wedge y) \vee (x \wedge \neg y) & \equiv & x \wedge (y \vee \neg y) & \text{(distribution)} \\
 & \equiv & x \wedge 1 & \text{(inverse)} \\
 & \equiv & x & \text{(identity)}
\end{array}
$$

https://xkcd.com/74

Notes:

Part 3: general-purpose derivation (4)

Method #2: Karnaugh map

Algorithm

Input: A truth table for some Boolean function f, with n inputs and 1 output
Output: A Boolean expression e that implements f

1. Draw a rectangular $(p \times q)$-element grid, st.
   1.1 $p \equiv q \equiv 0 \pmod{2}$, and
   1.2 $p \cdot q = 2^n$

   and each row and column represents one input combination; order rows and columns according to a **Gray code**.
2. Fill the grid elements with the output corresponding to inputs for that row and column.
3. Cover rectangular groups of adjacent 1 elements which are of total size 2^m for some m; groups can "wrap around" edges of the grid and overlap.
4. Translate each group into one term of an SoP form Boolean expression e where
   4.1 *bigger* groups, and
   4.2 *less* groups

   mean a simpler expression.

Notes:

Part 3: general-purpose derivation (5)

Method #2: Karnaugh map

Example

Natural sequence			Gray code sequence		
$\langle 0,0,0,0 \rangle$	$\mapsto$	$0_{(10)}$	$\langle 0,0,0,0 \rangle$	$\mapsto$	$0_{(10)}$
$\langle 1,0,0,0 \rangle$	$\mapsto$	$1_{(10)}$	$\langle 1,0,0,0 \rangle$	$\mapsto$	$1_{(10)}$
$\langle 0,1,0,0 \rangle$	$\mapsto$	$2_{(10)}$	$\langle 1,1,0,0 \rangle$	$\mapsto$	$3_{(10)}$
$\langle 1,1,0,0 \rangle$	$\mapsto$	$3_{(10)}$	$\langle 0,1,0,0 \rangle$	$\mapsto$	$2_{(10)}$
$\langle 0,0,1,0 \rangle$	$\mapsto$	$4_{(10)}$	$\langle 0,1,1,0 \rangle$	$\mapsto$	$6_{(10)}$
$\langle 1,0,1,0 \rangle$	$\mapsto$	$5_{(10)}$	$\langle 0,0,1,0 \rangle$	$\mapsto$	$4_{(10)}$
$\langle 0,1,1,0 \rangle$	$\mapsto$	$6_{(10)}$	$\langle 1,0,1,0 \rangle$	$\mapsto$	$5_{(10)}$
$\langle 1,1,1,0 \rangle$	$\mapsto$	$7_{(10)}$	$\langle 1,1,1,0 \rangle$	$\mapsto$	$7_{(10)}$
	$\vdots$			$\vdots$	

Notes:

Example

Consider an example 4-input, 1-output function:

w	x	y	z	r
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Notes:

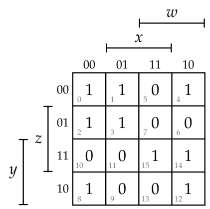
- The first two steps are simple: drawing a grid of an appropriate size then filling it with entries from the truth table are both trivial, even though they still demand some care.
- Forming the groups is a little harder, in the sense that their "quality" (i.e., number and size) dictates how optimised the resulting expression will be. This example has some non-intuitive cases: the blue group might not be obvious for example, *but* is within the rules (although it is sort of inside-out, it is rectangular and a power-of-two in size).
- The last step is the hardest: we need to translate each group into a term that covers it. Put another way, we want a term that specifies just the cells in that group. In a sense, the more variables that exist within a term place more restrictions on which cells we specify; this highlights the fact that larger groups therefor contain fewer variables, and are therefore simpler. This example has three groups:
  - The red group spans columns 0 and 1 and rows 0 and 1; provided $w = 0$ and $y = 0$ we specify *just* those cells, so the expression is $\neg w \wedge \neg y$. That is, $w = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $w = 1$) and $y = 0$ restricts us to rows 0 and 1 (rows 2 and 3 have $y = 1$). Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and y.
  - The green group spans columns 2 and 3 in row 2; provided $w = 1$, $y = 1$ and $z = 1$ we specify *just* those cells, so the expression is $w \wedge y \wedge z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ and $z = 1$ restricts us to row 2 (rows 0, 1 and 3 have at least one of $y = 0$ or $z = 0$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 0 and 3 and rows 0 and 3; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. That is, $x = 0$ restricts us to columns 0 and 3 (columns 1 and 2 have $x = 1$) and $z = 0$ restricts us to rows 0 and 3 (rows 1 and 2 have $z = 1$). Note that the values of w and y don't matter: cells in the group hold the value 1 regardless of w and y.

Part 3: general-purpose derivation (6)
Method #2: Karnaugh map

Example

Consider an example 4-input, 1-output function:

w	x	y	z	r
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Notes:

- The first two steps are simple: drawing a grid of an appropriate size then filling it with entries from the truth table are both trivial, even though they still demand some care.
- Forming the groups is a little harder, in the sense that their "quality" (i.e., number and size) dictates how optimised the resulting expression will be. This example has some non-intuitive cases: the blue group might not be obvious for example, *but* is within the rules (although it is sort of inside-out, it is rectangular and a power-of-two in size).
- The last step is the hardest: we need to translate each group into a term that covers it. Put another way, we want a term that specifies just the cells in that group. In a sense, the more variables that exist within a term place more restrictions on which cells we specify; this highlights the fact that larger groups therefor contain fewer variables, and are therefore simpler. This example has three groups:
  - The red group spans columns 0 and 1 and rows 0 and 1; provided $w = 0$ and $y = 0$ we specify *just* those cells, so the expression is $\neg w \wedge \neg y$. That is, $w = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $w = 1$) and $y = 0$ restricts us to rows 0 and 1 (rows 2 and 3 have $y = 1$). Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and y.
  - The green group spans columns 2 and 3 in row 2; provided $w = 1$, $y = 1$ and $z = 1$ we specify *just* those cells, so the expression is $w \wedge y \wedge z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ and $z = 1$ restricts us to row 2 (rows 0, 1 and 3 have at least one of $y = 0$ or $z = 0$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 0 and 3 and rows 0 and 3; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. That is, $x = 0$ restricts us to columns 0 and 3 (columns 1 and 2 have $x = 1$) and $z = 0$ restricts us to rows 0 and 3 (rows 1 and 2 have $z = 1$). Note that the values of w and y don't matter: cells in the group hold the value 1 regardless of w and y.

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

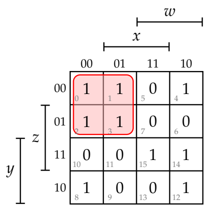
Part 3: general-purpose derivation (6)

Method #2: Karnaugh map

Example

Consider an example 4-input, 1-output function:

w	x	y	z	r
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Each group translates into one term of the SoP form expression

$$r = (\neg w \wedge \neg y)$$

Notes:

- The first two steps are simple: drawing a grid of an appropriate size then filling it with entries from the truth table are both trivial, even though they still demand some care.
- Forming the groups is a little harder, in the sense that their "quality" (i.e., number and size) dictates how optimised the resulting expression will be. This example has some non-intuitive cases: the blue group might not be obvious for example, *but* is within the rules (although it is sort of inside-out, it is rectangular and a power-of-two in size).
- The last step is the hardest: we need to translate each group into a term that covers it. Put another way, we want a term that specifies just the cells in that group. In a sense, the more variables that exist within a term place more restrictions on which cells we specify; this highlights the fact that larger groups therefor contain fewer variables, and are therefore simpler. This example has three groups:
  - The red group spans columns 0 and 1 and rows 0 and 1; provided $w = 0$ and $y = 0$ we specify *just* those cells, so the expression is $\neg w \wedge \neg y$. That is, $w = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $w = 1$) and $y = 0$ restricts us to rows 0 and 1 (rows 2 and 3 have $y = 1$). Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and y.
  - The green group spans columns 2 and 3 in row 2; provided $w = 1$, $y = 1$ and $z = 1$ we specify *just* those cells, so the expression is $w \wedge y \wedge z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ and $z = 1$ restricts us to row 2 (rows 0, 1 and 3 have at least one of $y = 0$ or $z = 0$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 0 and 3 and rows 0 and 3; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. That is, $x = 0$ restricts us to columns 0 and 3 (columns 1 and 2 have $x = 1$) and $z = 0$ restricts us to rows 0 and 3 (rows 1 and 2 have $z = 1$). Note that the values of w and y don't matter: cells in the group hold the value 1 regardless of w and y.

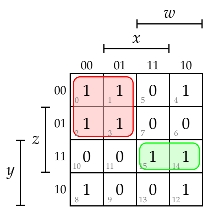
Part 3: general-purpose derivation (6)

Method #2: Karnaugh map

Example

Consider an example 4-input, 1-output function:

w	x	y	z	r
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Each group translates into one term of the SoP form expression

$$\begin{array}{rcl} r & = & (\neg w \wedge \neg y) \vee \\ & & (w \wedge y \wedge z) \end{array}$$

Notes:

- The first two steps are simple: drawing a grid of an appropriate size then filling it with entries from the truth table are both trivial, even though they still demand some care.
- Forming the groups is a little harder, in the sense that their "quality" (i.e., number and size) dictates how optimised the resulting expression will be. This example has some non-intuitive cases: the blue group might not be obvious for example, *but* is within the rules (although it is sort of inside-out, it is rectangular and a power-of-two in size).
- The last step is the hardest: we need to translate each group into a term that covers it. Put another way, we want a term that specifies just the cells in that group. In a sense, the more variables that exist within a term place more restrictions on which cells we specify; this highlights the fact that larger groups therefor contain fewer variables, and are therefore simpler. This example has three groups:
  - The red group spans columns 0 and 1 and rows 0 and 1; provided $w = 0$ and $y = 0$ we specify *just* those cells, so the expression is $\neg w \wedge \neg y$. That is, $w = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $w = 1$) and $y = 0$ restricts us to rows 0 and 1 (rows 2 and 3 have $y = 1$). Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and y.
  - The green group spans columns 2 and 3 in row 2; provided $w = 1$, $y = 1$ and $z = 1$ we specify *just* those cells, so the expression is $w \wedge y \wedge z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ and $z = 1$ restricts us to row 2 (rows 0, 1 and 3 have at least one of $y = 0$ or $z = 0$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 0 and 3 and rows 0 and 3; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. That is, $x = 0$ restricts us to columns 0 and 3 (columns 1 and 2 have $x = 1$) and $z = 0$ restricts us to rows 0 and 3 (rows 1 and 2 have $z = 1$). Note that the values of w and y don't matter: cells in the group hold the value 1 regardless of w and y.

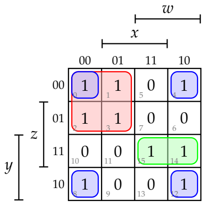
Part 3: general-purpose derivation (6)

Method #2: Karnaugh map

Example

Consider an example 4-input, 1-output function:

w	x	y	z	r
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



Each group translates into one term of the SoP form expression

$$
\begin{array}{rcl}
r & = & (\neg w \wedge \neg y) \vee \\
 & & (w \wedge y \wedge z) \vee \\
 & & (\neg x \wedge \neg z)
\end{array}
$$

Notes:

- The first two steps are simple: drawing a grid of an appropriate size then filling it with entries from the truth table are both trivial, even though they still demand some care.
- Forming the groups is a little harder, in the sense that their "quality" (i.e., number and size) dictates how optimised the resulting expression will be. This example has some non-intuitive cases: the blue group might not be obvious for example, *but* is within the rules (although it is sort of inside-out, it is rectangular and a power-of-two in size).
- The last step is the hardest: we need to translate each group into a term that covers it. Put another way, we want a term that specifies just the cells in that group. In a sense, the more variables that exist within a term place more restrictions on which cells we specify; this highlights the fact that larger groups therefor contain fewer variables, and are therefore simpler. This example has three groups:
  - The red group spans columns 0 and 1 and rows 0 and 1; provided $w = 0$ and $y = 0$ we specify *just* those cells, so the expression is $\neg w \wedge \neg y$. That is, $w = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $w = 1$) and $y = 0$ restricts us to rows 0 and 1 (rows 2 and 3 have $y = 1$). Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and y.
  - The green group spans columns 2 and 3 in row 2; provided $w = 1$, $y = 1$ and $z = 1$ we specify *just* those cells, so the expression is $w \wedge y \wedge z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ and $z = 1$ restricts us to row 2 (rows 0, 1 and 3 have at least one of $y = 0$ or $z = 0$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 0 and 3 and rows 0 and 3; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. That is, $x = 0$ restricts us to columns 0 and 3 (columns 1 and 2 have $x = 1$) and $z = 0$ restricts us to rows 0 and 3 (rows 1 and 2 have $z = 1$). Note that the values of w and y don't matter: cells in the group hold the value 1 regardless of w and y.

Part 3: general-purpose derivation (7)

Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	**?**
1	1	0	1
1	1	1	**?**

Notes:

- Note the differing shape of this grid versus the previous example: since there are $n = 3$ variables, we set $p = 2$ and $q = 4$ such that the grid contains $2 \cdot 4 = 2^3 = 8$ cells.
- Adopting the same approach as the previous example, by not ignoring the don't care entries (i.e., assuming they are 0) we have two groups. However, opting to treat one of them as a 1 (which is fine: by definition we don't care what the output is) we only have one:
  - The red group spans column 1 and rows 0 and 1; provided $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg x \wedge y$. That is, $x = 0$ and $y = 1$ restricts us to column 1 (columns 0, 2 and 3 have at least one of $x = 1$ or $y = 0$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The green group spans columns 1 and 2, in row 0; provided $y = 1$ and $z = 0$ we specify *just* those cells, so the expression is $y \wedge \neg z$. That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 1 and 2 and rows 0 and 1; provided $y = 1$ we specify *just* those cells, so the expression is y That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) which is all we need because the group spans *all* rows. Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	?
1	1	0	1
1	1	1	?

z \ xy	00	01	11	10
0	0	1	1	0
1	0	1	?	?

z \ xy	00	01	11	10
0	0	1	1	0
1	0	1	?	?

Notes:

- Note the differing shape of this grid versus the previous example: since there are $n = 3$ variables, we set $p = 2$ and $q = 4$ such that the grid contains $2 \cdot 4 = 2^3 = 8$ cells.
- Adopting the same approach as the previous example, by not ignoring the don't care entries (i.e., assuming they are 0) we have two groups. However, opting to treat one of them as a 1 (which is fine: by definition we don't care what the output is) we only have one:
  - The red group spans column 1 and rows 0 and 1; provided $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg x \wedge y$. That is, $x = 0$ and $y = 1$ restricts us to column 1 (columns 0, 2 and 3 have at least one of $x = 1$ or $y = 0$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The green group spans columns 1 and 2, in row 0; provided $y = 1$ and $z = 0$ we specify *just* those cells, so the expression is $y \wedge \neg z$. That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 1 and 2 and rows 0 and 1; provided $y = 1$ we specify *just* those cells, so the expression is y That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) which is all we need because the group spans *all* rows. Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Part 3: general-purpose derivation (7)
Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	?
1	1	0	1
1	1	1	?

z \ xy	00	01	11	10
0	0	1	1	0
1	0	1	?	?

z \ xy	00	01	11	10
0	0	1	1	0
1	0	1	?	?

Each group translates into one term of the SoP form expressions

$$r = (\neg x \wedge y)$$

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

Notes:

- Note the differing shape of this grid versus the previous example: since there are $n = 3$ variables, we set $p = 2$ and $q = 4$ such that the grid contains $2 \cdot 4 = 2^3 = 8$ cells.
- Adopting the same approach as the previous example, by not ignoring the don't care entries (i.e., assuming they are 0) we have two groups. However, opting to treat one of them as a 1 (which is fine: by definition we don't care what the output is) we only have one:
  - The red group spans column 1 and rows 0 and 1; provided $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg x \wedge y$. That is, $x = 0$ and $y = 1$ restricts us to column 1 (columns 0, 2 and 3 have at least one of $x = 1$ or $y = 0$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The green group spans columns 1 and 2, in row 0; provided $y = 1$ and $z = 0$ we specify *just* those cells, so the expression is $y \wedge \neg z$. That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 1 and 2 and rows 0 and 1; provided $y = 1$ we specify *just* those cells, so the expression is y That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) which is all we need because the group spans *all* rows. Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

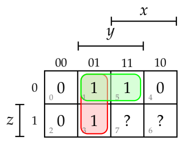
Part 3: general-purpose derivation (7)

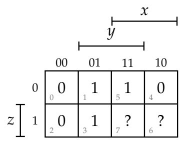
Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	**?**
1	1	0	1
1	1	1	**?**





Each group translates into one term of the SoP form expressions

$$r = (\neg x \wedge y) \vee (y \wedge \neg z)$$

Notes:

- Note the differing shape of this grid versus the previous example: since there are $n = 3$ variables, we set $p = 2$ and $q = 4$ such that the grid contains $2 \cdot 4 = 2^3 = 8$ cells.
- Adopting the same approach as the previous example, by not ignoring the don't care entries (i.e., assuming they are 0) we have two groups. However, opting to treat one of them as a 1 (which is fine: by definition we don't care what the output is) we only have one:
  - The red group spans column 1 and rows 0 and 1; provided $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg x \wedge y$. That is, $x = 0$ and $y = 1$ restricts us to column 1 (columns 0, 2 and 3 have at least one of $x = 1$ or $y = 0$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The green group spans columns 1 and 2, in row 0; provided $y = 1$ and $z = 0$ we specify *just* those cells, so the expression is $y \wedge \neg z$. That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 1 and 2 and rows 0 and 1; provided $y = 1$ we specify *just* those cells, so the expression is y That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) which is all we need because the group spans *all* rows. Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

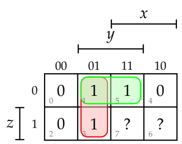
Part 3: general-purpose derivation (7)

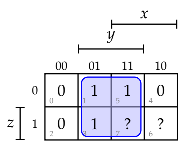
Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	**?**
1	1	0	1
1	1	1	**?**





Each group translates into one term of the SoP form expressions

$$r = (\neg x \wedge y) \vee (y \wedge \neg z) \qquad r = y$$

where effective use of don't care states yields a clear improvement!

Notes:

- Note the differing shape of this grid versus the previous example: since there are $n = 3$ variables, we set $p = 2$ and $q = 4$ such that the grid contains $2 \cdot 4 = 2^3 = 8$ cells.
- Adopting the same approach as the previous example, by not ignoring the don't care entries (i.e., assuming they are 0) we have two groups. However, opting to treat one of them as a 1 (which is fine: by definition we don't care what the output is) we only have one:
  - The red group spans column 1 and rows 0 and 1; provided $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg x \wedge y$. That is, $x = 0$ and $y = 1$ restricts us to column 1 (columns 0, 2 and 3 have at least one of $x = 1$ or $y = 0$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The green group spans columns 1 and 2, in row 0; provided $y = 1$ and $z = 0$ we specify *just* those cells, so the expression is $y \wedge \neg z$. That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The blue group spans columns 1 and 2 and rows 0 and 1; provided $y = 1$ we specify *just* those cells, so the expression is y That is, $y = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $y = 0$) which is all we need because the group spans *all* rows. Note that the values of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Part 3: general-purpose derivation (8)

Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Notes:

- – The red group spans columns 2 and 3, in row 1; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. $x = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $x = 0$) and $z = 1$ restricts us to row 1 (row 0 has $z = 0$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.
  – The green group spans column 3 and rows 0 and 1; provided $x = 1$ and $y = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y$. That is, $x = 1$ and $y = 0$ restricts us to column 3 (columns 0, 1 and 2 have at least one of $x = 0$ or $y = 1$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 0 and 1, in row 0; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. $x = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $x = 1$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.

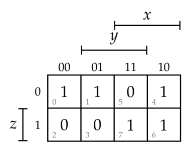
Part 3: general-purpose derivation (8)

Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



Notes:

- – The red group spans columns 2 and 3, in row 1; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. $x = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $x = 0$) and $z = 1$ restricts us to row 1 (row 0 has $z = 0$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.
  – The green group spans column 3 and rows 0 and 1; provided $x = 1$ and $y = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y$. That is, $x = 1$ and $y = 0$ restricts us to column 3 (columns 0, 1 and 2 have at least one of $x = 0$ or $y = 1$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 0 and 1, in row 0; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. $x = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $x = 1$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.

Part 3: general-purpose derivation (8)

Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

z \ xy	00	01	11	10
0	1 (0)	1 (1)	0 (5)	1 (4)
1	0 (2)	0 (3)	1 (7)	1 (6)

Each group translates into one term of the SoP form expression

$$r = (x \wedge z)$$

Notes:

- – The red group spans columns 2 and 3, in row 1; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. $x = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $x = 0$) and $z = 1$ restricts us to row 1 (row 0 has $z = 0$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.
  – The green group spans column 3 and rows 0 and 1; provided $x = 1$ and $y = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y$. That is, $x = 1$ and $y = 0$ restricts us to column 3 (columns 0, 1 and 2 have at least one of $x = 0$ or $y = 1$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 0 and 1, in row 0; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. $x = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $x = 1$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.

Part 3: general-purpose derivation (8)

Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

z \ xy	00	01	11	10
0	1 (0)	1 (1)	0 (5)	1 (4)
1	0 (2)	0 (3)	1 (7)	1 (6)

Each group translates into one term of the SoP form expression

$$\begin{array}{rcl} r & = & (x \wedge z) \vee \\ & & (x \wedge \neg y) \end{array}$$

Notes:

- – The red group spans columns 2 and 3, in row 1; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. $x = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $x = 0$) and $z = 1$ restricts us to row 1 (row 0 has $z = 0$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.
  – The green group spans column 3 and rows 0 and 1; provided $x = 1$ and $y = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y$. That is, $x = 1$ and $y = 0$ restricts us to column 3 (columns 0, 1 and 2 have at least one of $x = 0$ or $y = 1$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 0 and 1, in row 0; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. $x = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $x = 1$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.

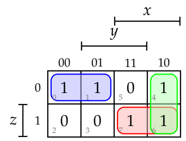
Part 3: general-purpose derivation (8)

Method #2: Karnaugh map

Example

Consider an example 3-input, 1-output function:

x	y	z	r
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1



Each group translates into one term of the SoP form expression

$$
\begin{array}{rcl}
r & = & (\; x \wedge z \;) \vee \\
  &   & (\; x \wedge \neg y \;) \vee \\
  &   & (\; \neg x \wedge \neg z \;)
\end{array}
$$

Notes:

-
  - The red group spans columns 2 and 3, in row 1; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. $x = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $x = 0$) and $z = 1$ restricts us to row 1 (row 0 has $z = 0$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.
  - The green group spans column 3 and rows 0 and 1; provided $x = 1$ and $y = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y$. That is, $x = 1$ and $y = 0$ restricts us to column 3 (columns 0, 1 and 2 have at least one of $x = 0$ or $y = 1$) which is all we need because the group spans *all* rows. Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The blue group spans columns 0 and 1, in row 0; provided $x = 0$ and $z = 0$ we specify *just* those cells, so the expression is $\neg x \wedge \neg z$. $x = 0$ restricts us to columns 0 and 1 (columns 2 and 3 have $x = 1$) and $z = 0$ restricts us to row 0 (row 1 has $z = 1$). Note that the value of y doesn't matter: cells in the group hold the value 1 regardless of y.

Part 3: general-purpose derivation (9)

Method #2: Karnaugh map

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	**?**	**?**
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	**?**	**?**
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	**?**	**?**
1	1	0	0	**?**	**?**
1	1	0	1	**?**	**?**
1	1	1	0	**?**	**?**
1	1	1	1	**?**	**?**

Notes:

-
  - The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  - The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  - The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
  - The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  - The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  - The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Part 3: general-purpose derivation (9)

Method #2: Karnaugh map

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	?	?
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	?	?
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	?	?
1	1	0	0	?	?
1	1	0	1	?	?
1	1	1	0	?	?
1	1	1	1	?	?

r_1	00	01	11	10
00	0	0	?	1
01	0	1	?	0
11	?	?	?	?
10	1	0	?	0

r_0	00	01	11	10
00	0	1	?	0
01	1	0	?	0
11	?	?	?	?
10	0	0	?	1

Notes:

- The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
- The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
- The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
- The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
- The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
- The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Part 3: general-purpose derivation (9)

Method #2: Karnaugh map

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	?	?
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	?	?
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	?	?
1	1	0	0	?	?
1	1	0	1	?	?
1	1	1	0	?	?
1	1	1	1	?	?

r_1	00	01	11	10
00	0	0	?	1
01	0	1	?	0
11	?	?	?	?
10	1	0	?	0

r_0	00	01	11	10
00	0	1	?	0
01	1	0	?	0
11	?	?	?	?
10	0	0	?	1

Each group translates into one term of the SoP form expressions

$$r_1 = (\; w \; \wedge \; \neg y \; \wedge \; \neg z \;)$$

Notes:

- The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
- The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
- The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
- The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
- The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
- The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Part 3: general-purpose derivation (9)

Method #2: Karnaugh map

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	**?**	**?**
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	**?**	**?**
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	**?**	**?**
1	1	0	0	**?**	**?**
1	1	0	1	**?**	**?**
1	1	1	0	**?**	**?**
1	1	1	1	**?**	**?**

r_1	00	01	11	10
00	0	0	?	1
01	0	1	?	0
11	?	?	?	?
10	1	0	?	0

r_0	00	01	11	10
00	0	1	?	0
01	1	0	?	0
11	?	?	?	?
10	0	0	?	1

Each group translates into one term of the SoP form expressions

$$
\begin{array}{rcl}
r_1 & = & (\; w \wedge \neg y \wedge \neg z \;) \vee \\
 & & (\; y \wedge \neg w \wedge \neg x \;)
\end{array}
$$

Notes:

- – The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  – The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
  – The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Part 3: general-purpose derivation (9)

Method #2: Karnaugh map

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	**?**	**?**
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	**?**	**?**
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	**?**	**?**
1	1	0	0	**?**	**?**
1	1	0	1	**?**	**?**
1	1	1	0	**?**	**?**
1	1	1	1	**?**	**?**

r_1	00	01	11	10
00	0	0	?	1
01	0	1	?	0
11	?	?	?	?
10	1	0	?	0

r_0	00	01	11	10
00	0	1	?	0
01	1	0	?	0
11	?	?	?	?
10	0	0	?	1

Each group translates into one term of the SoP form expressions

$$
\begin{array}{rcl}
r_1 & = & (\; w \wedge \neg y \wedge \neg z \;) \vee \\
 & & (\; y \wedge \neg w \wedge \neg x \;) \vee \\
 & & (\; x \wedge z \;)
\end{array}
$$

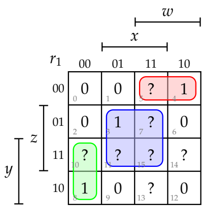
Notes:

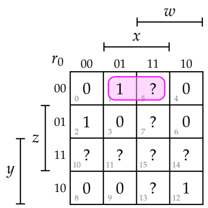
- – The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  – The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
  – The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	?	?
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	?	?
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	?	?
1	1	0	0	?	?
1	1	0	1	?	?
1	1	1	0	?	?
1	1	1	1	?	?





Each group translates into one term of the SoP form expressions

$$r_1 = (w \wedge \neg y \wedge \neg z) \vee (y \wedge \neg w \wedge \neg x) \vee (x \wedge z)$$

$$r_0 = (x \wedge \neg y \wedge \neg z)$$

Notes:

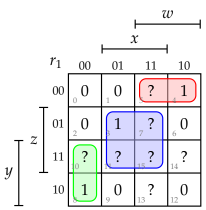
- – The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  – The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
  – The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

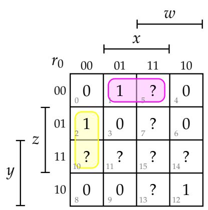
Part 3: general-purpose derivation (9)
Method #2: Karnaugh map

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	?	?
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	?	?
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	?	?
1	1	0	0	?	?
1	1	0	1	?	?
1	1	1	0	?	?
1	1	1	1	?	?





Each group translates into one term of the SoP form expressions

$$r_1 = (w \wedge \neg y \wedge \neg z) \vee (y \wedge \neg w \wedge \neg x) \vee (x \wedge z)$$

$$r_0 = (x \wedge \neg y \wedge \neg z) \vee (z \wedge \neg w \wedge \neg x)$$

© Daniel Page ⟨csdsp@bristol.ac.uk⟩
Computer Architecture
University of BRISTOL
git # c144985a @ 2024-10-14

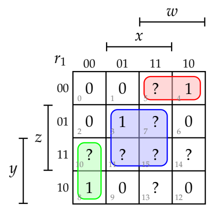
Notes:

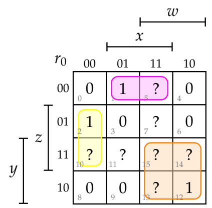
- – The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  – The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
  – The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Example

Consider an example 4-input, 2-output function:

w	x	y	z	r_1	r_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	**?**	**?**
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	**?**	**?**
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	1
1	0	1	1	**?**	**?**
1	1	0	0	**?**	**?**
1	1	0	1	**?**	**?**
1	1	1	0	**?**	**?**
1	1	1	1	**?**	**?**





Each group translates into one term of the SoP form expressions

$$r_1 = (w \wedge \neg y \wedge \neg z) \vee (y \wedge \neg w \wedge \neg x) \vee (x \wedge z)$$

$$r_0 = (x \wedge \neg y \wedge \neg z) \vee (z \wedge \neg w \wedge \neg x) \vee (w \wedge y)$$

Notes:

- – The red group spans columns 2 and 3 in row 0; provided $w = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $w \wedge \neg y \wedge \neg z$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$). Note that the value of x doesn't matter: cells in the group hold the value 1 regardless of x.
  – The green group spans column 1 and rows 2 and 3; provided $w = 0$, $x = 0$ and $y = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge y$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of z doesn't matter: cells in the group hold the value 1 regardless of z.
  – The blue group spans columns 1 and 2 and rows 1 and 2; provided $x = 1$ and $z = 1$ we specify *just* those cells, so the expression is $x \wedge z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $z = 1$ restricts us to rows 1 and 2 (rows 0 and 3 have $z = 0$). Note that the value of w and y don't matter: cells in the group hold the value 1 regardless of w and y.
  – The magenta group spans columns 1 and 2 in row 0; provided $x = 1$, $y = 0$ and $z = 0$ we specify *just* those cells, so the expression is $x \wedge \neg y \wedge \neg z$. That is, $x = 1$ restricts us to columns 1 and 2 (columns 0 and 3 have $x = 0$) and $y = 0$ and $z = 0$ restricts us to row 0 (rows 1, 2 and 3 have at least one of $y = 1$ or $z = 1$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The yellow group spans column 0 and rows 1 and 2; provided $w = 0$, $x = 0$ and $z = 1$ we specify *just* those cells, so the expression is $\neg w \wedge \neg x \wedge z$. That is, $w = 0$ and $x = 0$ restricts us to column 0 (columns 1, 2 and 3 have at least one of $w = 1$ or $x = 1$) and $y = 1$ restricts us to rows 1 and 2 (rows 0 and 1 have $y = 0$) Note that the value of w doesn't matter: cells in the group hold the value 1 regardless of w.
  – The orange group spans columns 2 and 3 and rows 2 and 3; provided $w = 1$ and $y = 1$ we specify *just* those cells, so the expression is $w \wedge y$. That is, $w = 1$ restricts us to columns 2 and 3 (columns 0 and 1 have $w = 0$) and $y = 1$ restricts us to rows 2 and 3 (rows 0 and 1 have $y = 0$). Note that the value of x and z don't matter: cells in the group hold the value 1 regardless of x and z.

Conclusions

- Take away points:
  1. There are a *huge* number of challenges, even with (relatively) simple problems, e.g.,
     - how do we describe what the design should do?
     - how do we structure the design?
     - what sort of standard cell library do we use?
     - do we aim for the fewest gates?
     - do we aim for shortest critical path?
     - how do we cope with propagation delay and fan-out?
     - ...
  2. The three themes we've covered, i.e.,
     - high-level design patterns,
     - low-level, mechanical derivation and optimisation of Boolean expressions,
     - building-block components,

     allows us to address such challenges: in combination, they support development of effective (combinatorial) design and implementation.
  3. In many cases, use of appropriate **Electronic Design Automation (EDA)** tools can provide (semi-)automatic solutions.

Notes:

Additional Reading

- *Wikipedia: Combinational logic.* URL: https://en.wikipedia.org/wiki/Combinational_logic.
- D. Page. "Chapter 2: Basics of digital logic". In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009.
- W. Stallings. "Chapter 11: Digital logic". In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013.
- A.S. Tanenbaum and T. Austin. "Section 3.2.2: Combinatorial circuits". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012.

Notes:

References

[1] *Wikipedia: Combinational logic.* URL: https://en.wikipedia.org/wiki/Combinational_logic (see p. 125).

[2] D. Page. "Chapter 2: Basics of digital logic". In: *A Practical Introduction to Computer Architecture*. 1st ed. Springer, 2009 (see p. 125).

[3] W. Stallings. "Chapter 11: Digital logic". In: *Computer Organisation and Architecture*. 9th ed. Prentice Hall, 2013 (see p. 125).

[4] A.S. Tanenbaum and T. Austin. "Section 3.2.2: Combinatorial circuits". In: *Structured Computer Organisation*. 6th ed. Prentice Hall, 2012 (see p. 125).

Notes: