Boolean Algebras

Carmi Merimovich

The Academic College of Tel-Aviv

January 29, 2025

Aim

- Introducing boolean algebras
- Present examples for their usage

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^{-} \rangle$, where

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^{-} \rangle$, where

- $\mathbb{B}$ is a set,

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^{-} \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumably, not equal),

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^{-} \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumably, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, \bar{} \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumbly, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $\bar{}$ is a unary operator (1-ary functions),

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^- \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumbly, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $^-$ is a unary operator (1-ary functions),

is a boolean algebra if the following hold for each x, y, z,

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^- \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumbly, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $^-$ is a unary operator (1-ary functions),

is a boolean algebra if the following hold for each x, y, z,

Neutrality: $x + 0 = x \quad x \cdot 1 = x$

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, \bar{\ } \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumbly, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $\bar{\ }$ is a unary operator (1-ary functions),

is a boolean algebra if the following hold for each x, y, z,

Neutrality:	$x + 0 = x$	$x \cdot 1 = x$
Comutativity:	$x + y = y + x$	$x \cdot y = y \cdot x$

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, \bar{\ } \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumbly, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $\bar{\ }$ is a unary operator (1-ary functions),

is a boolean algebra if the following hold for each x, y, z,

Neutrality:	$x + 0 = x$	$x \cdot 1 = x$
Comutativity:	$x + y = y + x$	$x \cdot y = y \cdot x$
Associativity:	$(x + y) + z = x + (y + z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, \bar{} \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumably, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $\bar{}$ is a unary operator (1-ary functions),

is a boolean algebra if the following hold for each x, y, z,

Neutrality:	$x + 0 = x$	$x \cdot 1 = x$
Comutativity:	$x + y = y + x$	$x \cdot y = y \cdot x$
Associativity:	$(x + y) + z = x + (y + z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
Distributivity:	$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + (y \cdot z) = (x + y) \cdot (x + z)$

Boolean algebra

Definition

The structure $\langle \mathbb{B}, 0, 1, +, \cdot, ^- \rangle$, where

- $\mathbb{B}$ is a set,
- 0 and 1 are constants (presumably, not equal),
- $+$ and $\cdot$ are binary operators (i.e., 2-ary functions),
- $^-$ is a unary operator (1-ary functions),

is a boolean algebra if the following hold for each x, y, z,

Neutrality:	$x + 0 = x$	$x \cdot 1 = x$
Comutativity:	$x + y = y + x$	$x \cdot y = y \cdot x$
Associativity:	$(x + y) + z = x + (y + z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
Distributivity:	$x \cdot (y + z) = x \cdot y + x \cdot z$	$x + (y \cdot z) = (x + y) \cdot (x + z)$
Complement:	$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$

unique?

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$\square$

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$$y = y \cdot 1 =$$

□

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) =
\end{aligned}
$$

$\square$

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) = \\
=& y \cdot x + y \cdot z =
\end{aligned}
$$

□

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ *and* $x + z = 1$ *then* $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) = \\
=& y \cdot x + y \cdot z = \\
=& 0 + y \cdot z =
\end{aligned}
$$

□

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) = \\
=& y \cdot x + y \cdot z = \\
=& 0 + y \cdot z = \\
=& x \cdot z + y \cdot z =
\end{aligned}
$$

□

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) = \\
=& y \cdot x + y \cdot z = \\
=& 0 + y \cdot z = \\
=& x \cdot z + y \cdot z = \\
=& z \cdot (x + y) =
\end{aligned}
$$

□

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ *and* $x + z = 1$ *then* $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) = \\
=& y \cdot x + y \cdot z = \\
=& 0 + y \cdot z = \\
=& x \cdot z + y \cdot z = \\
=& z \cdot (x + y) = \\
=& z \cdot 1 =
\end{aligned}
$$

□

Lemma (Uniqueness of complement)

If $xy = 0$, $x + y = 1$, $xz = 0$ and $x + z = 1$ then $y = z$.

Proof.

$$
\begin{aligned}
y =& y \cdot 1 = \\
=& y \cdot (x + z) = \\
=& y \cdot x + y \cdot z = \\
=& 0 + y \cdot z = \\
=& x \cdot z + y \cdot z = \\
=& z \cdot (x + y) = \\
=& z \cdot 1 = \\
=& z.
\end{aligned}
$$

□

Minimal Boolean Algebra

Lemma

Assume $\mathbb{B}$ is a boolean algebra. Then the set of constants $\{0, 1\}$ is closed under the operators $+$, $\cdot$ and $^{-}$. (i.e., $\{0, 1\}$ is a boolean subalgebra of $\mathbb{B}$.)

Proof.

Minimal Boolean Algebra

Lemma

Assume $\mathbb{B}$ is a boolean algebra. Then the set of constants $\{0,1\}$ is closed under the operators $+$, $\cdot$ and $^{-}$. (i.e., $\{0,1\}$ is a boolean subalgebra of $\mathbb{B}$.)

Proof.

1. $^{-}$: By the relevant neutralities of 0 and 1 we have $0+1=1$ and $0\cdot 1=0$. Thus $\bar{0}=1$ and $\bar{1}=0$.

Minimal Boolean Algebra

Boolean Algebras

©C.M.

Lemma

Assume $\mathbb{B}$ is a boolean algebra. Then the set of constants $\{0,1\}$ is closed under the operators $+$, $\cdot$ and $^-$. (i.e., $\{0,1\}$ is a boolean subalgebra of $\mathbb{B}$.)

Proof.

1. $^-$: By the relevant neutralities of 0 and 1 we have $0+1=1$ and $0\cdot 1=0$. Thus $\bar{0}=1$ and $\bar{1}=0$.
2. $\cdot$: By the neutrality of 1 we get $1\cdot 0=0\cdot 1=0$ and $1\cdot 1=1$. For $0\cdot 0$ we work as follows. $0\cdot 0=0\cdot 0+0=0\cdot 0+(0\cdot 1)=0\cdot(0+1)=0\cdot 1=0.$

Minimal Boolean Algebra

Lemma

Assume $\mathbb{B}$ is a boolean algebra. Then the set of constants $\{0,1\}$ is closed under the operators $+$, $\cdot$ and $^-$. (i.e., $\{0,1\}$ is a boolean subalgebra of $\mathbb{B}$.)

Proof.

1. $^-$: By the relevant neutralities of 0 and 1 we have $0+1=1$ and $0\cdot 1=0$. Thus $\bar{0}=1$ and $\bar{1}=0$.
2. $\cdot$: By the neutrality of 1 we get $1\cdot 0=0\cdot 1=0$ and $1\cdot 1=1$. For $0\cdot 0$ we work as follows. $0\cdot 0=0\cdot 0+0=0\cdot 0+(0\cdot 1)=0\cdot(0+1)=0\cdot 1=0.$
3. $+$: By the neutrality of 0 we get $1+0=0+1=1$ and $0+0=0$. For $1+1$ we work as follows. $1+1=(1+1)\cdot 1=(1+1)\cdot(1+0)=1+(1\cdot 0)=1+0=1.$

□

Truth table form of the lemma

x	$\bar{x}$
0	1
1	0

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

- **If** there is a boolean algebra then
  - its 0 and 1 follow the above tables
- No boolean algebra has been spotted as of yet!

The 2-Valued boolean algebra

Let us take the truth tables above as a **definition**.

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0			
0	0	1	0			
0	1	0	0			
0	1	1	0			
1	0	0	0			
1	0	1	0			
1	1	0	1			
1	1	1	1			

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0	0		
0	0	1	0	0		
0	1	0	0	0		
0	1	1	0	0		
1	0	0	0	0		
1	0	1	0	0		
1	1	0	1	0		
1	1	1	1	1		

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0	0	0	
0	0	1	0	0	0	
0	1	0	0	0	0	
0	1	1	0	0	1	
1	0	0	0	0	0	
1	0	1	0	0	0	
1	1	0	1	0	0	
1	1	1	1	1	1	

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Associativity of $\cdot$

x	y	z	$x \cdot y$	$(x \cdot y) \cdot z$	$y \cdot z$	$x \cdot (y \cdot z)$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0	0			
0	0	1	0			
0	1	0	1			
0	1	1	1			
1	0	0	1			
1	0	1	1			
1	1	0	1			
1	1	1	1			

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0	0	0		
0	0	1	0	1		
0	1	0	1	1		
0	1	1	1	1		
1	0	0	1	1		
1	0	1	1	1		
1	1	0	1	1		
1	1	1	1	1		

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0	0	0	0	
0	0	1	0	1	1	
0	1	0	1	1	1	
0	1	1	1	1	1	
1	0	0	1	1	0	
1	0	1	1	1	1	
1	1	0	1	1	1	
1	1	1	1	1	1	

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Associativity of $+$

x	y	z	$x+y$	$(x+y)+z$	$y+z$	$x+(y+z)$
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	1	1	0	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x \cdot (y+z)$	$x \cdot y$	$x \cdot z$	$x \cdot y + x \cdot z$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x \cdot (y+z)$	$x \cdot y$	$x \cdot z$	$x \cdot y + x \cdot z$
0	0	0	0				
0	0	1	1				
0	1	0	1				
0	1	1	1				
1	0	0	0				
1	0	1	1				
1	1	0	1				
1	1	1	1				

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$x\cdot y+x\cdot z$
0	0	0	0	0			
0	0	1	1	0			
0	1	0	1	0			
0	1	1	1	0			
1	0	0	0	0			
1	0	1	1	1			
1	1	0	1	1			
1	1	1	1	1			

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$x\cdot y+x\cdot z$
0	0	0	0	0	0		
0	0	1	1	0	0		
0	1	0	1	0	0		
0	1	1	1	0	0		
1	0	0	0	0	0		
1	0	1	1	1	0		
1	1	0	1	1	1		
1	1	1	1	1	1		

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$x\cdot y+x\cdot z$
0	0	0	0	0	0	0	
0	0	1	1	0	0	0	
0	1	0	1	0	0	0	
0	1	1	1	0	0	0	
1	0	0	0	0	0	0	
1	0	1	1	1	0	1	
1	1	0	1	1	1	0	
1	1	1	1	1	1	1	

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$x\cdot y+x\cdot z$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$x\cdot y+x\cdot z$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distributivity of $\cdot$ over $+$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$x\cdot y+x\cdot z$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x+y) \cdot (x+z)$
0	0	0	0				
0	0	1	0				
0	1	0	0				
0	1	1	1				
1	0	0	0				
1	0	1	0				
1	1	0	0				
1	1	1	1				

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0	0	0			
0	0	1	0	0			
0	1	0	0	0			
0	1	1	1	1			
1	0	0	0	1			
1	0	1	0	1			
1	1	0	0	1			
1	1	1	1	1			

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0	0	0	0		
0	0	1	0	0	0		
0	1	0	0	0	1		
0	1	1	1	1	1		
1	0	0	0	1	1		
1	0	1	0	1	1		
1	1	0	0	1	1		
1	1	1	1	1	1		

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0	0	0	0	0	
0	0	1	0	0	0	1	
0	1	0	0	0	1	0	
0	1	1	1	1	1	1	
1	0	0	0	1	1	1	
1	0	1	0	1	1	1	
1	1	0	0	1	1	1	
1	1	1	1	1	1	1	

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Distributivity of $+$ over $\cdot$

x	y	z	$y \cdot z$	$x + (y \cdot z)$	$x + y$	$x + z$	$(x + y) \cdot (x + z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

2-Valued Booean Algebra

Corollary

$\langle \{0,1\}, 0, 1, +, \cdot, {}^{-} \rangle$ *is a boolean algebra.*

Digital Design

Only the 2-valued boolean algebra is used

Convention

High school conventions

1. The '$\cdot$' symbol can be dropped, i.e., $xy = x \cdot y$.
2. '$\cdot$' takes precedence over '$+$', e.g., $x + yz = x + (y \cdot z)$.

Boolean Functions

Definition

A function $f : \mathbb{B}^n \to \mathbb{B}^m$ is called a boolean function.

Note

1. $^-$ is a 1-ary function
2. Both $\cdot$ and $+$ are 2-ary functions

Exclusive-or, xor ($\oplus$)

This is a function we all know and we use infix notation for it

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

Definable Functions

- There might be functions without a defining formula
- This is the situation with functions $\mathbb{R} \to \mathbb{R}$

Definition

- A boolean function f is definable if it falls into one of the following cases:
  - $f = 0$ or $f = 1$ or $f = x_i$.
  - $f = (\bar{g})$, where g is a definable
  - $f = (f_0 + f_1)$ where both f_0 and f_1 are definable
  - $f = (f_0 \cdot f_1)$ where both f_0 and f_1 are definable.
  - $f = g(h_0, \ldots, h_{n-1})$ where $g, h_0, \ldots, h_{n-1}$ are definable

Truth tables

Definable functions have a truth table representation

			$f(x,y,z) = x + yz$	
x	y	z	$y \cdot z$	$x + y \cdot z$
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

EXPONENTIAL

Truth tables

Definable functions have a truth table representation

			$f(x,y,z) = x + yz$	
x	y	z	$y \cdot z$	$x + y \cdot z$
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	1	

EXPONENTIAL

Truth tables

Definable functions have a truth table representation

			$f(x,y,z) = x + yz$	
x	y	z	$y \cdot z$	$x + y \cdot z$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

EXPONENTIAL

Truth tables

Definable functions have a truth table representation

			$f(x,y,z) = x + yz$	
x	y	z	$y \cdot z$	$x + y \cdot z$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

EXPONENTIAL

Why Definable Functions?

- Definable boolean functions can be realized in hardware
- Boolean functions can realize useful operations
- We need some motiviation!

(Carmi) Lecture 4 reached here

Motivation

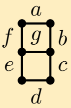
- Seven segment
- mod 3
- Addition

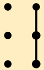
The Seven-Segment

Seven Segment

Problem

The seven segment component has seven light segments as follows:





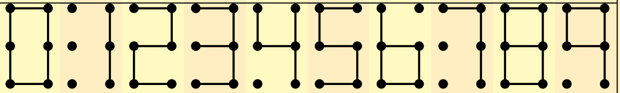
$$b = c = 1, a = d = e = f = g = 0$$

Devise a boolean formula accepting a decimal digit and showing it on the seven segment

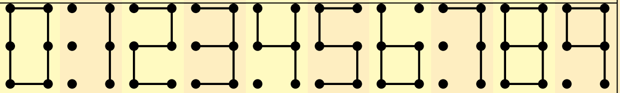
The function form

- The decimal digits will be coded in binary
- Each of the segments needs its own line
- $f = \langle g, f, e, d, c, b, a \rangle : \mathbb{B}^4 \to \mathbb{B}^7$

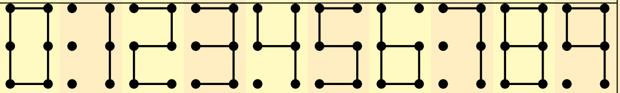
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a										
b										
c										
d										
e										
f										
g										

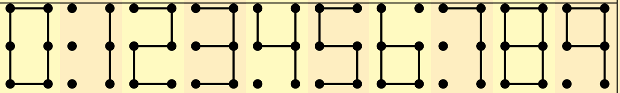
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1									
b	1									
c	1									
d	1									
e	1									
f	1									
g	0									

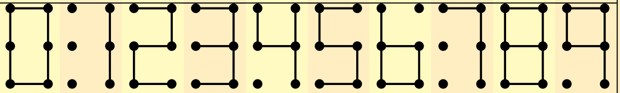
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0								
b	1	1								
c	1	1								
d	1	0								
e	1	0								
f	1	0								
g	0	0								

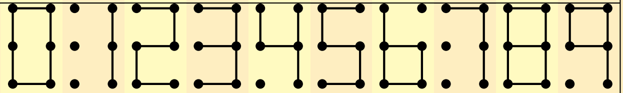
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1							
b	1	1	1							
c	1	1	0							
d	1	0	1							
e	1	0	1							
f	1	0	0							
g	0	0	1							

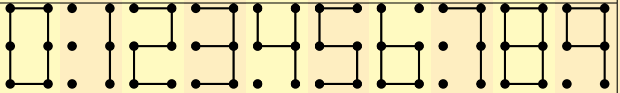
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1						
b	1	1	1	1						
c	1	1	0	1						
d	1	0	1	1						
e	1	0	1	0						
f	1	0	0	0						
g	0	0	1	1						

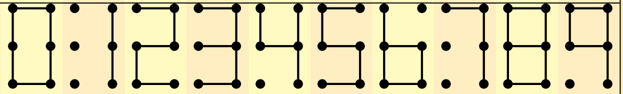
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0					
b	1	1	1	1	1					
c	1	1	0	1	1					
d	1	0	1	1	0					
e	1	0	1	0	0					
f	1	0	0	0	1					
g	0	0	1	1	1					

Seven Segment (Symbols to Boolean)

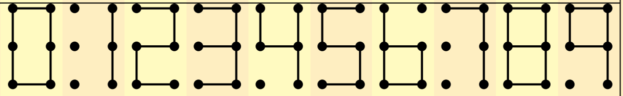
	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0	1				
b	1	1	1	1	1	0				
c	1	1	0	1	1	1				
d	1	0	1	1	0	1				
e	1	0	1	0	0	0				
f	1	0	0	0	1	1				
g	0	0	1	1	1	1				

Seven Segment (Symbols to Boolean)

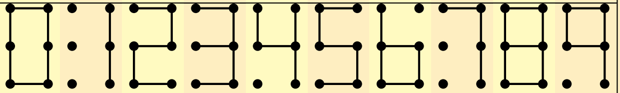


	0	1	2	3	4	5	6	7	8	9
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0	1	0			
b	1	1	1	1	1	0	0			
c	1	1	0	1	1	1	1			
d	1	0	1	1	0	1	1			
e	1	0	1	0	0	0	1			
f	1	0	0	0	1	1	1			
g	0	0	1	1	1	1	1			

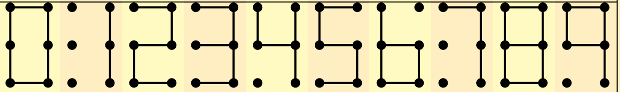
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0	1	0	1		
b	1	1	1	1	1	0	0	1		
c	1	1	0	1	1	1	1	1		
d	1	0	1	1	0	1	1	0		
e	1	0	1	0	0	0	1	0		
f	1	0	0	0	1	1	1	0		
g	0	0	1	1	1	1	1	0		

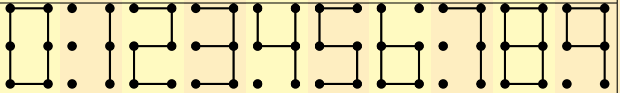
Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0	1	0	1	1	
b	1	1	1	1	1	0	0	1	1	
c	1	1	0	1	1	1	1	1	1	
d	1	0	1	1	0	1	1	0	1	
e	1	0	1	0	0	0	1	0	1	
f	1	0	0	0	1	1	1	0	1	
g	0	0	1	1	1	1	1	0	1	

Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0	1	0	1	1	1
b	1	1	1	1	1	0	0	1	1	1
c	1	1	0	1	1	1	1	1	1	1
d	1	0	1	1	0	1	1	0	1	0
e	1	0	1	0	0	0	1	0	1	0
f	1	0	0	0	1	1	1	0	1	1
g	0	0	1	1	1	1	1	0	1	1

Seven Segment (Symbols to Boolean)

	0	1	2	3	4	5	6	7	8	9
										
b_0	0	1	0	1	0	1	0	1	0	1
b_1	0	0	1	1	0	0	1	1	0	0
b_2	0	0	0	0	1	1	1	1	0	0
b_3	0	0	0	0	0	0	0	0	1	1
a	1	0	1	1	0	1	0	1	1	1
b	1	1	1	1	1	0	0	1	1	1
c	1	1	0	1	1	1	1	1	1	1
d	1	0	1	1	0	1	1	0	1	0
e	1	0	1	0	0	0	1	0	1	0
f	1	0	0	0	1	1	1	0	1	1
g	0	0	1	1	1	1	1	0	1	1

Seven Segment (thinking)

- There are four inputs b_3, b_2, b_1, b_0
- The target functions depend on the decimal digits
- (As is evident by the previous slide)
- Temporary functions $m_0, \ldots, m_9$ might be easier on us

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0										
0	0	0	1										
0	0	1	0										
0	0	1	1										
0	1	0	0										
0	1	0	1										
0	1	1	0										
0	1	1	1										
1	0	0	0										
1	0	0	1										

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1									
0	0	0	1	0									
0	0	1	0	0									
0	0	1	1	0									
0	1	0	0	0									
0	1	0	1	0									
0	1	1	0	0									
0	1	1	1	0									
1	0	0	0	0									
1	0	0	1	0									

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0								
0	0	0	1	0	1								
0	0	1	0	0	0								
0	0	1	1	0	0								
0	1	0	0	0	0								
0	1	0	1	0	0								
0	1	1	0	0	0								
0	1	1	1	0	0								
1	0	0	0	0	0								
1	0	0	1	0	0								

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0							
0	0	0	1	0	1	0							
0	0	1	0	0	0	1							
0	0	1	1	0	0	0							
0	1	0	0	0	0	0							
0	1	0	1	0	0	0							
0	1	1	0	0	0	0							
0	1	1	1	0	0	0							
1	0	0	0	0	0	0							
1	0	0	1	0	0	0							

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0						
0	0	0	1	0	1	0	0						
0	0	1	0	0	0	1	0						
0	0	1	1	0	0	0	1						
0	1	0	0	0	0	0	0						
0	1	0	1	0	0	0	0						
0	1	1	0	0	0	0	0						
0	1	1	1	0	0	0	0						
1	0	0	0	0	0	0	0						
1	0	0	1	0	0	0	0						

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0					
0	0	0	1	0	1	0	0	0					
0	0	1	0	0	0	1	0	0					
0	0	1	1	0	0	0	1	0					
0	1	0	0	0	0	0	0	1					
0	1	0	1	0	0	0	0	0					
0	1	1	0	0	0	0	0	0					
0	1	1	1	0	0	0	0	0					
1	0	0	0	0	0	0	0	0					
1	0	0	1	0	0	0	0	0					

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0				
0	0	0	1	0	1	0	0	0	0				
0	0	1	0	0	0	1	0	0	0				
0	0	1	1	0	0	0	1	0	0				
0	1	0	0	0	0	0	0	1	0				
0	1	0	1	0	0	0	0	0	1				
0	1	1	0	0	0	0	0	0	0				
0	1	1	1	0	0	0	0	0	0				
1	0	0	0	0	0	0	0	0	0				
1	0	0	1	0	0	0	0	0	0				

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0	0			
0	0	0	1	0	1	0	0	0	0	0			
0	0	1	0	0	0	1	0	0	0	0			
0	0	1	1	0	0	0	1	0	0	0			
0	1	0	0	0	0	0	0	1	0	0			
0	1	0	1	0	0	0	0	0	1	0			
0	1	1	0	0	0	0	0	0	0	1			
0	1	1	1	0	0	0	0	0	0	0			
1	0	0	0	0	0	0	0	0	0	0			
1	0	0	1	0	0	0	0	0	0	0			

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0	0	0		
0	0	0	1	0	1	0	0	0	0	0	0		
0	0	1	0	0	0	1	0	0	0	0	0		
0	0	1	1	0	0	0	1	0	0	0	0		
0	1	0	0	0	0	0	0	1	0	0	0		
0	1	0	1	0	0	0	0	0	1	0	0		
0	1	1	0	0	0	0	0	0	0	1	0		
0	1	1	1	0	0	0	0	0	0	0	1		
1	0	0	0	0	0	0	0	0	0	0	0		
1	0	0	1	0	0	0	0	0	0	0	0		

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0	0	0	0	
0	0	0	1	0	1	0	0	0	0	0	0	0	
0	0	1	0	0	0	1	0	0	0	0	0	0	
0	0	1	1	0	0	0	1	0	0	0	0	0	
0	1	0	0	0	0	0	0	1	0	0	0	0	
0	1	0	1	0	0	0	0	0	1	0	0	0	
0	1	1	0	0	0	0	0	0	0	1	0	0	
0	1	1	1	0	0	0	0	0	0	0	1	0	
1	0	0	0	0	0	0	0	0	0	0	0	1	
1	0	0	1	0	0	0	0	0	0	0	0	0	

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

Seven Segment (temporary functions)

b_3	b_2	b_1	b_0	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

$m_0 = \bar{b}_3\bar{b}_2\bar{b}_1\bar{b}_0 \quad m_3 = \bar{b}_3\bar{b}_2 b_1 b_0 \quad m_6 = \bar{b}_3 b_2 b_1\bar{b}_0 \quad m_9 = b_3\bar{b}_2\bar{b}_1 b_0$

$m_1 = \bar{b}_3\bar{b}_2\bar{b}_1 b_0 \quad m_4 = \bar{b}_3 b_2\bar{b}_1\bar{b}_0 \quad m_7 = \bar{b}_3 b_2 b_1 b_0$

$m_2 = \bar{b}_3\bar{b}_2 b_1\bar{b}_0 \quad m_5 = \bar{b}_3 b_2\bar{b}_1 b_0 \quad m_8 = b_3\bar{b}_2\bar{b}_1\bar{b}_0$

Seven Segment (calculating a)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	a
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

Seven Segment (calculating a)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	a
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

$$a = m_0 + m_2 + m_3 + m_5 + m_7 + m_8 + m_9$$

Seven Segment (calculating a)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	a
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

$$a = m_0 + m_2 + m_3 + m_5 + m_7 + m_8 + m_9$$
$$a = \bar{m}_1\bar{m}_4\bar{m}_6$$

Seven Segment (calculating b)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	b
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

$$b = m_0 + m_1 + m_2 + m_3 + m_4 + m_7 + m_8 + m_9$$
$$b = \bar{m}_5 \bar{m}_6$$

Seven Segment (calculating c)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	c
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

$$c = m_0 + m_1 + m_3 + m_4 + m_5 + m_6 + m_7 + m_8 + m_9$$
$$c = \bar{m}_2$$

Seven Segment (calculating d)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	d
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0

$$d = m_0 + m_2 + m_3 + m_5 + m_6 + m_8$$
$$d = \bar{m}_1 \bar{m}_4 \bar{m}_7 \bar{m}_9$$

Seven Segment (calculating e)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	d_9	e
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	0

$$e = m_0 + m_2 + m_6 + m_8$$

$$e = \bar{m}_1 \bar{m}_3 \bar{m}_4 \bar{m}_5 \bar{m}_7 \bar{m}_9$$

Seven Segment (calculating f)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	d_9	f
1	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

$$
\begin{aligned}
f &= m_0 + m_4 + m_5 + m_6 + m_8 + m_9 \\
f &= \bar{m}_1 \bar{m}_2 \bar{m}_3 \bar{m}_7
\end{aligned}
$$

Seven Segment (calculating g)

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	d_9	g
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	0	1	1

$$g = m_2 + m_3 + m_4 + m_5 + m_6 + m_8 + m_9$$
$$g = \bar{m}_0 \bar{m}_1 \bar{m}_7$$

Seven Segment (notes)

- **Lots** of work
- For illegal input the output is garbage
- No mathematical laws used in the table construction
- Here, we have no way but to use truth tables
- In **this** case reading formula from the table is easy

mod 3

A mod 3 function

Problem

Devise a boolean formula for computing n mod 3 for $0 \leq n \leq 15$

The function form

- We use binary coding
- 4-bits input
- 2-bits output
- Thus the function is of the form $f = \langle f_1, f_0 \rangle : \mathbb{B}^4 \to \mathbb{B}^2$

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0		0	0	0	0		
1		0	0	0	1		
2		0	0	1	0		
3		0	0	1	1		
4		0	1	0	0		
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1		0	0	0	1		
2		0	0	1	0		
3		0	0	1	1		
4		0	1	0	0		
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2		0	0	1	0		
3		0	0	1	1		
4		0	1	0	0		
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3		0	0	1	1		
4		0	1	0	0		
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4		0	1	0	0		
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5		0	1	0	1		
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6		0	1	1	0		
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7		0	1	1	1		
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8		1	0	0	0		
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9		1	0	0	1		
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10		1	0	1	0		
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11		1	0	1	1		
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12		1	1	0	0		
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12	0	1	1	0	0	0	0
13		1	1	0	1		
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12	0	1	1	0	0	0	0
13	1	1	1	0	1	0	1
14		1	1	1	0		
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12	0	1	1	0	0	0	0
13	1	1	1	0	1	0	1
14	2	1	1	1	0	1	0
15		1	1	1	1		

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12	0	1	1	0	0	0	0
13	1	1	1	0	1	0	1
14	2	1	1	1	0	1	0
15	0	1	1	1	1	0	0

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12	0	1	1	0	0	0	0
13	1	1	1	0	1	0	1
14	2	1	1	1	0	1	0
15	0	1	1	1	1	0	0

mod 3 (truth table)

Decimal		Binary					
n	n mod 3	n_3	n_2	n_1	n_0	f_1	f_0
0	0	0	0	0	0	0	0
1	1	0	0	0	1	0	1
2	2	0	0	1	0	1	0
3	0	0	0	1	1	0	0
4	1	0	1	0	0	0	1
5	2	0	1	0	1	1	0
6	0	0	1	1	0	0	0
7	1	0	1	1	1	0	1
8	2	1	0	0	0	1	0
9	0	1	0	0	1	0	0
10	1	1	0	1	0	0	1
11	2	1	0	1	1	1	0
12	0	1	1	0	0	0	0
13	1	1	1	0	1	0	1
14	2	1	1	1	0	1	0
15	0	1	1	1	1		0

How do we get formulae from this?

Definability of Boolean Functions (SOP form) extra

Theorem

The boolean functions are definable.

Proof.

It is enough to show that the functions $f : \mathbb{B}^n \to \mathbb{B}$ are definable. We do this by induction.

$\underline{n = 0}$: A 0-ary function is a constant, that is either 0 or 1. Thus definability is immediate.

$\underline{n + 1}$: Let $f : \mathbb{B}^{n+1} \to \mathbb{B}$ be a function. Let $f_0(x_{n-1}, \ldots, x_0) = f(0, x_{n-1}, \ldots, x_0)$ and $f_1(x_{n-1}, \ldots, x_0) = f(1, x_{n-1}, \ldots, x_0)$. By induction the functions f_0 and f_1 are definable, hence the function $f(x_n, \ldots, x_0) = \bar{x}_n \cdot f_0(x_{n-1}, \ldots, x_0) + x_n \cdot f_1(x_{n-1}, \ldots, x_0)$ is definable. $\square$

Proof of definability (POS form) extra

Proof.

$\underline{n+1}$: Let $f : \mathbb{B}^{n+1} \to \mathbb{B}$ be a function. Let $f_0(x_{n-1}, \ldots, x_0) = f(0, x_{n-1}, \ldots, x_0)$ and $f_1(x_{n-1}, \ldots, x_0) = f(1, x_{n-1}, \ldots, x_0)$. By induction the functions f_0 and f_1 are definable, hence the function $f(x_n, \ldots, x_0) = (x_n + f_0(x_{n-1}, \ldots, x_0)) \cdot (\bar{x}_n + f_1(x_{n-1}, \ldots, x_0))$ is definable. $\square$

mod 3 unwinding f_1 (step 1) extra

n_3	n_2	n_1	n_0	f_1
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
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	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

mod 3 unwinding f_1 (step 2) extra

$\bar{n}_3$			
n_2	n_1	n_0	f_1^0
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

n_3			
n_2	n_1	n_0	f_1^1
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$$f_1 = \bar{n}_3 f_1^0 + n_3 f_1^1$$

mod 3 unwinding f_1^0, f_1^1 (step 3) extra

$\bar{n}_3\bar{n}_2$		
n_1	n_0	f_1^{00}
0	0	0
0	1	0
1	0	1
1	1	0

$\bar{n}_3 n_2$		
n_1	n_0	f_1^{01}
0	0	0
0	1	1
1	0	0
1	1	0

$$f_1^0 = \bar{n}_3\bar{n}_2 f_1^{00} + \bar{n}_3\bar{n}_2 f_1^{01}$$

$n_3\bar{n}_2$		
n_1	n_0	f_1^{10}
0	0	1
0	1	0
1	0	0
1	1	1

$n_3 n_2$		
n_1	n_0	f_1^{11}
0	0	0
0	1	0
1	0	1
1	1	0

$$f_1^0 = n_3\bar{n}_2 f_1^{10} + n_3\bar{n}_2 f_1^{11}$$

mod 3 unwinding $f_1^{00}, f_1^{01}, f_1^{10}, f_1^{11}$ (step 4) extra

$\bar{n}_3\bar{n}_2\bar{n}_1$	
n_0	f_1^{000}
0	0
1	0

$f_1^{000} = 0$

$\bar{n}_3\bar{n}_2 n_1$	
n_0	f_1^{001}
0	1
1	0

$f_1^{001} = \bar{n}_0$

$\bar{n}_3 n_2\bar{n}_1$	
n_0	f_1^{010}
0	0
1	1

$f_1^{010} = n_0$

$\bar{n}_3 n_2 n_1$	
n_0	f_1^{011}
0	0
1	0

$f_1^{011} = 0$

$n_3\bar{n}_2\bar{n}_1$	
n_0	f_1^{100}
0	1
1	0

$f_1^{100} = \bar{n}_0$

$n_3\bar{n}_2 n_1$	
n_0	f_1^{101}
0	0
1	1

$f_1^{101} = n_0$

$n_3 n_2\bar{n}_1$	
	f_1^{110}
0	0
1	0

$f_1^{110} = 0$

$n_3 n_2 n_1$	
n_0	f_1^{111}
0	1
1	0

$f_1^{111} = \bar{n}_0$

$$f_1^{00} = \bar{n}_3\bar{n}_2\bar{n}_1 f_1^{000} + \bar{n}_3\bar{n}_2 n_1 f_1^{001} \qquad f_1^{01} = \bar{n}_3 n_2\bar{n}_1 f_1^{010} + \bar{n}_3 n_2 n_1 f_1^{011}$$
$$f_1^{10} = n_3\bar{n}_2\bar{n}_1 f_1^{100} + n_3\bar{n}_2 n_1 f_1^{101} \qquad f_1^{11} = n_3 n_2\bar{n}_1 f_1^{110} + n_3 n_2 n_1 f_1^{111}$$

mod 3 f_1 extra

- Note that we skipped the last unwinding
- We can do one variable, no need to go to zero variables
- But possible...

$$
\begin{aligned}
f_1 = & \bar{n}_3\bar{n}_2\bar{n}_1 f_1^{000} + \bar{n}_3\bar{n}_2 n_1 f_1^{001} + \bar{n}_3 n_2\bar{n}_1 f_1^{010} + \bar{n}_3 n_2 n_1 f_1^{011} + \\
& n_3\bar{n}_2\bar{n}_1 f_1^{100} + n_3\bar{n}_2 n_1 f_1^{101} + n_3 n_2\bar{n}_1 f_1^{110} + n_3 n_2 n_1 f_1^{111} \\
= & \bar{n}_3\bar{n}_2\bar{n}_1 \cdot 0 + \bar{n}_3\bar{n}_2 n_1\bar{n}_0 + \bar{n}_3 n_2\bar{n}_1 n_0 + \bar{n}_3 n_2 n_1 \cdot 0 + \\
& n_3\bar{n}_2\bar{n}_1\bar{n}_0 + n_3\bar{n}_2 n_1 n_0 + n_3 n_2\bar{n}_1 \cdot 0 + n_3 n_2 n_1\bar{n}_0 \\
= & \bar{n}_3\bar{n}_2 n_1\bar{n}_0 + \bar{n}_3 n_2\bar{n}_1 n_0 + \\
& n_3\bar{n}_2\bar{n}_1\bar{n}_0 + n_3\bar{n}_2 n_1 n_0 + n_3 n_2 n_1\bar{n}_0
\end{aligned}
$$

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
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	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
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	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
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	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	1	0
1	1	1	1	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	1	0
1	1	1	1	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	1	0
1	1	1	1	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	
0	0	0	0	0	0	0	
0	0	0	1	0	0	0	
0	0	1	0	1	1	0	
0	0	1	1	0	0	0	
0	1	0	0	0	0	0	
0	1	0	1	1	0	1	
0	1	1	0	0	0	0	
0	1	1	1	0	0	0	
1	0	0	0	1	0	0	
1	0	0	1	0	0	0	
1	0	1	0	0	0	0	
1	0	1	1	1	0	0	
1	1	0	0	0	0	0	
1	1	0	1	0	0	0	
1	1	1	0	1	0	0	
1	1	1	1	0	0	0	

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	1	0	1
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	1	0	1
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	0	0	0
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	
0	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	
0	0	1	0	1	1	0	0	
0	0	1	1	0	0	0	0	
0	1	0	0	0	0	0	0	
0	1	0	1	1	0	1	0	
0	1	1	0	0	0	0	0	
0	1	1	1	0	0	0	0	
1	0	0	0	1	0	0	1	
1	0	0	1	0	0	0	0	
1	0	1	0	0	0	0	0	
1	0	1	1	1	0	0	0	
1	1	0	0	0	0	0	0	
1	1	0	1	0	0	0	0	
1	1	1	0	1	0	0	0	
1	1	1	1	0	0	0	0	

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	1	0	1	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	1	0	1	0
0	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	0	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	m_{11}	
0	0	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	0	0	
0	0	1	0	1	1	0	0	0	
0	0	1	1	0	0	0	0	0	
0	1	0	0	0	0	0	0	0	
0	1	0	1	1	0	1	0	0	
0	1	1	0	0	0	0	0	0	
0	1	1	1	0	0	0	0	0	
1	0	0	0	1	0	0	1	0	
1	0	0	1	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	
1	0	1	1	1	0	0	0	1	
1	1	0	0	0	0	0	0	0	
1	1	0	1	0	0	0	0	0	
1	1	1	0	1	0	0	0	0	
1	1	1	1	0	0	0	0	0	

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	m_{11}
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	1	1	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	1	0	0	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	m_{11}
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0
0	0	1	0	1	1	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0	0
0	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0
1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1
1	1	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0
1	1	1	0	1	0	0	0	0
1	1	1	1	0	0	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	m_{11}	m_{14}
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	m_{11}	m_{14}
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0

mod 3 f_1 in canonical SOP

n_3	n_2	n_1	n_0	f_1	m_2	m_5	m_8	m_{11}	m_{14}
0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	1	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	1	1	0	1	0	0	0
0	1	1	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
1	0	0	0	1	0	0	1	0	0
1	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0

$$f_1(n_3, n_2, n_1, n_0) = \sum(2, 5, 8, 11, 14) = \bar{n}_3\bar{n}_2 n_1\bar{n}_0 + \bar{n}_3 n_2\bar{n}_1 n_0 + n_3\bar{n}_2\bar{n}_1\bar{n}_0 + n_3\bar{n}_2 n_1 n_0 + n_3 n_2 n_1\bar{n}_0$$

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
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	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
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	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
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	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	1	0	
1	1	1	0	1	
1	1	1	1	0	

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	1	1
1	1	1	1	0	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	1	1
1	1	1	1	0	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	1	1
1	1	1	1	0	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	1	1	1
0	0	1	1	0	1	1
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	1	1	1
0	0	1	1	0	1	1
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	1	1	1
0	0	1	1	0	1	1
0	1	0	0	0	1	1
0	1	0	1	1	1	1
0	1	1	0	0	1	1
0	1	1	1	0	1	1
1	0	0	0	1	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	1	1	1
1	1	0	0	0	1	1
1	1	0	1	0	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	1
0	0	1	0	1	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	0	1	1	1
0	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	1
1	0	1	1	1	1	1	1
1	1	0	0	0	1	1	1
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	1
0	0	1	0	1	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	0	1	1	1
0	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	1
1	0	1	1	1	1	1	1
1	1	0	0	0	1	1	1
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	1
0	0	1	0	1	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	1	1	1
0	1	1	0	0	1	1	1
0	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	1
1	0	1	1	1	1	1	1
1	1	0	0	0	1	1	1
1	1	0	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4
0	0	0	0	0	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	1	0	1	1	1	1	1
0	0	1	1	0	1	1	0	1
0	1	0	0	0	1	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1
0	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1
1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1
1	1	0	1	0	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4
0	0	0	0	0	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	1	0	1	1	1	1	1
0	0	1	1	0	1	1	0	1
0	1	0	0	0	1	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1
0	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1
1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1
1	1	0	1	0	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4
0	0	0	0	0	0	1	1	1
0	0	0	1	0	1	0	1	1
0	0	1	0	1	1	1	1	1
0	0	1	1	0	1	1	0	1
0	1	0	0	0	1	1	1	0
0	1	0	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1
0	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1
1	0	0	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1
1	1	0	1	0	1	1	1	1
1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6
0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	1	0	1	1	1
0	0	1	0	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1
0	1	0	0	0	1	1	1	0	1
0	1	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6
0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	1	0	1	1	1
0	0	1	0	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1
0	1	0	0	0	1	1	1	0	1
0	1	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6
0	0	0	0	0	0	1	1	1	1
0	0	0	1	0	1	0	1	1	1
0	0	1	0	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1
0	1	0	0	0	1	1	1	0	1
0	1	0	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7
0	0	0	0	0	0	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1
0	1	0	0	0	1	1	1	0	1	1
0	1	0	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7
0	0	0	0	0	0	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1
0	1	0	0	0	1	1	1	0	1	1
0	1	0	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7
0	0	0	0	0	0	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1
0	1	0	0	0	1	1	1	0	1	1
0	1	0	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1
0	1	1	1	0	1	1	1	1	1	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9
0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0
1	0	1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9
0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0
1	0	1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9
0	0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0
1	0	1	0	0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}
0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1
1	0	1	0	0	1	1	1	1	1	1	1	0
1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}
0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1
1	0	1	0	0	1	1	1	1	1	1	1	0
1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}
0	0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1
1	0	1	0	0	1	1	1	1	1	1	1	0
1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1
1	1	0	1	0	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}
0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0
1	1	0	1	0	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}
0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0
1	1	0	1	0	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}
0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0
1	1	0	1	0	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}	M_{13}	
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	
0	0	1	1	0	1	1	0	1	1	1	1	1	1	1	
0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	
0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	
0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	
1	0	0	1	0	1	1	1	1	1	1	0	1	1	1	
1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	
1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}	M_{13}
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	1	0	1	0	1	1	1	1	1	1	1	1	1	0
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}	M_{13}
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	1	0	1	0	1	1	1	1	1	1	1	1	1	0
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}	M_{13}	M_{15}
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1
1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}	M_{13}	M_{15}
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1
1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0

mod 3 f_1 in canonical POS

n_3	n_2	n_1	n_0	f_1	M_0	M_1	M_3	M_4	M_6	M_7	M_9	M_{10}	M_{12}	M_{13}	M_{15}
0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	0	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1
1	0	1	0	0	1	1	1	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1
1	1	0	1	0	1	1	1	1	1	1	1	1	1	0	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0

$f_1(n_3, n_2, n_1, n_0) =$
$\prod(0, 1, 3, 4, 6, 7,$
$9, 10, 12, 13, 15) =$
$(n_3 + n_2 + n_1 + n_0)\cdot$
$(n_3 + n_2 + n_1 + \bar{n}_0)\cdot$
$(n_3 + n_2 + \bar{n}_1 + \bar{n}_0)\cdot$
$(n_3 + \bar{n}_2 + n_1 + n_0)\cdot$
$(n_3 + \bar{n}_2 + n_1 + n_0)\cdot$
$(n_3 + \bar{n}_2 + \bar{n}_1 + \bar{n}_0)\cdot$
$(\bar{n}_3 + n_2 + n_1 + \bar{n}_0)\cdot$
$(\bar{n}_3 + n_2 + \bar{n}_1 + n_0)\cdot$
$(\bar{n}_3 + \bar{n}_2 + n_1 + n_0)\cdot$
$(\bar{n}_3 + \bar{n}_2 + n_1 + \bar{n}_0)\cdot$
$(\bar{n}_3 + \bar{n}_2 + \bar{n}_1 + \bar{n}_0)\cdot$

n mod 3 formulae

SOP

$$f_1 = \bar{n}_3\bar{n}_2 n_1\bar{n}_0 + \bar{n}_3 n_2\bar{n}_1 n_0 + n_3\bar{n}_2\bar{n}_1\bar{n}_0 + n_3\bar{n}_2 n_1 n_0 + n_3 n_2 n_1\bar{n}_0$$

$$f_0 = \bar{n}_3\bar{n}_2\bar{n}_1 n_0 + \bar{n}_3 n_2\bar{n}_1\bar{n}_0 + \bar{n}_3 n_2 n_1 n_0 + n_3\bar{n}_2 n_1\bar{n}_0 + n_3 n_2\bar{n}_1 n_0$$

Note

In this case the POS form is uglier

Larger n's, kind of nasty

mod 3 (notes)

- There is an algorithm for mod
- We used it to **build** the truth table
- We did **not** implement the algorithm
- Truth table implementation is mechanical
- However, it is not practical for large n's
- Usually it is also fast (relevant when we have hardware)
- Algorithm implemetation is usually harder

Truth tables do not scale up!!!!

However, they might give us more optimized functions

(Carmi) Lecture 5 reached here

Addition

The plan:

- One bit adder (a.k.a Half Adder)
- Two bits adder
- Three bits adder?! Wrong direction
- Summing three bits (a.k.a Full Adder)
- n-Bits adder (a.k.a Binary Adder)

One Bit Adder

Problem

Devise a formula to add two one-bit numbers

One Bit Adder

Problem

Devise a formula to add two one-bit numbers

- 2-bits inputs: Maximal possible sum is 2.
- Hence $f = \langle f_1, f_0 \rangle : \mathbb{B}^2 \to \mathbb{B}^2$

One Bit Adder

Problem

Devise a formula to add two one-bit numbers

- 2-bits inputs: Maximal possible sum is 2.
- Hence $f = \langle f_1, f_0 \rangle : \mathbb{B}^2 \to \mathbb{B}^2$

		Decimal	Binary	
x	y	$x + y$	f_1	f_0
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

One Bit Adder

Problem

Devise a formula to add two one-bit numbers

- 2-bits inputs: Maximal possible sum is 2.
- Hence $f = \langle f_1, f_0 \rangle : \mathbb{B}^2 \to \mathbb{B}^2$

		Decimal	Binary	
x	y	$x+y$	f_1	f_0
0	0	0	0	0
0	1	1	0	1
1	0	1	0	1
1	1	2	1	0

$$f_1 = xy \qquad f_0 = x \oplus y$$

Two Bits Adder

Problem

Devise a formula to calculate the sum of two numbers each in the range 0–3

The function form

Two Bits Adder

Problem

Devise a formula to calculate the sum of two numbers each in the range 0–3

The function form

- Of course we use binary coding

Two Bits Adder

Problem

Devise a formula to calculate the sum of two numbers each in the range 0–3

The function form

- Of course we use binary coding
- Each of the inputs is 2-bits wide

Two Bits Adder

Problem

Devise a formula to calculate the sum of two numbers each in the range 0–3

The function form

- Of course we use binary coding
- Each of the inputs is 2-bits wide
- Thus sum is 6 at most

Two Bits Adder

Problem

Devise a formula to calculate the sum of two numbers each in the range 0–3

The function form

- Of course we use binary coding
- Each of the inputs is 2-bits wide
- Thus sum is 6 at most
- Thus the output is 3-bits wide

Two Bits Adder

Problem

Devise a formula to calculate the sum of two numbers each in the range 0–3

The function form

- Of course we use binary coding
- Each of the inputs is 2-bits wide
- Thus sum is 6 at most
- Thus the output is 3-bits wide
- The function is of the form $f = \langle f_2, f_1, f_0 \rangle : \mathbb{B}^4 \to \mathbb{B}^3$

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0		0	0	0	0			
0	1		0	0	0	1			
0	2		0	0	1	0			
0	3		0	0	1	1			
1	0		0	1	0	0			
1	1		0	1	0	1			
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1		0	0	0	1			
0	2		0	0	1	0			
0	3		0	0	1	1			
1	0		0	1	0	0			
1	1		0	1	0	1			
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2		0	0	1	0			
0	3		0	0	1	1			
1	0		0	1	0	0			
1	1		0	1	0	1			
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3		0	0	1	1			
1	0		0	1	0	0			
1	1		0	1	0	1			
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0		0	1	0	0			
1	1		0	1	0	1			
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1		0	1	0	1			
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2		0	1	1	0			
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3		0	1	1	1			
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0		1	0	0	0			
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1		1	0	0	1			
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2		1	0	1	0			
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3		1	0	1	1			
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3	5	1	0	1	1	1	0	1
3	0		1	1	0	0			
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3	5	1	0	1	1	1	0	1
3	0	3	1	1	0	0	0	1	1
3	1		1	1	0	1			
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3	5	1	0	1	1	1	0	1
3	0	3	1	1	0	0	0	1	1
3	1	4	1	1	0	1	1	0	0
3	2		1	1	1	0			
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3	5	1	0	1	1	1	0	1
3	0	3	1	1	0	0	0	1	1
3	1	4	1	1	0	1	1	0	0
3	2	5	1	1	1	0	1	0	1
3	3		1	1	1	1			

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3	5	1	0	1	1	1	0	1
3	0	3	1	1	0	0	0	1	1
3	1	4	1	1	0	1	1	0	0
3	2	5	1	1	1	0	1	0	1
3	3	6	1	1	1	1	1	1	0

2-Bits Binary Adder (truth table)

Decimal			Binary						
x	y	$x+y$	x_1	x_0	y_1	y_0	f_2	f_1	f_0
0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	0	0	1
0	2	2	0	0	1	0	0	1	0
0	3	3	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0	0	1
1	1	2	0	1	0	1	0	1	0
1	2	3	0	1	1	0	0	1	1
1	3	4	0	1	1	1	1	0	0
2	0	2	1	0	0	0	0	1	0
2	1	3	1	0	0	1	0	1	1
2	2	4	1	0	1	0	1	0	0
2	3	5	1	0	1	1	1	0	1
3	0	3	1	1	0	0	0	1	1
3	1	4	1	1	0	1	1	0	0
3	2	5	1	1	1	0	1	0	1
3	3	6	1	1	1	1	1	1	0

2-Bits Binary Adder (formula)

$$
\begin{aligned}
f_2 =& \bar{x}_1 x_0 y_1 y_0 + x_1 \bar{x}_0 y_1 \bar{y}_0 + x_1 \bar{x}_0 y_1 y_0 + x_1 x_0 \bar{y}_1 y_0 + \\
& x_1 x_0 y_1 \bar{y}_0 + x_1 x_0 y_1 y_0 \\
f_1 =& \bar{x}_1 \bar{x}_0 y_1 \bar{y}_0 + \bar{x}_1 \bar{x}_0 y_1 y_0 + \bar{x}_1 x_0 \bar{y}_1 y_0 + \bar{x}_1 x_0 y_1 \bar{y}_0 + \\
& x_1 \bar{x}_0 \bar{y}_1 \bar{y}_0 + x_1 \bar{x}_0 \bar{y}_1 y_0 + x_1 x_0 \bar{y}_1 \bar{y}_0 + x_1 x_0 y_1 y_0 \\
f_0 =& \bar{x}_1 \bar{x}_0 y_1 \bar{y}_0 + \bar{x}_1 \bar{x}_0 y_1 y_0 + \bar{x}_1 x_0 \bar{y}_1 \bar{y}_0 + \bar{x}_1 x_0 y_1 \bar{y}_0 + \\
& x_1 \bar{x}_0 \bar{y}_1 y_0 + x_1 \bar{x}_0 y_1 y_0 + x_1 x_0 \bar{y}_1 \bar{y}_0 + x_1 x_0 y_1 \bar{y}_0
\end{aligned}
$$

The notes of mod 3 hold also here

More than Two Bits Adder

- Three bits: $f : \mathbb{B}^6 \to \mathbb{B}^4$
- The general case looks hopeless
- However, we have seen how to add numbers:
  - Long addition of representations
- Until now our method was as follows:
  - Use an algoritm to generate truth table
  - Generate a formula from the truth table

  The exponential explosion requires something else
- The formula we generate will implement the algorithm

Recalling Long Addition

$$
\begin{array}{r|}
 & & & & \\
x_3 & x_2 & x_1 & x_0 \\
y_3 & y_2 & y_1 & y_0 \\
\hline
 & & & \\
\end{array}
$$

Recalling Long Addition

$$\begin{array}{rrrrr|}
 & & & & 0 \\
 & x_3 & x_2 & x_1 & x_0 \\
 & y_3 & y_2 & y_1 & y_0 \\
\hline
 & & & & \\
\end{array}$$

Recalling Long Addition

$$
\begin{array}{cccc|cc}
 & & & 0 & & 0 \\
x_3 & x_2 & x_1 & x_0 & & x_0 \\
y_3 & y_2 & y_1 & y_0 & & y_0 \\
\hline
 & & & & c_0 & z_0
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccc|cc}
 & & c_0 & & & 0 \\
x_3 & x_2 & x_1 & x_0 & & x_0 \\
y_3 & y_2 & y_1 & y_0 & & y_0 \\
\hline
 & & & z_0 & c_0 & z_0
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccc|cc}
 & & c_0 & & & c_0 \\
x_3 & x_2 & x_1 & x_0 & & x_1 \\
y_3 & y_2 & y_1 & y_0 & & y_1 \\
\hline
 & & & z_0 & c_1 & z_1
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccc|cc}
 & c_1 & & & & c_0 \\
x_3 & x_2 & x_1 & x_0 & & x_1 \\
y_3 & y_2 & y_1 & y_0 & & y_1 \\
\hline
 & & z_1 & z_0 & c_1 & z_1
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccc|cc}
 & c_1 & & & & c_1 \\
x_3 & x_2 & x_1 & x_0 & & x_2 \\
y_3 & y_2 & y_1 & y_0 & & y_2 \\
\hline
 & & z_1 & z_0 & c_2 & z_2
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{ccccc|cc}
 & c_2 & & & & & c_1 \\
 & x_3 & x_2 & x_1 & x_0 & & x_2 \\
 & y_3 & y_2 & y_1 & y_0 & & y_2 \\
\hline
 & & z_2 & z_1 & z_0 & c_2 & z_2
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{ccccc|cc}
 & c_2 & & & & & c_2 \\
 & x_3 & x_2 & x_1 & x_0 & & x_3 \\
 & y_3 & y_2 & y_1 & y_0 & & y_3 \\
\hline
 & & z_2 & z_1 & z_0 & c_3 & z_3
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccccc|cc}
c_3 & c_2 & & & & & & c_2 \\
 & x_3 & x_2 & x_1 & x_0 & & & x_3 \\
 & y_3 & y_2 & y_1 & y_0 & & & y_3 \\
\hline
 & z_3 & z_2 & z_1 & z_0 & & c_3 & z_3
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccccc|}
c_3 & c_2 & & & \\
 & x_3 & x_2 & x_1 & x_0 \\
 & y_3 & y_2 & y_1 & y_0 \\
\hline
 & z_3 & z_2 & z_1 & z_0
\end{array}
$$

Recalling Long Addition

$$
\begin{array}{cccccc|}
c_3 & c_2 & & & \\
 & x_3 & x_2 & x_1 & x_0 \\
 & y_3 & y_2 & y_1 & y_0 \\
\hline
 & z_3 & z_2 & z_1 & z_0
\end{array}
$$

- Unsigned overflow: $c_3 = 1$
- Signed overflow: $c_3 \neq c_2$

Recalling Long Addition

$$\begin{array}{ccccc|} c_3 & c_2 & & & \\ & x_3 & x_2 & x_1 & x_0 \\ & y_3 & y_2 & y_1 & y_0 \\ \hline & z_3 & z_2 & z_1 & z_0 \end{array}$$

- Unsigned overflow: $c_3 = 1$
- Signed overflow: $c_3 \neq c_2$

Corollary

There is a function f such that for each n,

$$\langle c_n, z_n \rangle = f(c_{n-1}, x_n, y_n),$$

where $c_{-1} = 0$

Summing Three Bits (full adder)

Problem

Devise a formula to sum three bits

The form of the function

- Three bits input
- The sum is at most three
- Thus the output is two bits
- $f = \langle f_1, f_0 \rangle : \mathbb{B}^3 \to \mathbb{B}^2$

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Addition of Three Bits (formula)

x	y	z	Decimal $x+y+z$	Binary f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1			
1	1	0			
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0			
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1			

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

Addition of Three Bits (formula)

			Decimal	Binary	
x	y	z	$x+y+z$	f_1	f_0
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

- $f_0 = x \oplus y \oplus z$
- $f_1(x, y, z) = \sum(3, 5, 6, 7) = \prod(0, 1, 2, 4)$

Simplifying the SOP version

$$
\begin{aligned}
f_1 =& \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz = \\
=& (\bar{x} + x)yz + (\bar{y} + y)xz + xy(\bar{z} + z) = \\
=& yz + xz + xy
\end{aligned}
$$

Binary Adder **without** Truth Tables

Well, almost, we use the full adder

n-Bits Binary Adder

Problem

For each n devise a function to compute the sum of two numbers each in the range $0 - 2^n - 1$

Function Form

- Of course, binary
- Each of the input numbers is n-bits wide
- The output is $n+1$-bits wide
- The form is $f = \langle f_n, \cdots, f_0 \rangle : \mathbb{B}^{2n} \to \mathbb{B}^{n+1}$

n-Bits Binary Adder (Algorithm)

Definitions

- Let $x_{n-1} \cdots x_0$ and $y_{n-1} \cdots y_0$ be the binary representation of the two input numbers
- Let $z_n \cdots z_0$ be the binary representation of the sum
- Let $f = \langle f_1, f_0 \rangle : \mathbb{B}^3 \to \mathbb{B}^2$ be the full adder

Long addition

$$
\begin{aligned}
\langle c_0, z_0 \rangle &= \langle f_1(x_0, y_0, 0), f_0(x_0, y_0, 0) \rangle \\
\langle c_1, z_1 \rangle &= \langle f_1(x_1, y_1, c_0), f_0(x_1, y_1, c_0) \rangle \\
\vdots &= \vdots \\
\langle c_{n-1}, z_{n-1} \rangle &= \langle f_1(x_{n-1}, y_{n-1}, c_{n-2}), f_0(x_{n-1}, y_{n-1}, c_{n-2}) \rangle \\
z_n &= c_{n-1}
\end{aligned}
$$

Adder, more formal extra

Define the functions g_0 and g_1

$$g_0(k, x_k, y_k) = \begin{cases} f_0(x_k, y_k, 0) & k = 0, \\ f_0(x_k, y_k, g_1(k-1, x_{k-1}, y_{k-1})) & 0 < k < n, \\ f_0(0, 0, g_1(n-1, x_{n-1}, y_{n-1})) & k = n \end{cases}$$

$$g_1(k, x_k, y_k) = \begin{cases} f_1(x_k, y_k, 0) & k = 0, \\ f_1(x_k, y_k, g_1(k-1, x_{k-1}, y_{k-1})) & 0 < k < n \end{cases}$$

The following holds

For each $i \leq n$, $z_i = g_0(i, x_i, y_i)$

g_1 is going to take long to compute

Extras

In the following we deal with arbitrary boolean algebras

Field of Sets extra

Definition

The structure $\langle P, \emptyset, X, \cup, \cap, \backslash \rangle$, where $P \subseteq \mathcal{P}(X)$, is a field of sets if the following hold for each $x, y \in P$: $\emptyset \in P$, $x \cup y \in P$, $x \cap y \in P$, and $X \setminus x \in P$. (By $\backslash$ we mean the operation $X \setminus x$.)

extra I

Lemma

The structure $\langle P, \emptyset, X, \cup, \cap, \setminus \rangle$, where $P \subseteq \mathcal{P}(X)$ is a field of sets, is a boolean algebra.

extra II

Proof.

There is really nothing to prove here assuming one understands the meaning of the set operations $\cup$, $\cap$ and $\setminus$.

1. $x \cup \emptyset = x$ and $x \cap X = x$.
2. $x \cup y = y \cup x$ and $x \cap y = y \cap x$.
3. $(x \cup y) \cup z = x \cup (y \cup z)$ and $(x \cap y) \cap z = x \cap (y \cap z)$.
4. $x \cup (y \cap z) = (x \cup y) \cap (x \cup z)$ and $x \cap (y \cup z) = (x \cap y) \cup (x \cap z)$.
5. $x \cup (X \setminus x) = X$ and $x \cap (A \setminus x) = \emptyset$.

$\square$

Corollary

For each set X, the structure $\langle \mathcal{P}(X), \emptyset, X, \cup, \cap, \setminus \rangle$ is a boolean algebra.

Taking X to be the empty set in the above corollary we have $\mathcal{P}(X) = \{\emptyset, \{\emptyset\}\}$! Thus we got the 2-valued boolean algebra!! This might lead us to suspect (correctly!) that somehow every boolean algebra can be realized as a field of sets with the usual set operations!

Theorem (Stone representation theorem [?])

Each boolean algebra is isomorphic to a field of sets.

Definition

Let $\mathbb{B}$ be a boolean algebra.

1. We will say that $x \leq y$ if $x = xy$.
2. A subset $U \subsetneq \mathbb{B}$ is called an ultrafilter if the following hold:
    2.1 $0 \notin U$ and $1 \in U$.
    2.2 If $x \in U$ and $x \leq y$ then $y \in U$.
    2.3 If $x, y \in U$ then $x \cdot y \in U$.
    2.4 If $x \in B$ then either $x \in U$ or $\bar{x} \in U$.

Proof.

Let $\mathbb{B}$ be a boolean algebra. Let $X = \{U \mid U$ is an ultrafilter over $\mathbb{B}\}$. For each $b \in \mathbb{B}$ let $\mathcal{U}_b = \{U \in X \mid b \in U\}$. Let $P = \{\mathcal{U}_b \mid b \in \mathbb{B}\}$. Then $P \subseteq \mathcal{P}(X)$ and $\langle P, \emptyset, X, \cup, \cap, \setminus \rangle$ is a field of sets.
Define the function $\pi : \mathbb{B} \to P$ by letting $\pi(b) = \mathcal{U}_b$ for each $b \in \mathbb{B}$. It is not hard to check that

$$\pi : \langle \mathbb{B}, 0, 1, +, \cdot, \bar{\ } \rangle \to \langle P, \emptyset, X, \cup, \cap, \setminus \rangle$$

is an isomorphism. □