

Let's us Design a Logic Circuit...

REVIEW

Design a logic circuit

1. Implement with basic logic gates
2. Implement with Decoder + Gates (**ROM**)

Example; Design a Logic Circuit

- Design a binary logic circuit, having two inputs. The binary output $f(X)$, follows the rule: If the input $f(A)$, is less or equal to 1. Then: $f(X) = f(A) + 2_{10}$. Otherwise: $f(X) = f(A) - 1$.

Set Up the truth table

Example

- Design a binary logic circuit, having two inputs. The binary output $f(X)$, follows the rule: If the input $f(A)$, is less or equal to 1. Then: $f(X) = f(A) + 2_{10}$. Otherwise: $f(X) = f(A) - 1$.

	$f(A)$		$f(X)$	
	A_1	A_0	X_1	X_0
0	0	0		
1	0	1		
2	1	0		
3	1	1		

Truth table

- Design a binary logic circuit, having two inputs. The binary output $f(X)$, follows the rule: If the input $f(A)$, is less or equal to 1. Then: $f(X) = f(A) + 2_{10}$. Otherwise: $f(X) = f(A) - 1$.

	$f(A)$		$f(X)$	
	A_1	A_0	X_1	X_0
0	0	0	1	0
1	0	1	1	1
2	1	0	0	1
3	1	1	1	0

$n=2$: Inputs
 $m=2$: Outputs

Logic equations?

Logic equations

$X_1 =$

$X_0 =$

A_1	A_0	X_1	X_0
0	0	1	0
0	1	1	1
1	0	0	1
1	1	1	0

Logic equations

$$X_1 = \bar{A}_1\bar{A}_0 + \bar{A}_1A_0 + A_1A_0$$

$$X_0 = \bar{A}_1A_0 + A_1\bar{A}_0$$

A ₁	A ₀	X ₁	X ₀
0	0	1	0
0	1	1	1
1	0	0	1
1	1	1	0

Implement X₀ and X₁ using:

1. Basic logic gates
2. Decoder + OR gates

Next Implement with basic logic gates

Using basic gates (Must USE K-Maps)

	X ₁	
	\bar{A}_0	A_0
\bar{A}_1	1	1
A_1	0	1

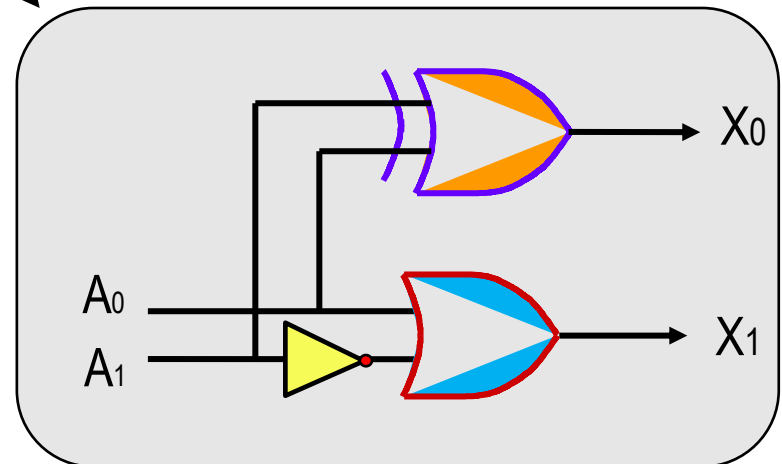
$$X_1 = \bar{A}_1 + A_0$$

$$X_0 = A_1 \text{ XOR } A_0$$

$$X_1 = \bar{\bar{A}}_1\bar{\bar{A}}_0 + \bar{\bar{A}}_1A_0 + A_1A_0$$

$$X_0 = \bar{A}_1A_0 + A_1\bar{A}_0$$

A ₁	A ₀	X ₁	X ₀
0	0	1	0
0	1	1	1
1	0	0	1
1	1	1	0



$$X_1 = \bar{A}_1\bar{A}_0 + \bar{A}_1A_0 + A_1A_0$$

$$X_0 = \bar{A}_1A_0 + A_1\bar{A}_0$$

Next Implement with Decoder + Gates

OR gates and Decoder

$$X_1 = \bar{A}_1\bar{A}_0 + \bar{A}_1A_0 + A_1A_0$$

$$X_0 = \bar{A}_1A_0 + A_1\bar{A}_0$$

- Size of the Decoder: ?
- Number of OR gates: ?

Decoder 2-4 and 2 OR gates

- Since we have (n) 2 inputs ...
- ... the size of the Decoder is $(n-2^n)$: 2-4
- Since we have (m) 2 outputs ...
- ... the number of the OR gates is (m) : 2

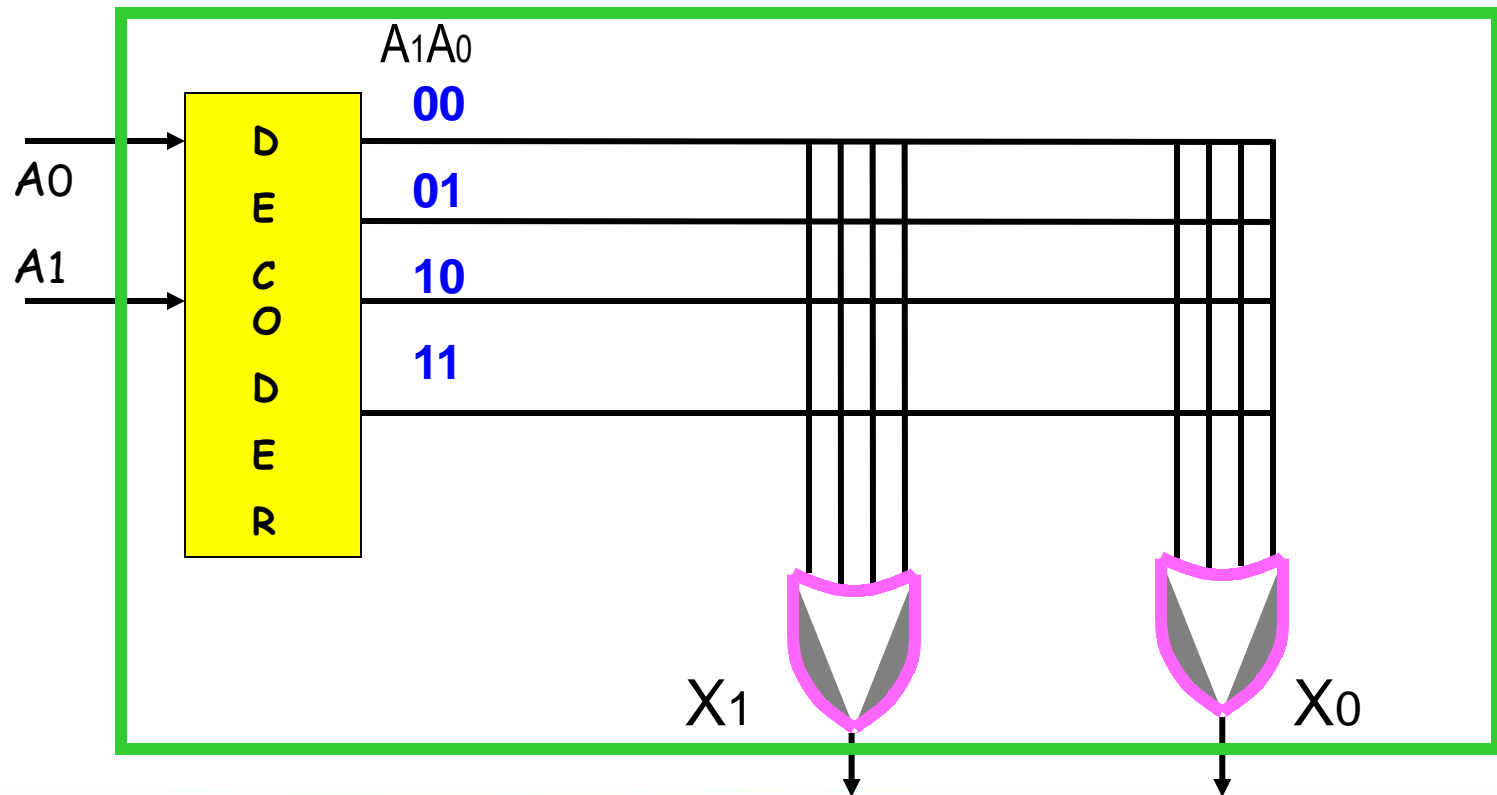
Implement with a Decoder + Gates

$$X_1 = \bar{A}_1\bar{A}_0 + \bar{A}_1A_0 + A_1A_0$$

$$X_0 = \bar{A}_1A_0 + A_1\bar{A}_0$$

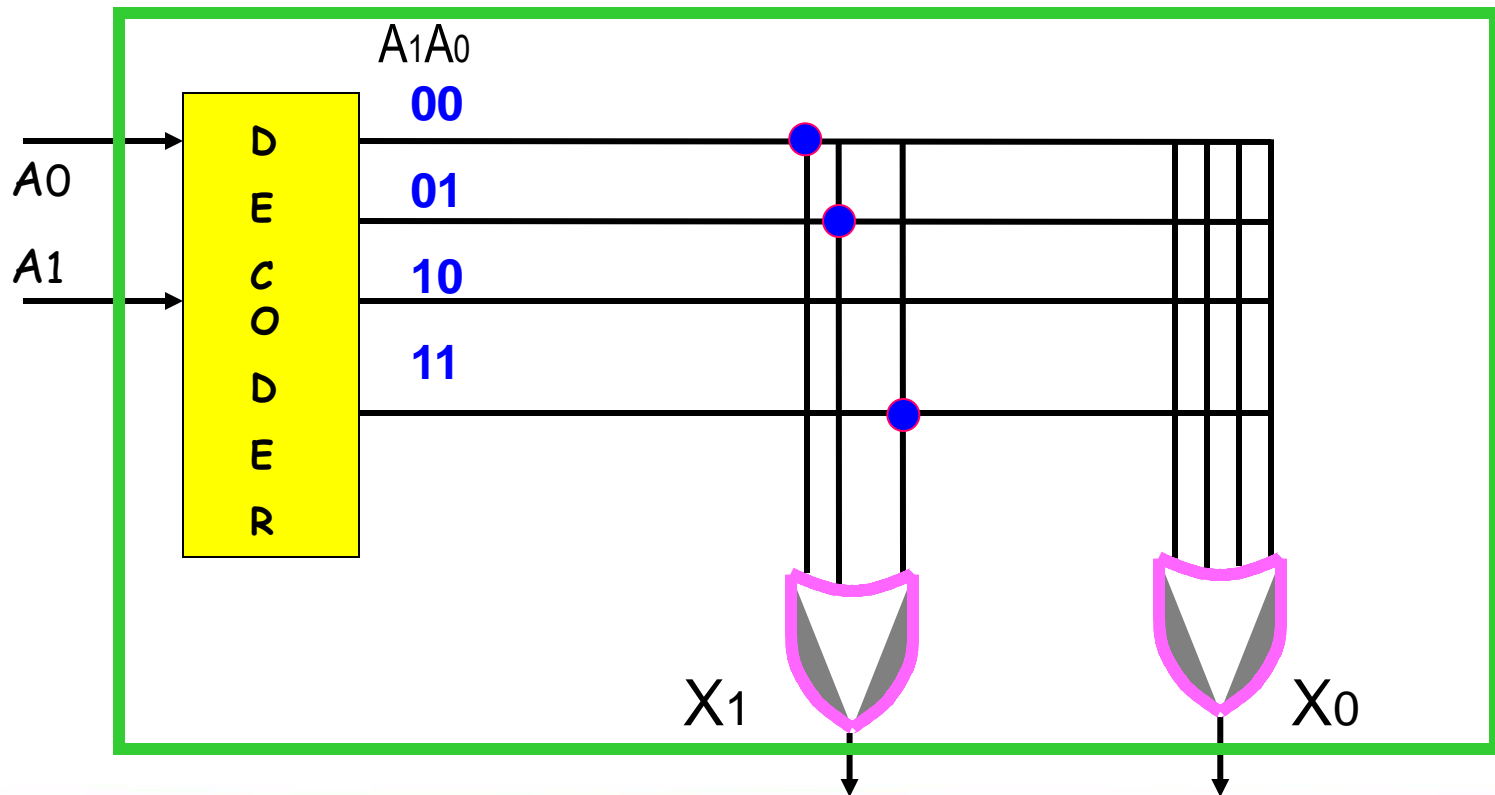
- Size of the Decoder: 2-4
- Number of OR gates: 2

Decoder + Gates (unprogrammed circuit)



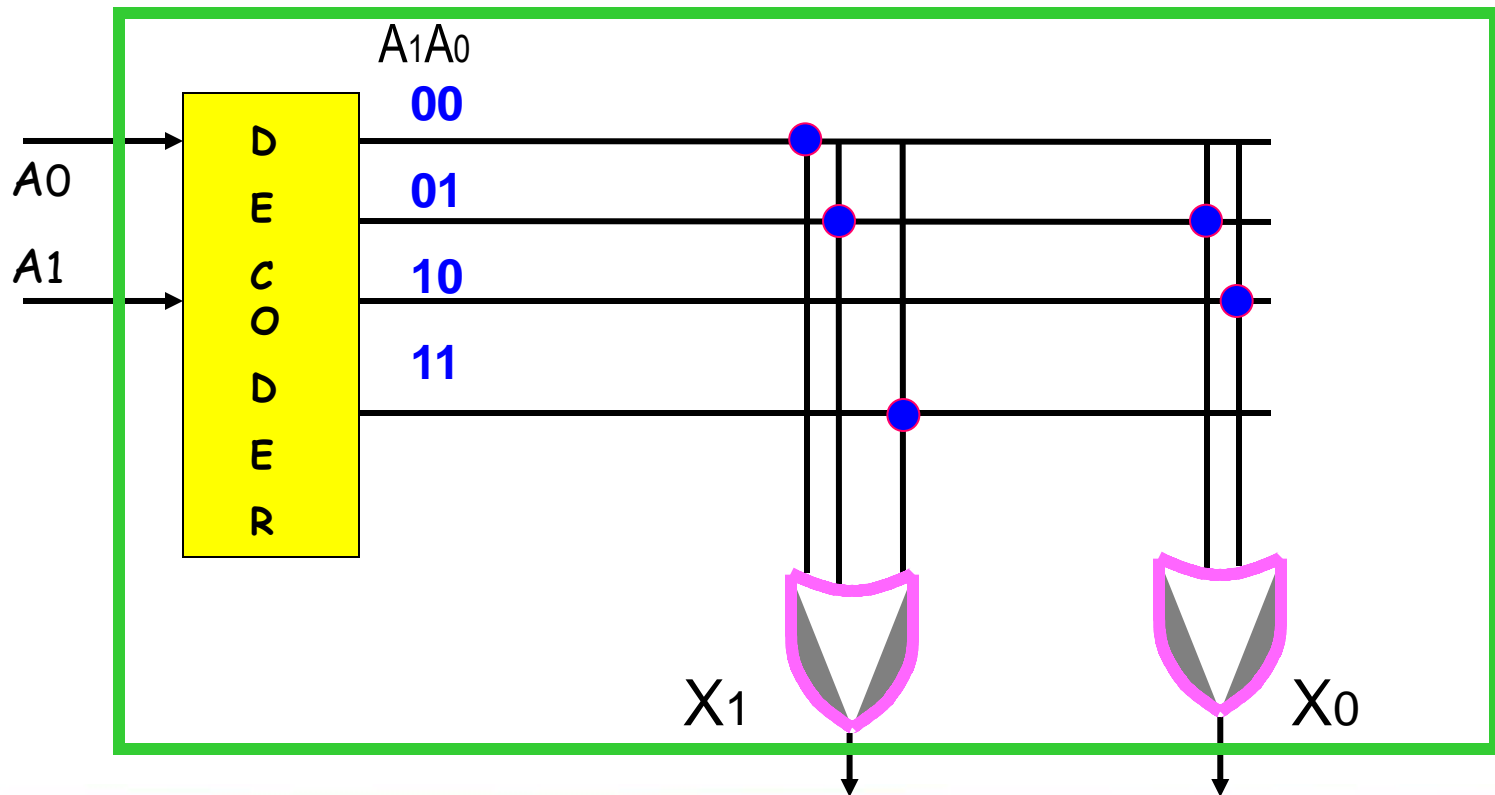
X_1

$$X_1 = \bar{A}_1\bar{A}_0 + \bar{A}_1A_0 + A_1A_0$$



X_0

$$X_0 = \bar{A}_1 A_0 + A_1 \bar{A}_0$$



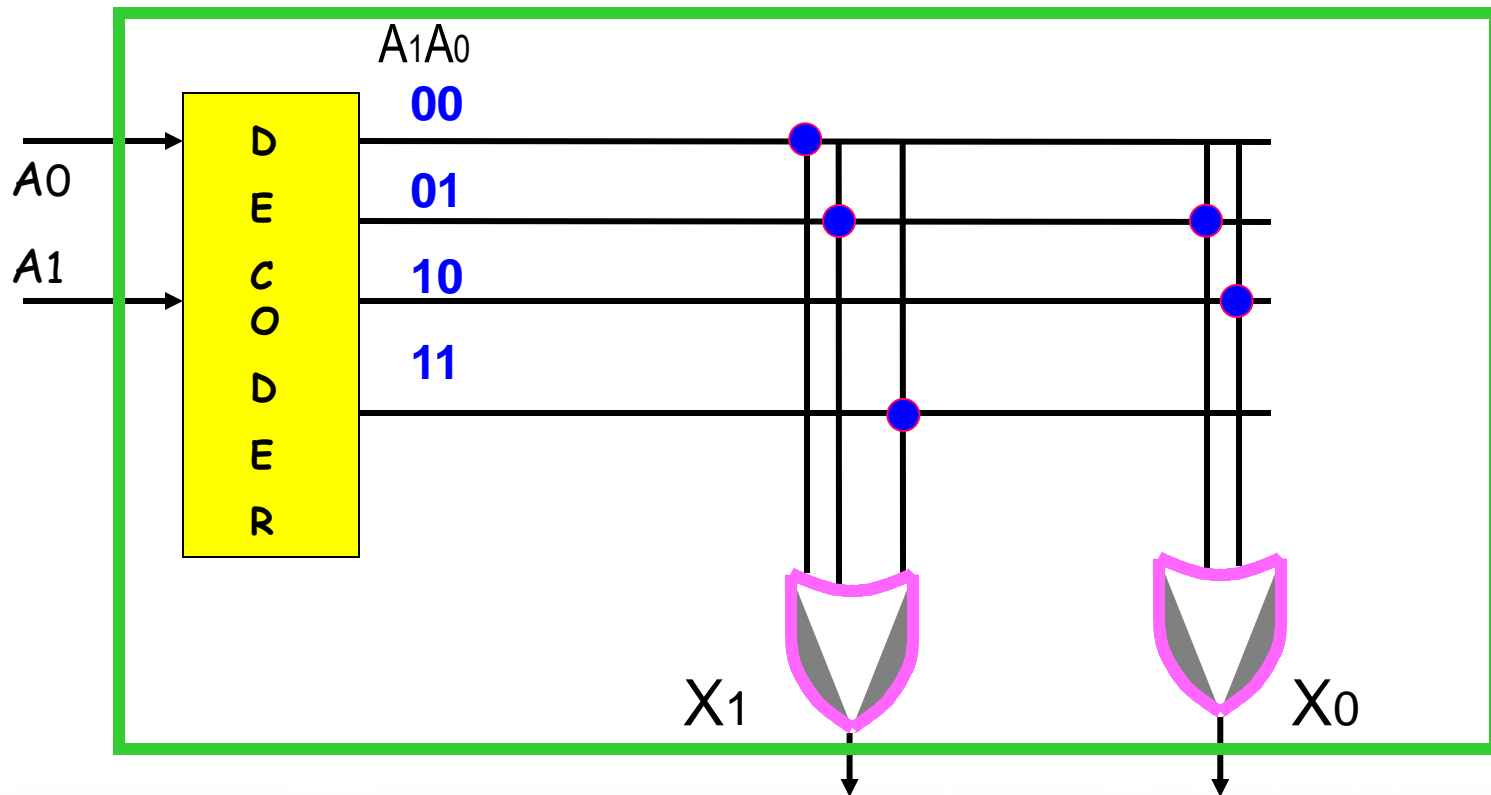
X_0 and X_1

(Programmed circuit)


$$X_1 = \bar{A}_1\bar{A}_0 + \bar{A}_1A_0 + A_1A_0$$

$$X_0 = \bar{A}_1A_0 + A_1\bar{A}_0$$

Burning



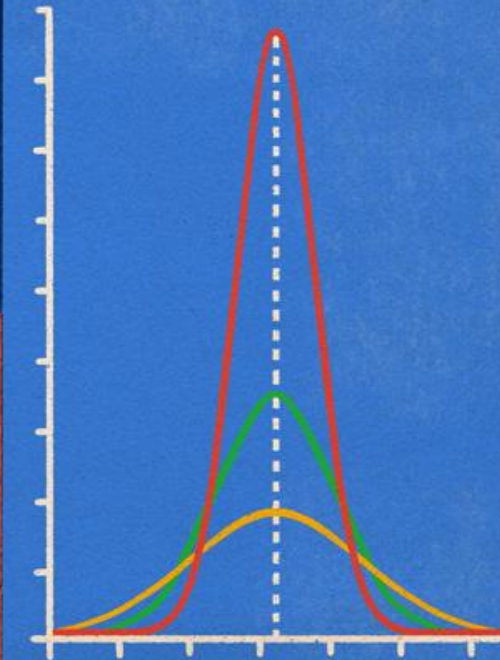
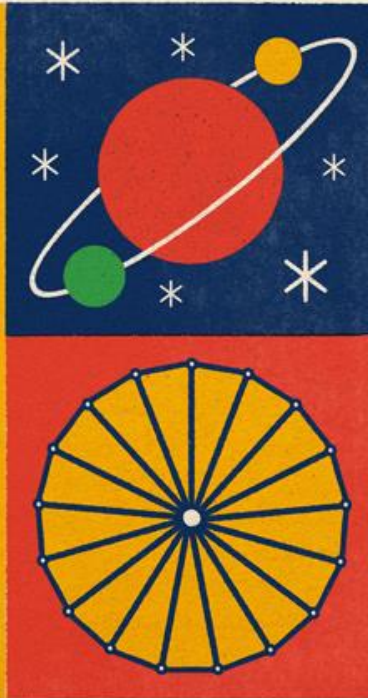
The design of a logic circuit using a Decoder + OR gates **is the main idea for the design of Read Only Memories (ROM).**

The letters 'ROM' are displayed in a large, bold, blue font. They are centered within a light green rectangular box that has a thin black border. This box is itself centered within a larger, slightly darker green rectangular background.

ROM

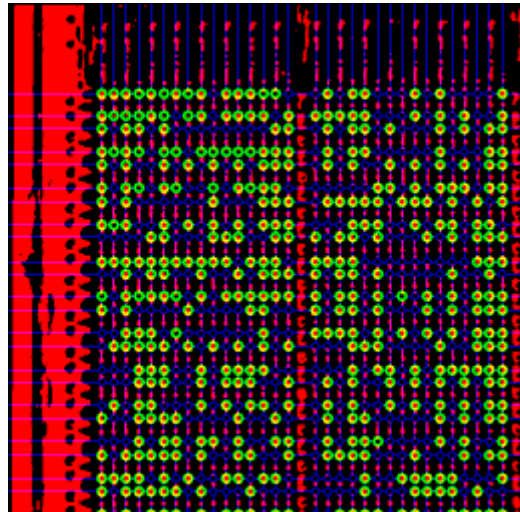
A solid green horizontal bar with a slight 3D effect and a shadow. It contains the text 'Read Only Memory' in white, bold, sans-serif font.

Read Only Memory



Read Only Memory (ROM)

- **ROM** = Electronic component
= I.C (Integrated Circuit) package
- Permanent binary information is stored
- “ROM chips contain a software that is burned onto the chip during the design phase of the semiconductor manufacturing process”



A ROM can be organized in different ways



4-bits

4 x 4 memory array



2-bits

8 x 2 memory array



8-bits

2 x 8 memory array

16-bit memories

Today ... 64-bit systems and memories

The screenshot shows the Windows 8 'System' window in the Control Panel. The window title is 'System'. The breadcrumb path is 'Control Panel > System and Security > System'. The left sidebar contains links to 'Control Panel Home', 'Device Manager', 'Remote settings', 'System protection', and 'Advanced system settings'. The main content area is titled 'View basic information about your computer' and features the Windows 8 logo. A red box highlights the 'Windows edition' section, which displays 'Windows 8 Enterprise', '© 2012 Microsoft Corporation. All rights reserved.', and the Windows Experience Index rating of '4.9'. Another red box highlights the 'System type' section, which displays '64-bit Operating System, x64-based processor'. Two red arrows point from the 'System type' box to the 'Windows edition' box. Other system information visible includes 'Processor: Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz', 'Installed memory (RAM): 6,00 GB', and 'Pen and Touch: No Pen or Touch Input is available for this Display'.

System

Control Panel > System and Security > System

Control Panel Home

View basic information about your computer

Windows edition

Windows 8 Enterprise
© 2012 Microsoft Corporation. All rights reserved.

System

Rating: 4.9 Windows Experience Index

Processor: Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz

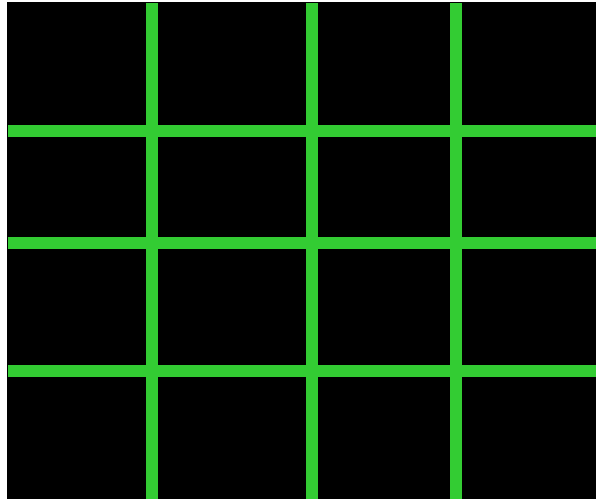
Installed memory (RAM): 6,00 GB

System type: 64-bit Operating System, x64-based processor

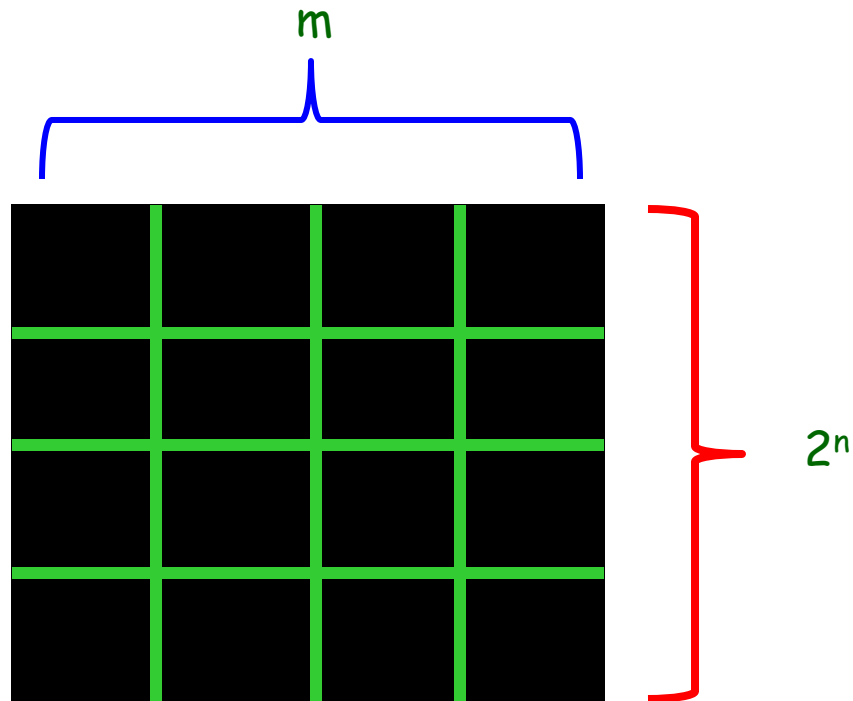
Pen and Touch: No Pen or Touch Input is available for this Display

Computer name, domain, and workgroup settings

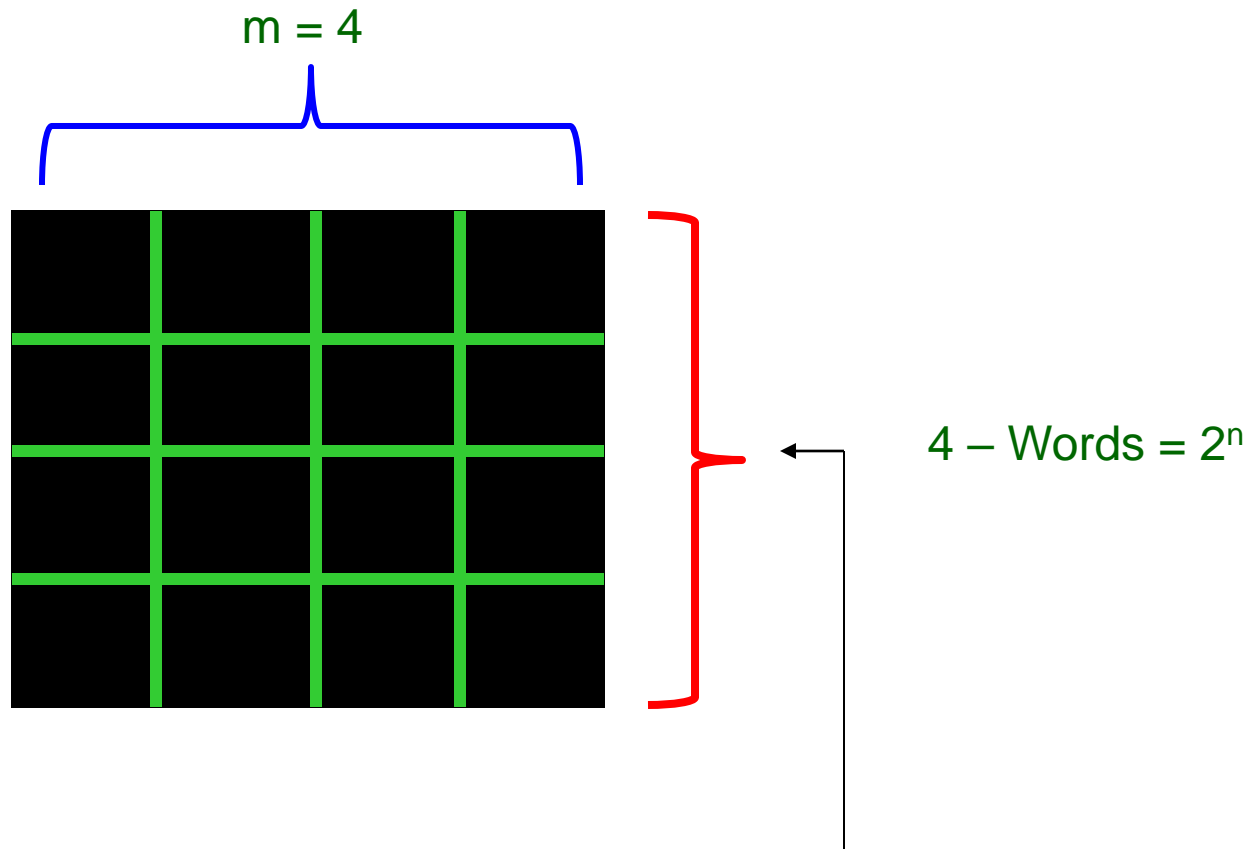
ROM



ROM capacity



ROM capacity: $2^n * m$



Total ROM capacity \longrightarrow 16 Bits x 4 memory

$$2^n * m = 4 * 4 = 16 \text{ Bits x 4 memory}$$

Capacity (size) of a ROM

- A ROM is characterized by the number of the bits

$$2^n * m$$

2^n = number of words stored in the ROM

m = number of bits per word

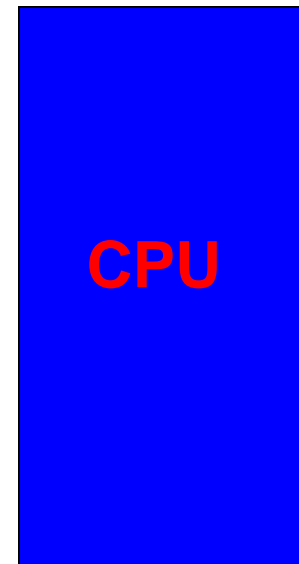
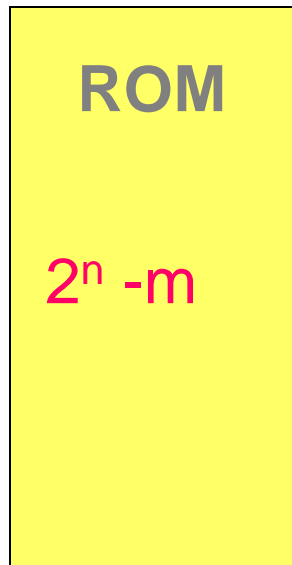
ROM: Data & Address

0				
1	1	0	0	1
2				
3				

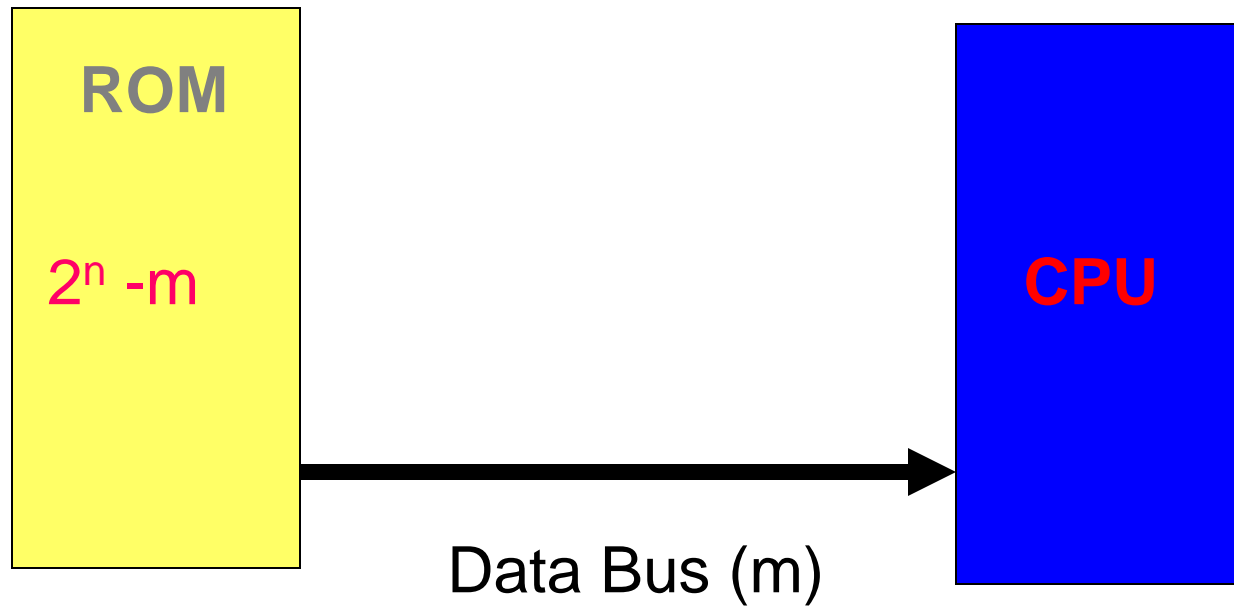


The **address** of the selected word in yellow, with **data**:
1001 is 1

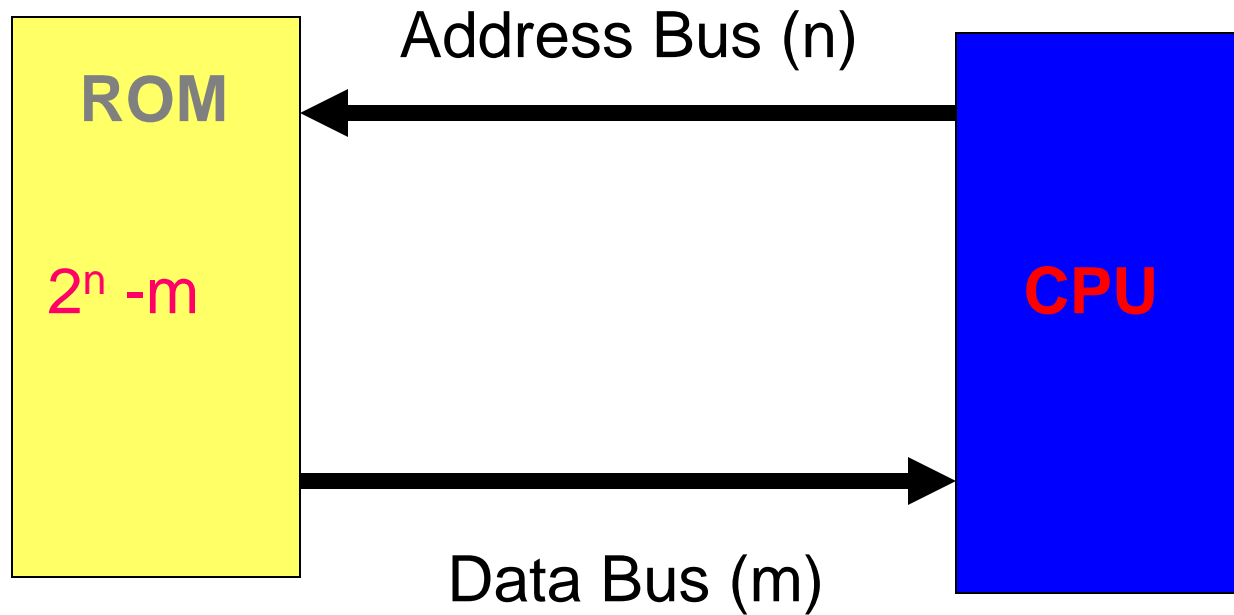
ROM and CPU



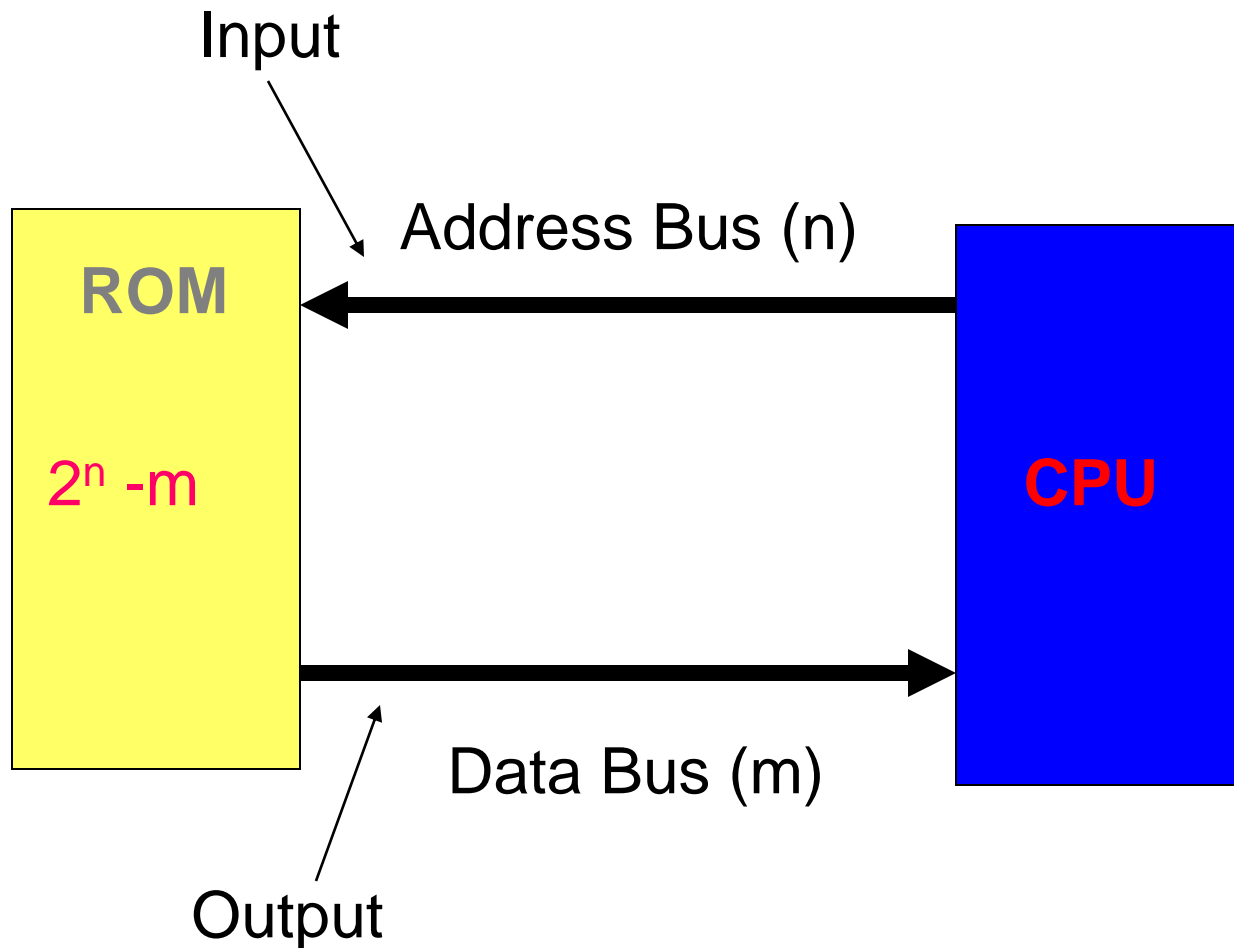
Data Bus



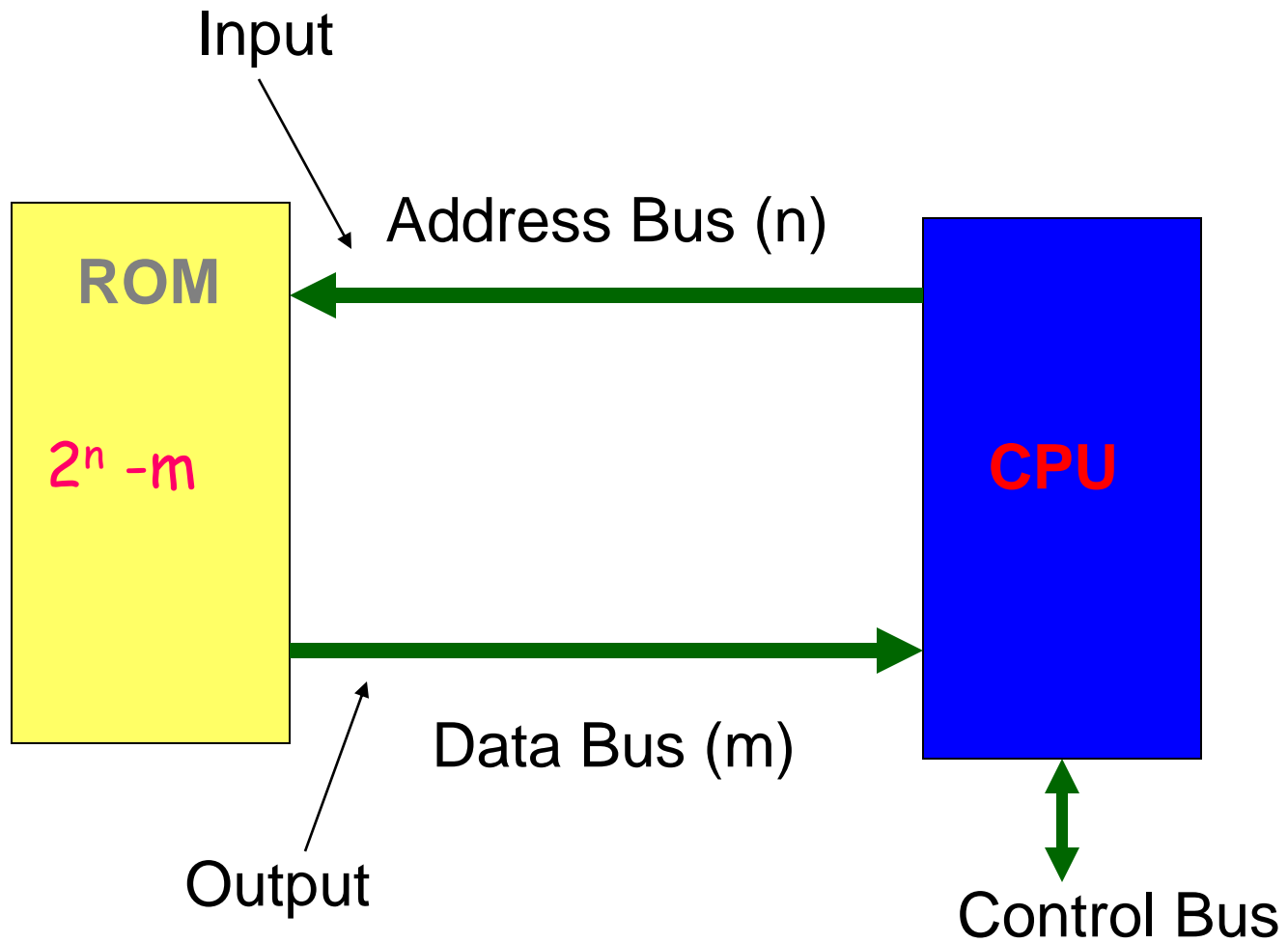
Address bus



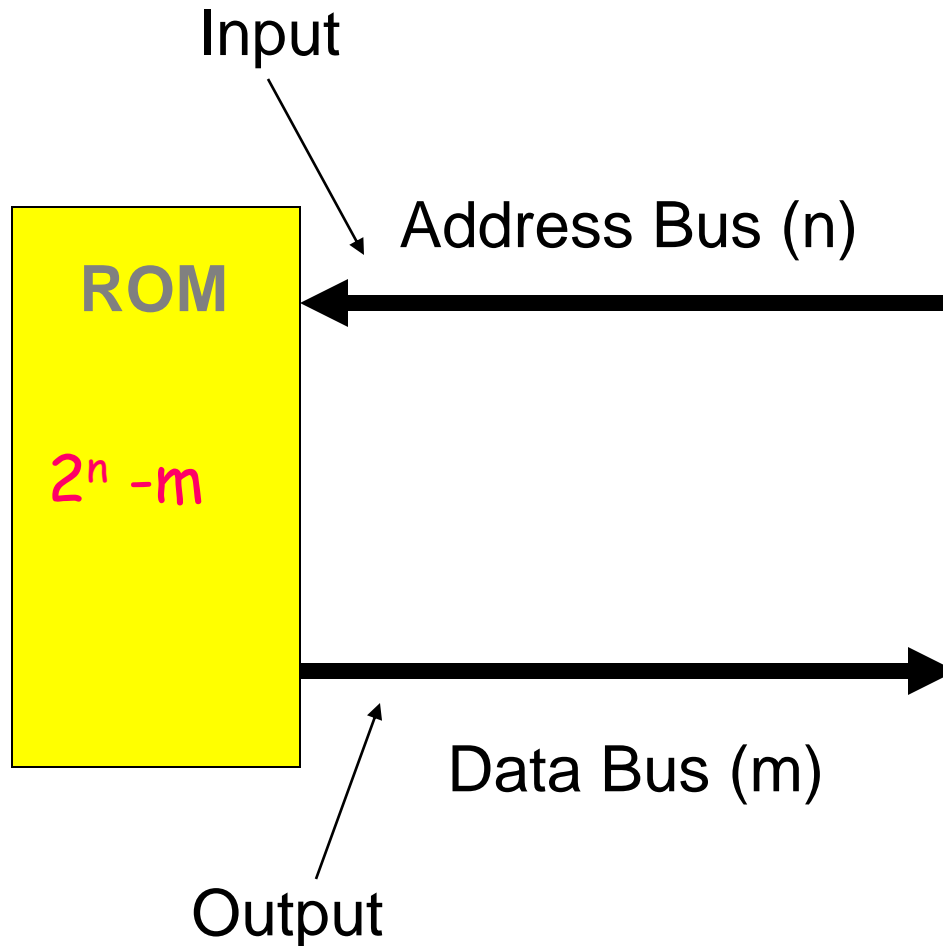
Input/Output



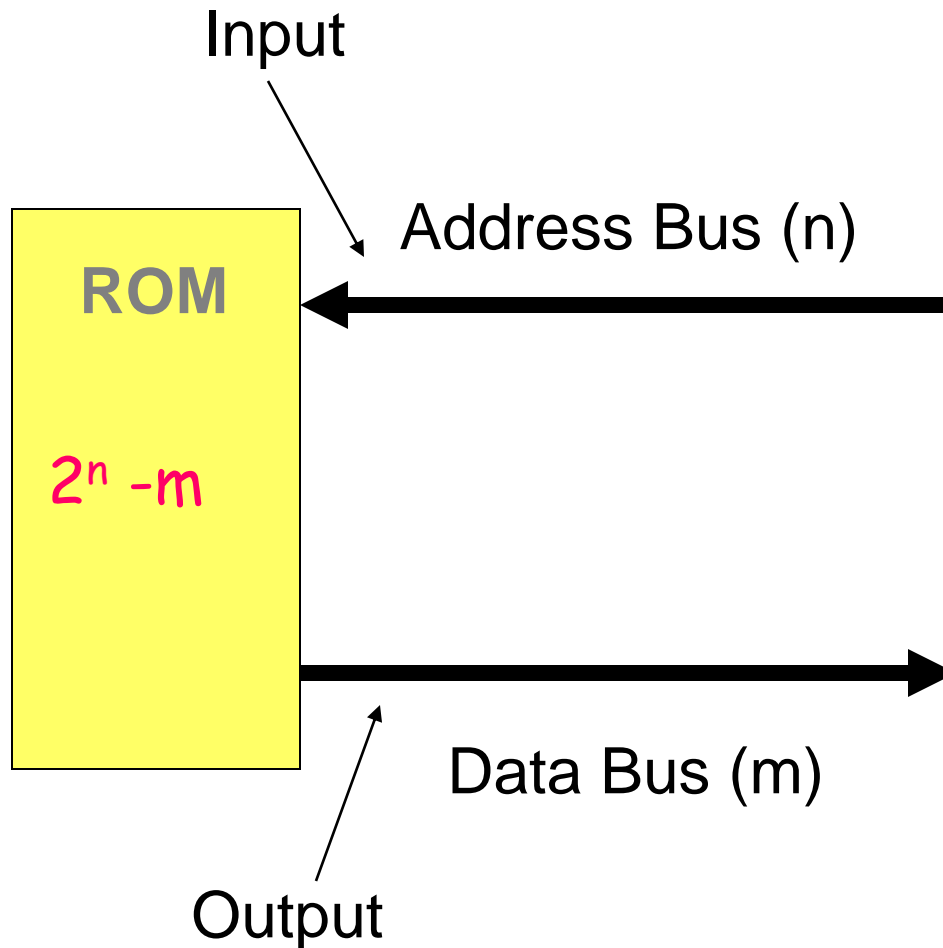
The 3 buses



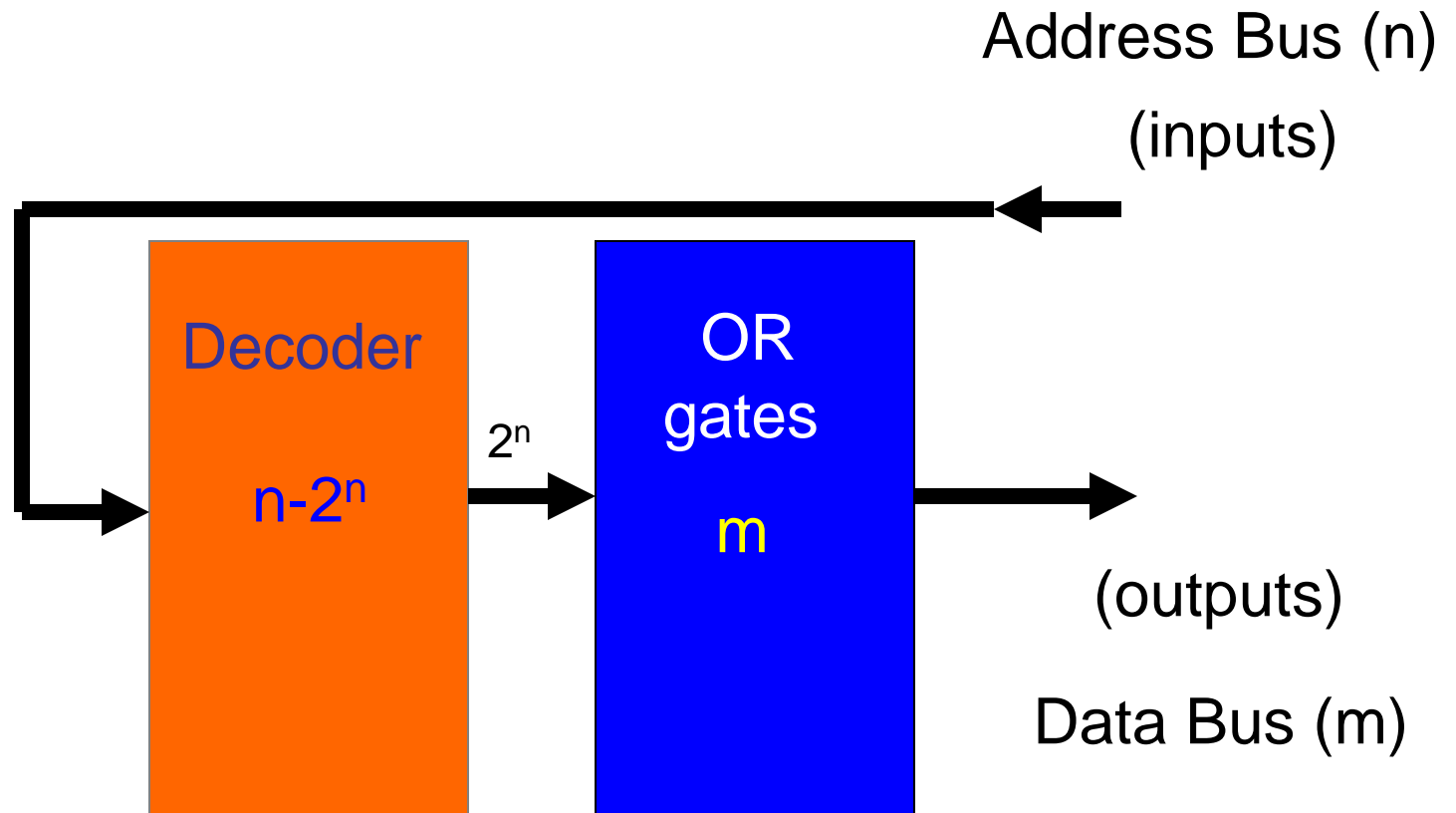
ROM (with n -inputs and m -outputs)



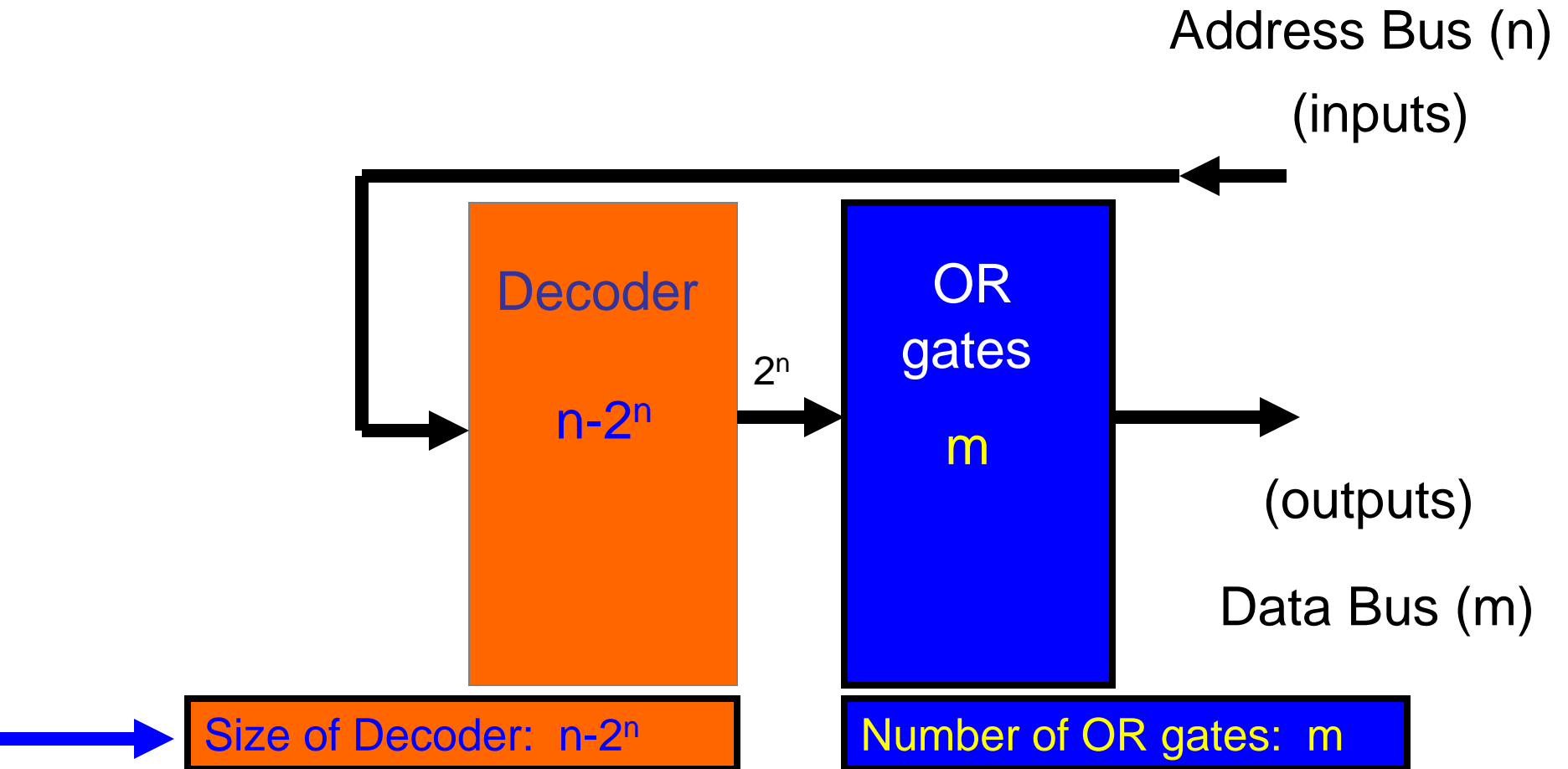
What is inside the ROM



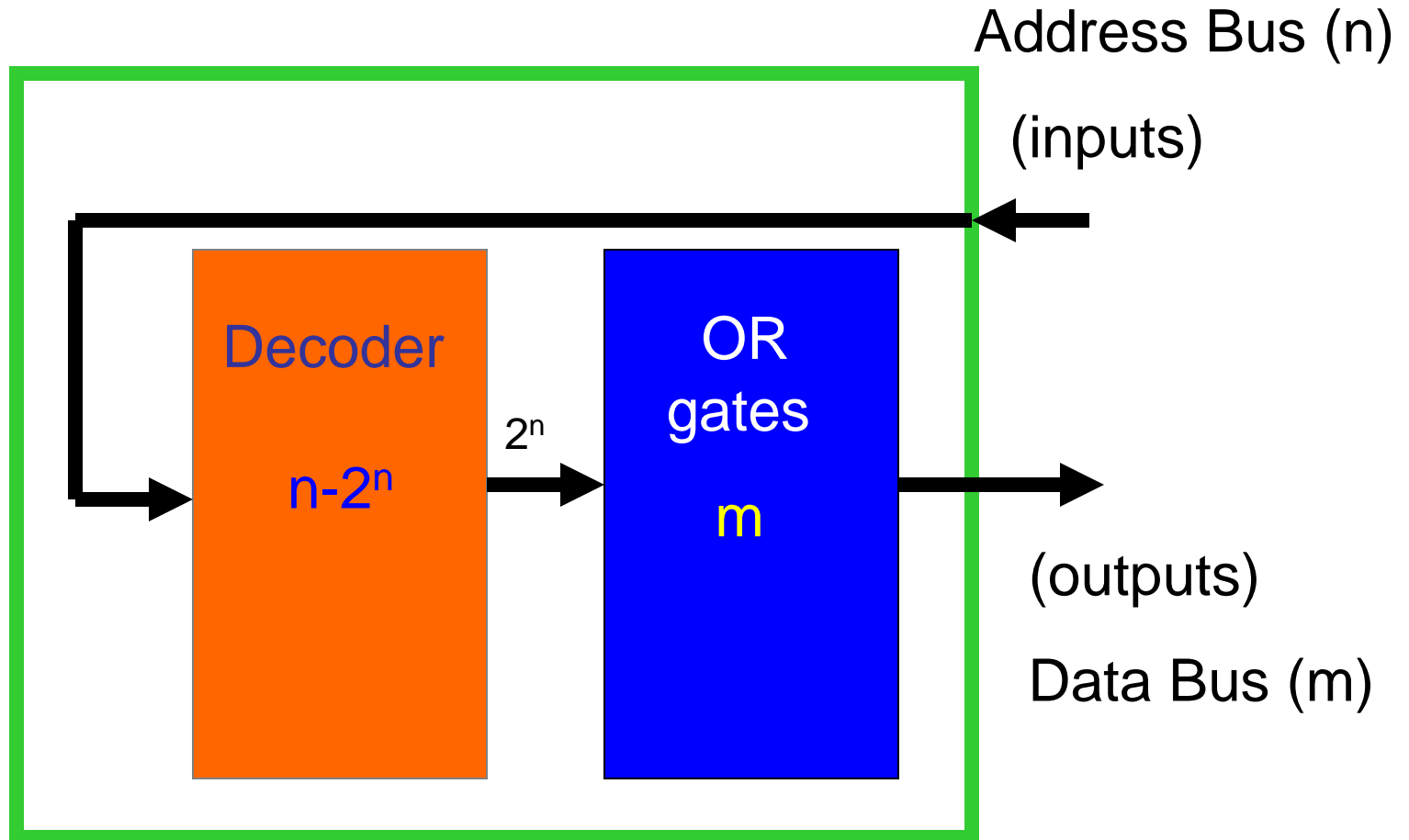
ROM = Decoder + OR gates



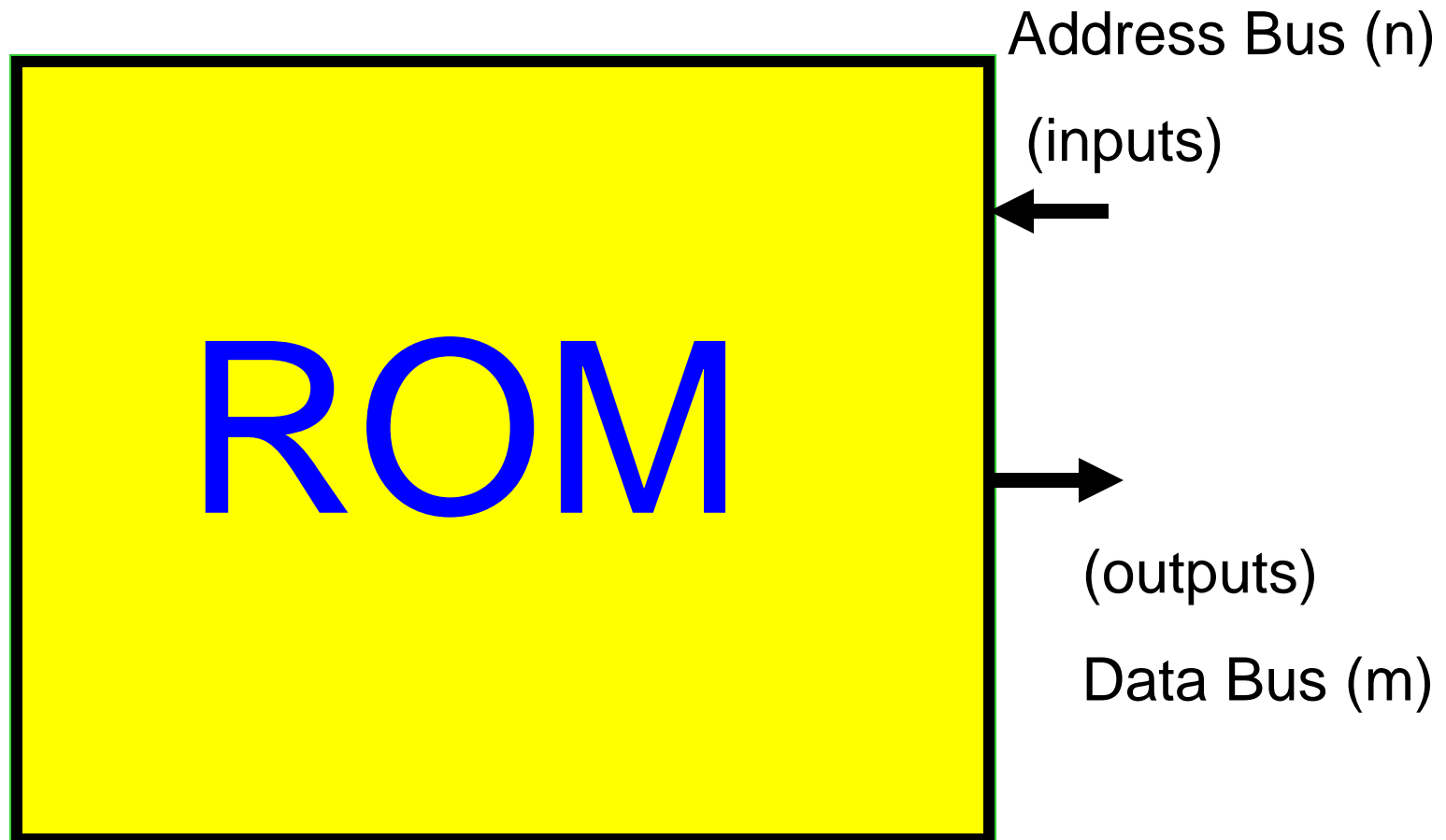
Size of ROM/Number of OR gates



ROM architecture



ROM: $(2^n) * m$



ROM Characteristic

- ROM = Realizes a truth table

... Note that a truth table can represent mathematical and logic functions.

ROM applications

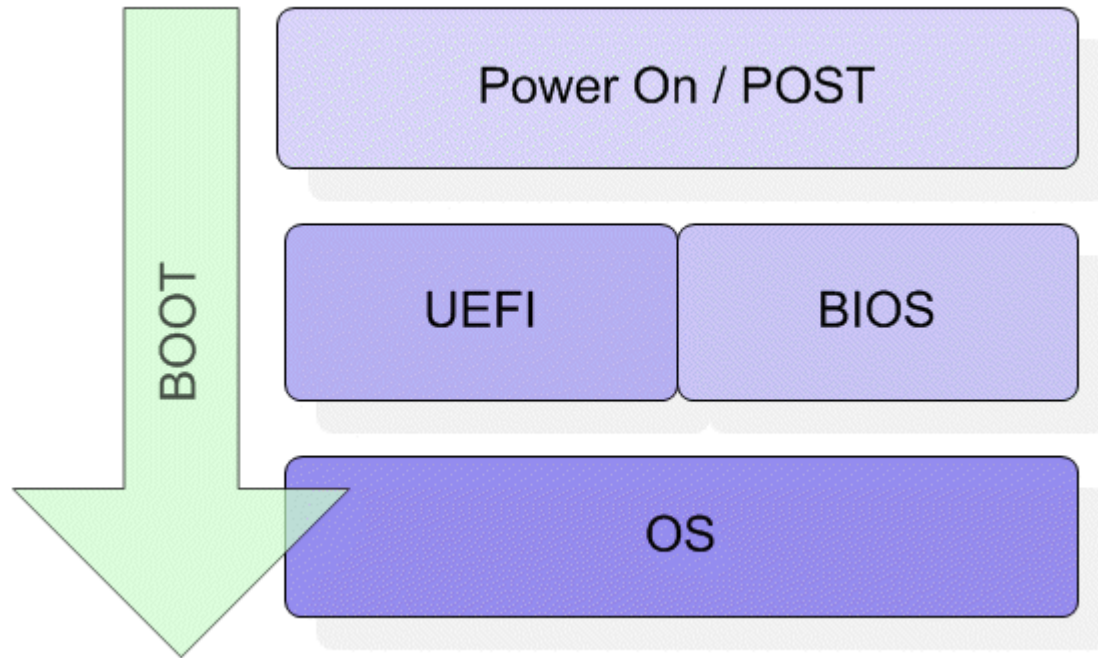
- Dynamic
- Static.

ROM (Dynamic application)

- **BIOS** (**B**asic **I**nput **O**utput **S**ystem)
- All PC's had a **BIOS** [microcode (Intel Assembly) -hardwire] chip built in ...
- «**BIOS** gets the computer started before the operating system is loaded from the hard drive».



Unified Extensible Firmware Interface (UEFI)



Today ... all Intel based CPU Computer Systems use UEFI (not BIOS)

BIOS vs UEFI

BIOS vs UEFI

#UnhappyGhost

Bootling Old Way



Bootling New Way



For more posts visit:
unhappyghost.com

[Fb.com/geeksch001](https://www.facebook.com/geeksch001)

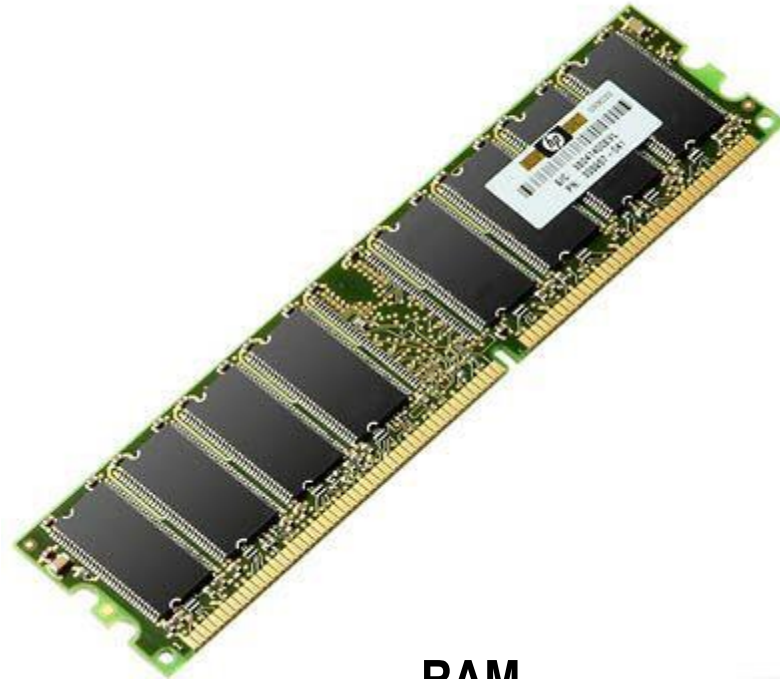
Kernel is a computer program that manages I/O (input/output) requests from software, and translates them into data processing instructions for the central processing unit and other electronic components of a computer. The kernel is a fundamental part of a modern computer's operating system. (Wikipedia.com)

ROM (Static application)

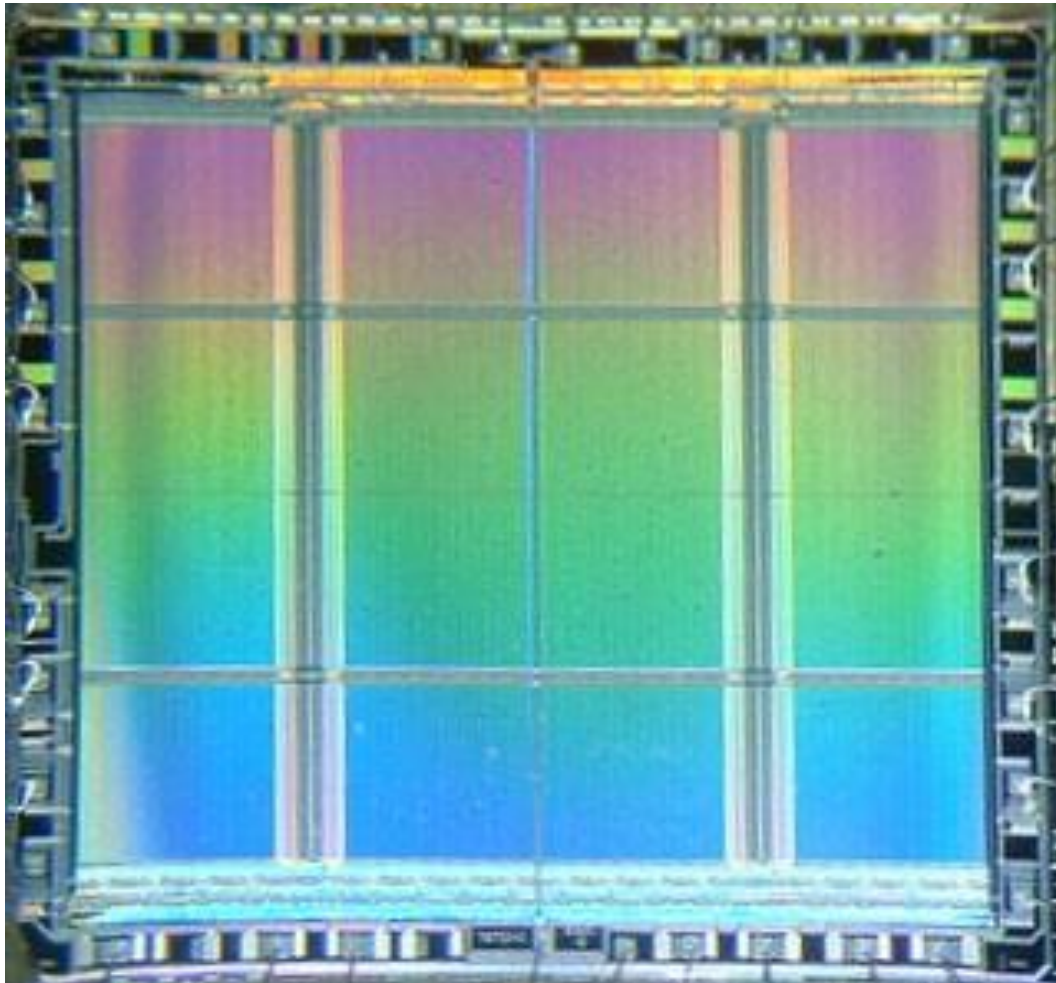
- A ROM can be used to control fonts in printers
- As table-Look-Up (data storage)
- Code converters
- Video games
- Terminals in stores
- Security systems.



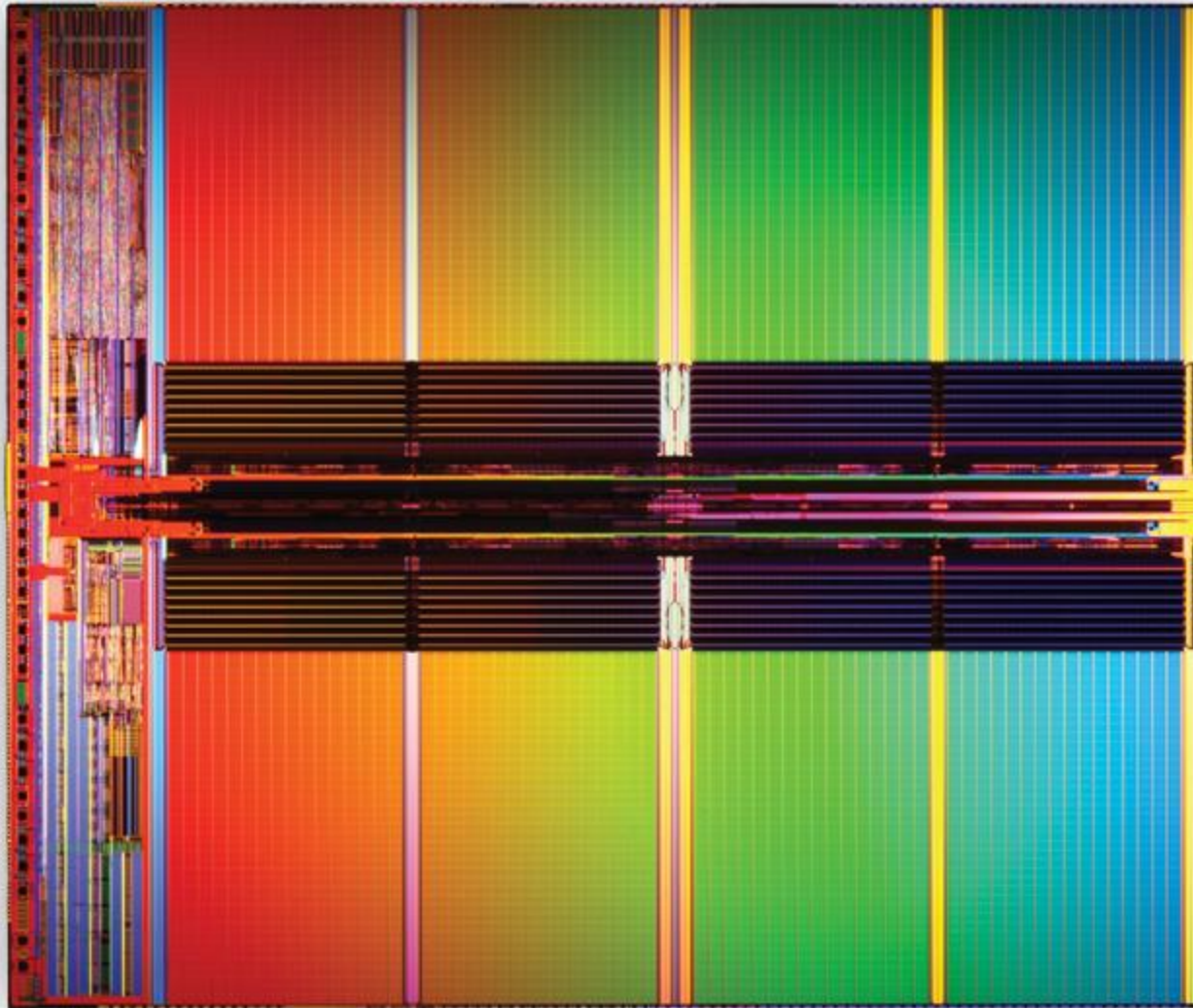
ROM



RAM



4Mbit EPROM Texas Instruments TMS27C040 (Wikipedia)



2. Micron's triple-level cell (TLC) flash memory stores 3 bits of data in each transistor.

ROM design problem

ROM design problem


Design a ROM (static) that will accept a 3-bit (binary) number and generate a binary number equal to the square of the input.

Lets create the required truth table...

How many inputs ... outputs do we need?

Truth Table

Design a ROM (static) that will accept a 3-bit (binary) number and generate a binary number equal to the square of the input.

Inputs				Outputs	
	A ₂	A ₁	A ₀		
0	0	0	0		
1	0	0	1		
2	0	1	0		
3	0	1	1		
4	1	0	0		
5	1	0	1		
6	1	1	0		
7	1	1	1		

Truth Table

Design a ROM (static) that will accept a 3-bit (binary) number and generate a binary number equal to the square of the input.

Inputs				Outputs		
	A ₂	A ₁	A ₀	B ₂	B ₁	B ₀
0	0	0	0			0
1	0	0	1			1
2	0	1	0	1	0	0
3	0	1	1			
4	1	0	0			
5	1	0	1			
6	1	1	0			
7	1	1	1			

Truth Table

Inputs				Outputs						
	A2	A1	A0	B5	B4	B3	B2	B1	B0	
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	1	0	0	4
3	0	1	1	0	0	1	0	0	1	9
4	1	0	0	0	1	0	0	0	0	16
5	1	0	1	0	1	1	0	0	1	25
6	1	1	0	1	0	0	1	0	0	36
7	1	1	1	1	1	0	0	0	1	49

Inputs ... Outputs

- Inputs = $n = 3$
- Outputs = $m = 6$

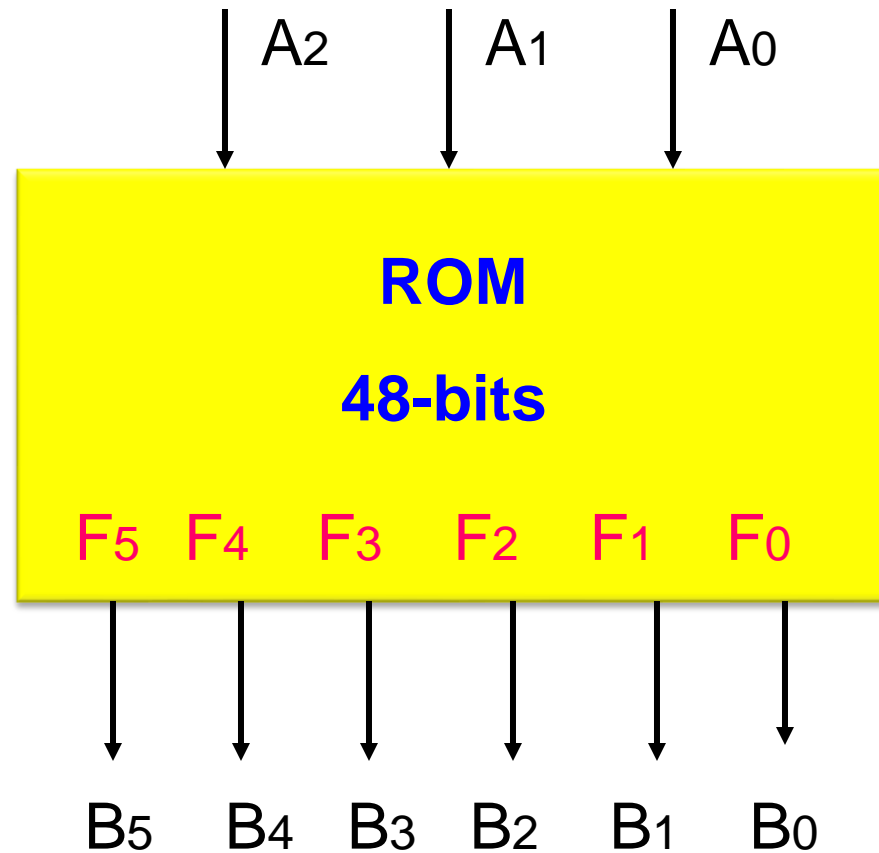
Size of ROM

- Inputs = $n = 3$
- Outputs = $m = 6$

$$2^n * m$$

$$2^n * m = 2^3 * 6 = 48\text{-bits}$$

The ROM has 48-bits



Can we simplify?



The ROM simplification, ONLY involves columns (truth table)

K-Maps or Boolean Theorems can NOT be used in ROM simplification

Eliminate Outputs [ONLY Columns]

Inputs				Outputs						
	A ₂	A ₁	A ₀	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀	
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	1	0	0	4
3	0	1	1	0	0	1	0	0	1	9
4	1	0	0	0	1	0	0	0	0	16
5	1	0	1	0	1	1	0	0	1	25
6	1	1	0	1	0	0	1	0	0	36
7	1	1	1	1	1	0	0	0	1	49

Eliminate Outputs [ONLY Columns]

Inputs			Outputs							
	A ₂	A ₁	A ₀	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀	
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1	1
2	0	1	0	0	0	0	1	0	0	4
3	0	1	1	0	0	1	0	0	1	9
4	1	0	0	0	1	0	0	0	0	16
5	1	0	1	0	1	1	0	0	1	25
6	1	1	0	1	0	0	1	0	0	36
7	1	1	1	1	1	0	0	0	1	49



Simplification equations

$$\mathbf{B_0 = A_0} \text{ and } \mathbf{B_1 = 0}$$

Our new simplified ROM: Truth Table

Inputs				Outputs			
	A ₂	A ₁	A ₀	F ₃	F ₂	F ₁	F ₀
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1
3	0	1	1	0	0	1	0
4	1	0	0	0	1	0	0
5	1	0	1	0	1	1	0
6	1	1	0	1	0	0	1
7	1	1	1	1	1	0	0

Inputs ... Outputs

- Inputs = $n = 3$
- Outputs = $m = 4$

Size of simplified ROM?

- Inputs = $n = 3$
- Outputs = $m = 4$

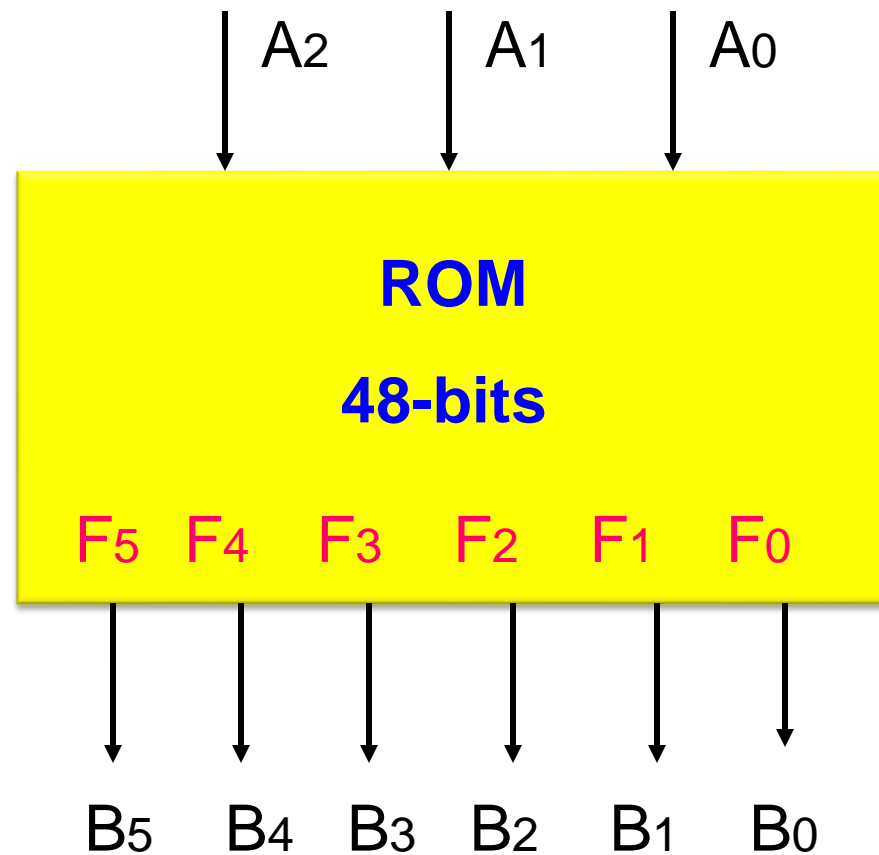
Size of simplified ROM?

- Inputs = $n = 3$
- Outputs = $m = 4$

$$\text{Size} = 2^n * m = 2^3 * 4 = 32 \text{ bits}$$

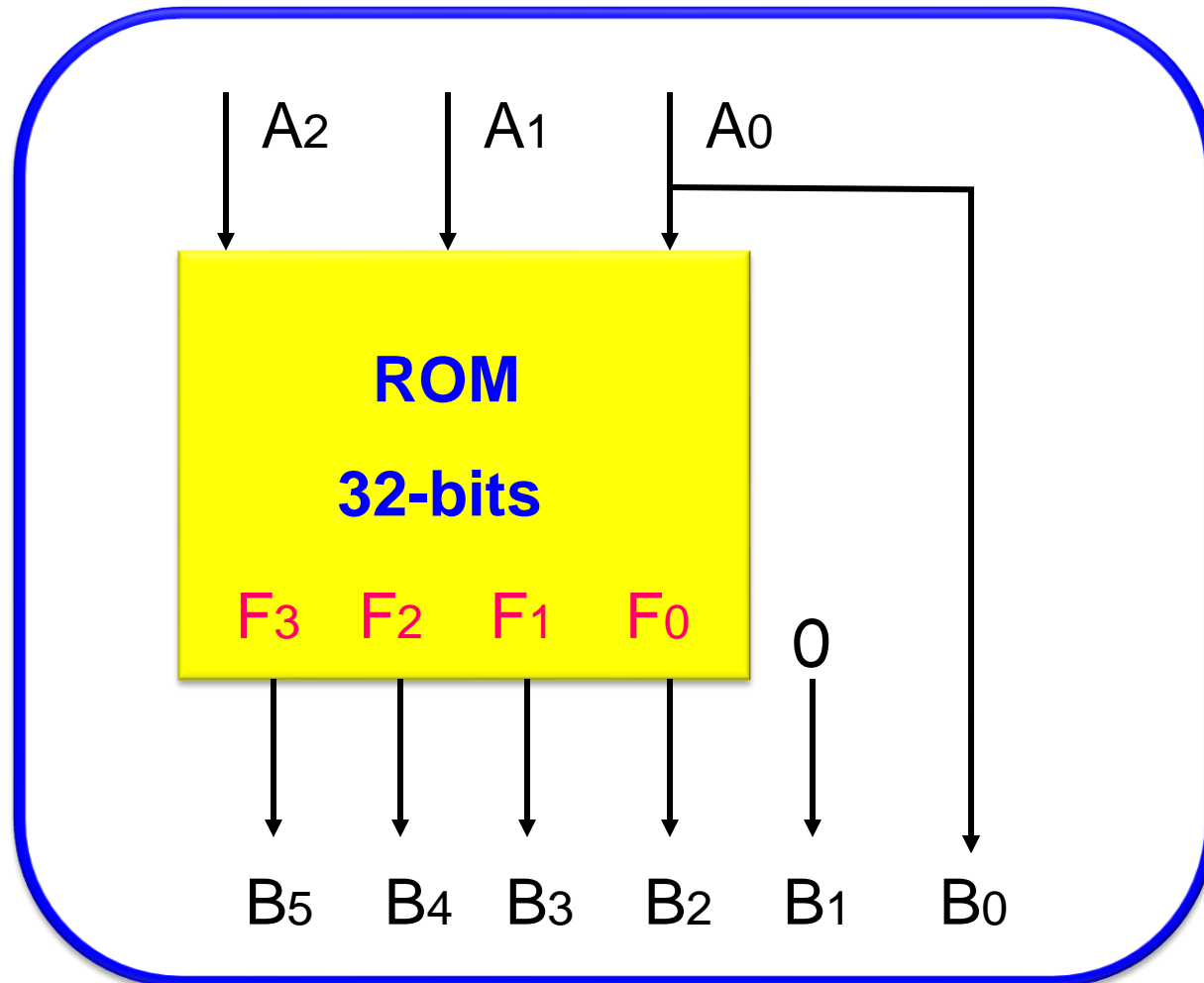
- Initially 48 bits
- Therefore we saved: 33.3%

Initial 48-bit ROM

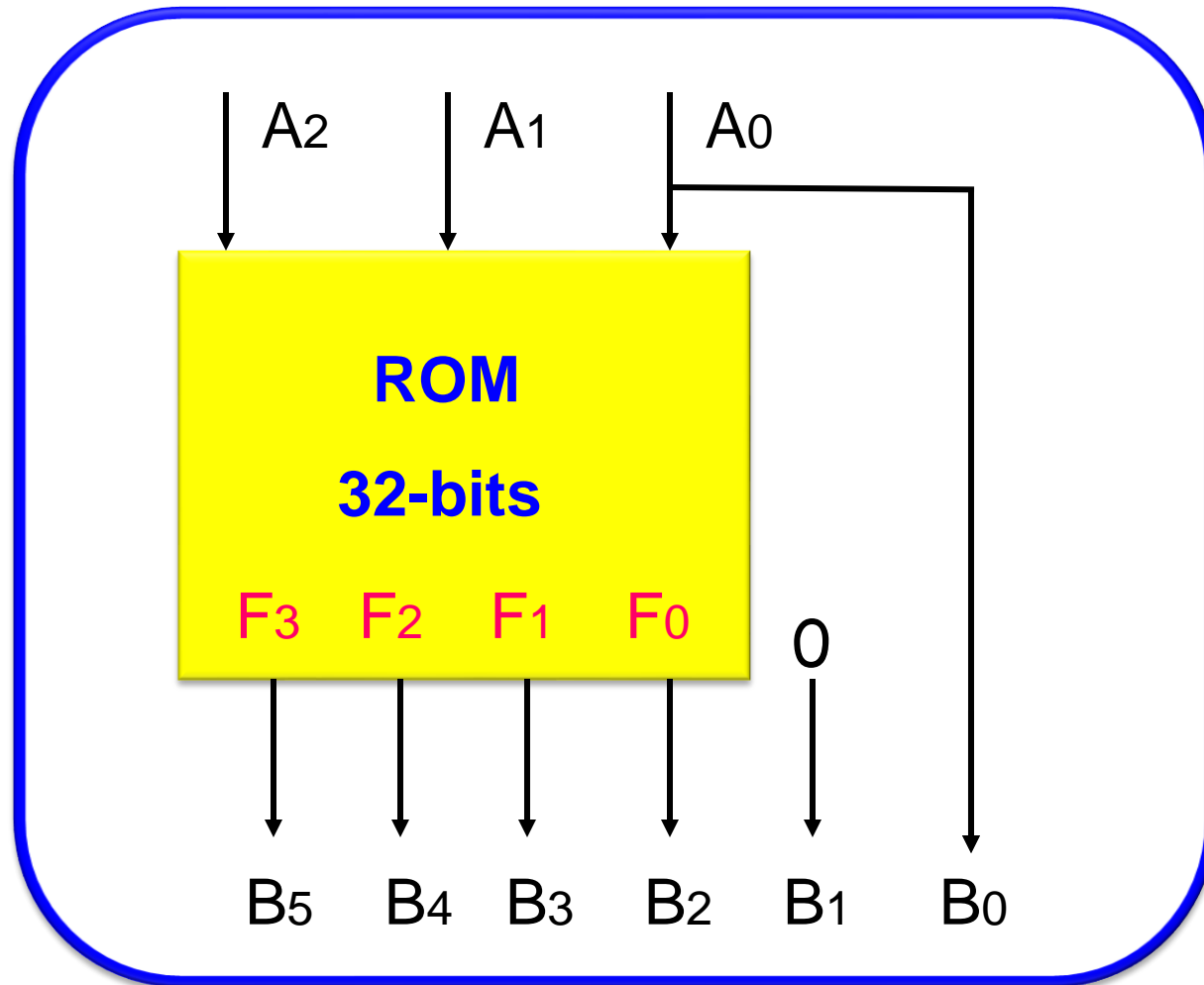


Simplified 32-bit ROM

B₀ = A₀ and **B₁ = 0**



Inside the yellow box ...



Go to the simplified truth table and derive the logic equations

Our new simplified ROM: Truth Table

Inputs				Outputs			
	A ₂	A ₁	A ₀	F ₃	F ₂	F ₁	F ₀
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1
3	0	1	1	0	0	1	0
4	1	0	0	0	1	0	0
5	1	0	1	0	1	1	0
6	1	1	0	1	0	0	1
7	1	1	1	1	1	0	0

Our new output logic equations

$$F_3 = \bar{A}_0 A_1 A_2 + A_0 A_1 A_2$$

$$F_2 = \bar{A}_0 \bar{A}_1 A_2 + A_0 \bar{A}_1 A_2 + A_0 A_1 A_2$$

$$F_1 = A_0 A_1 \bar{A}_2 + A_0 \bar{A}_1 A_2$$

$$F_0 = \bar{A}_0 A_1 \bar{A}_2 + \bar{A}_0 A_1 A_2$$

- When we design ROM's, DO NOT use K-Maps or logic theorems to simplify the above equations
- The above equations are not further simplified.

Use a Decoder and Gates to implement

- Size of the Decoder: ?
- How many OR gates: ?

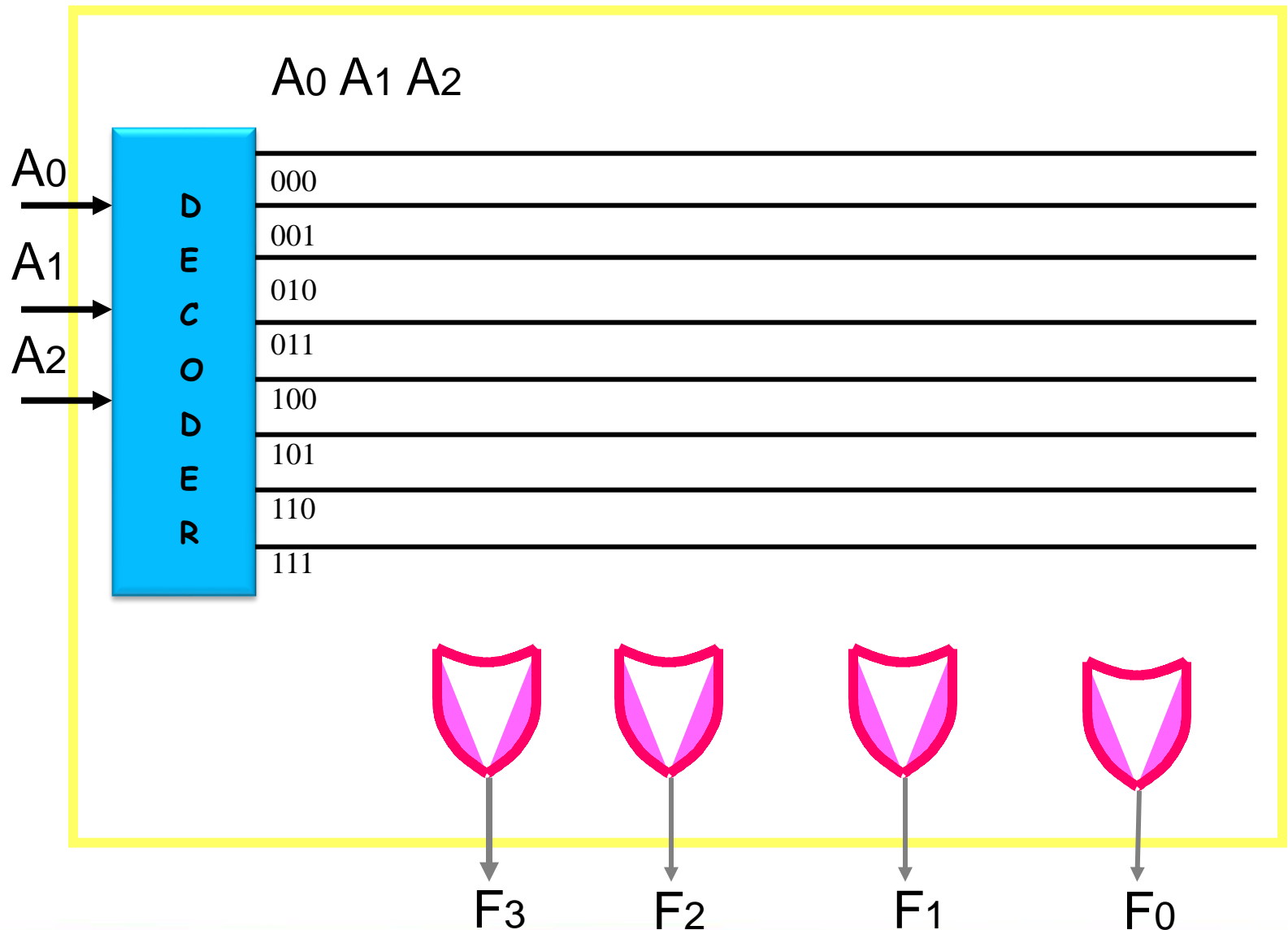
3-8 Decoder and 4 OR gates

- Since we have (n) 3 inputs...
- ... the size of the Decoder ($n-2^n$) is: 3-8
- Since we have 4 outputs ...
- ... the number of OR gates is: 4

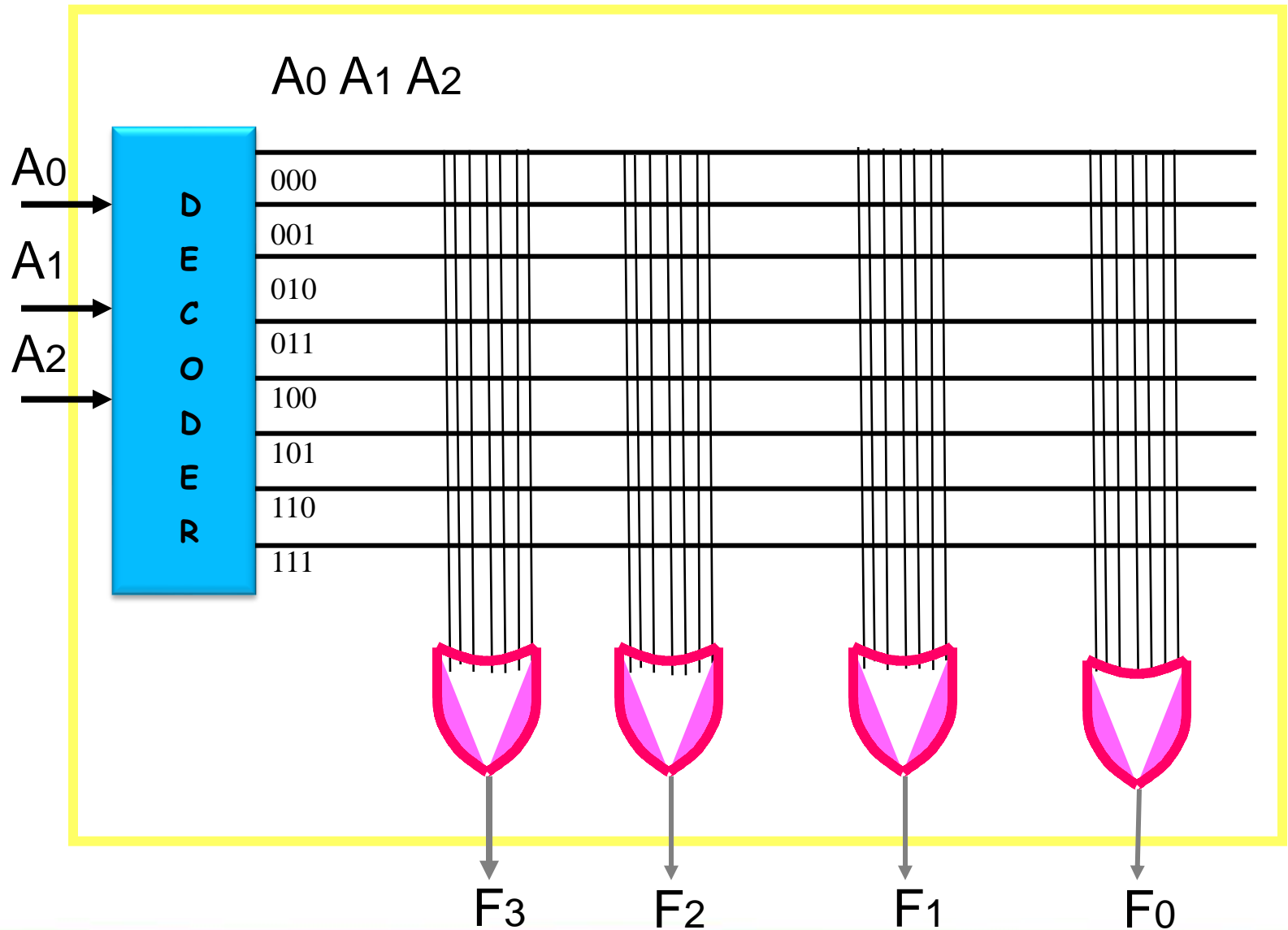
Implement using Decoder and OR-gates

- Size of the Decoder: 3-8
- How many OR gates: 4

3-8 Decoder and 4 OR gates



Un-programmed ROM



To program use our logic equations

$$F_3 = \bar{A}_0 A_1 A_2 + A_0 A_1 A_2$$

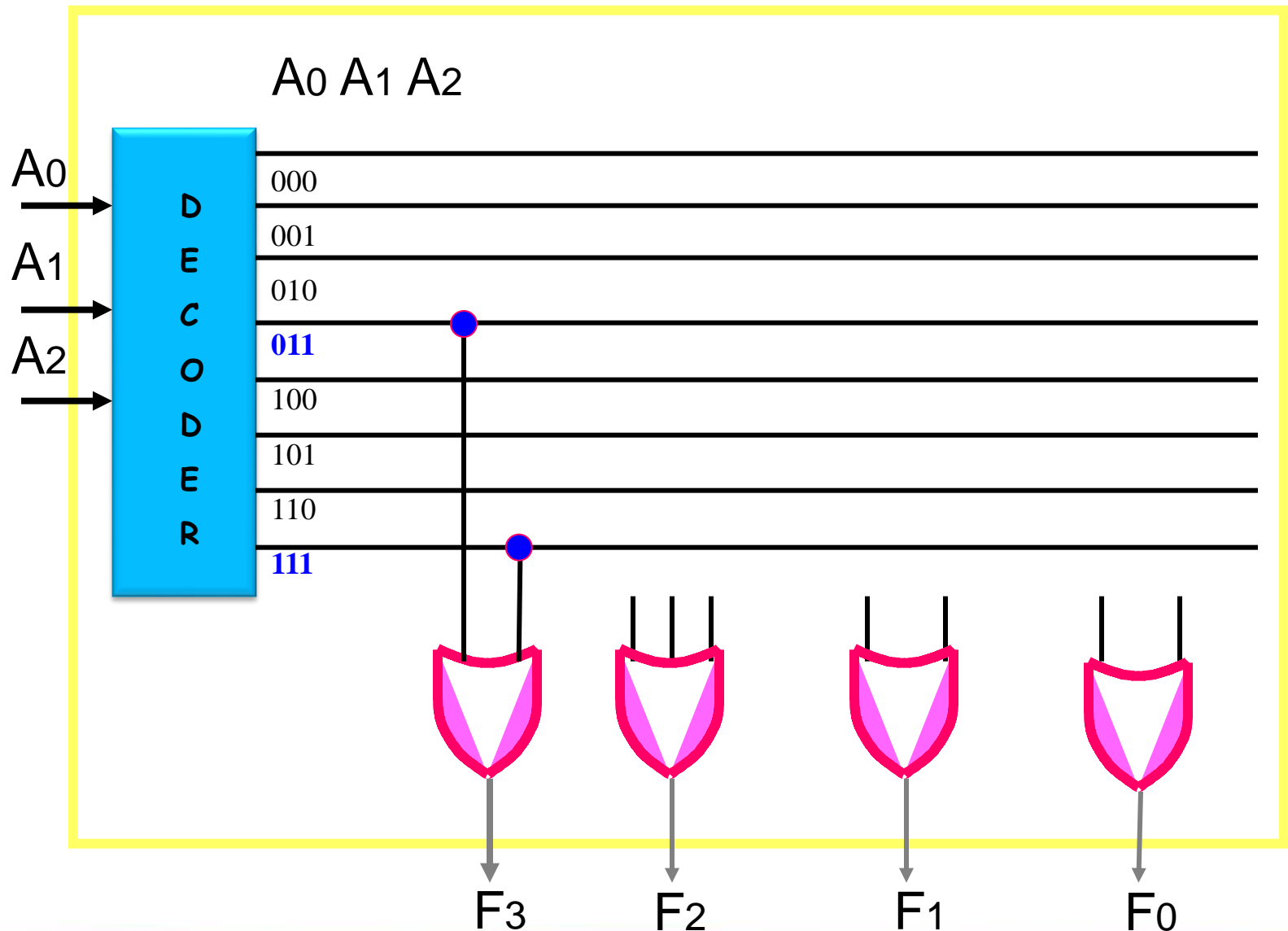
$$F_2 = \bar{A}_0 \bar{A}_1 A_2 + A_0 \bar{A}_1 A_2 + A_0 A_1 A_2$$

$$F_1 = A_0 A_1 \bar{A}_2 + A_0 \bar{A}_1 A_2$$

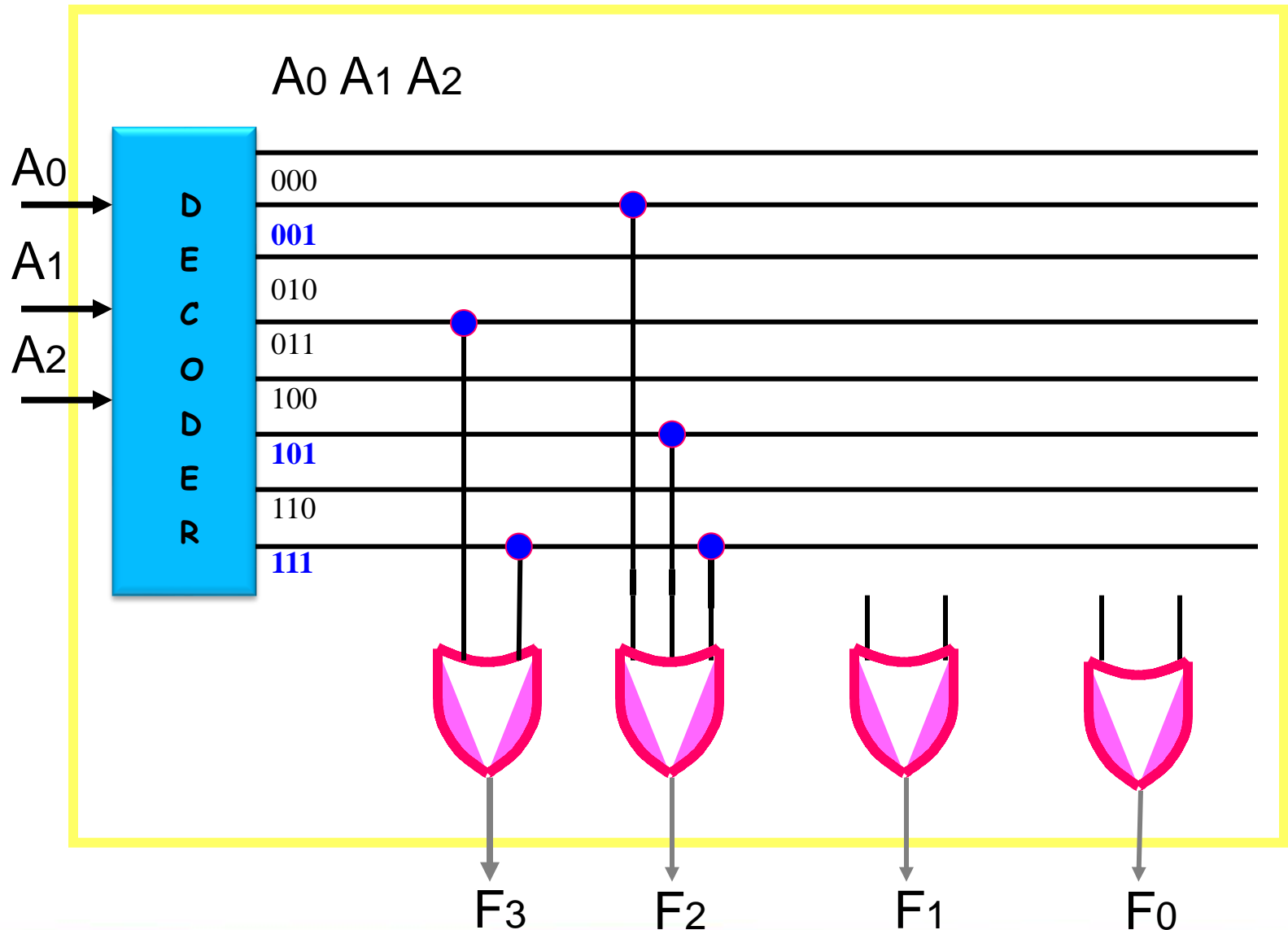
$$F_0 = \bar{A}_0 A_1 \bar{A}_2 + \bar{A}_0 A_1 A_2$$

- When we design ROM's, DO NOT use K-Maps or logic theorems to simplify the above equations
- The above equations are not further simplified.

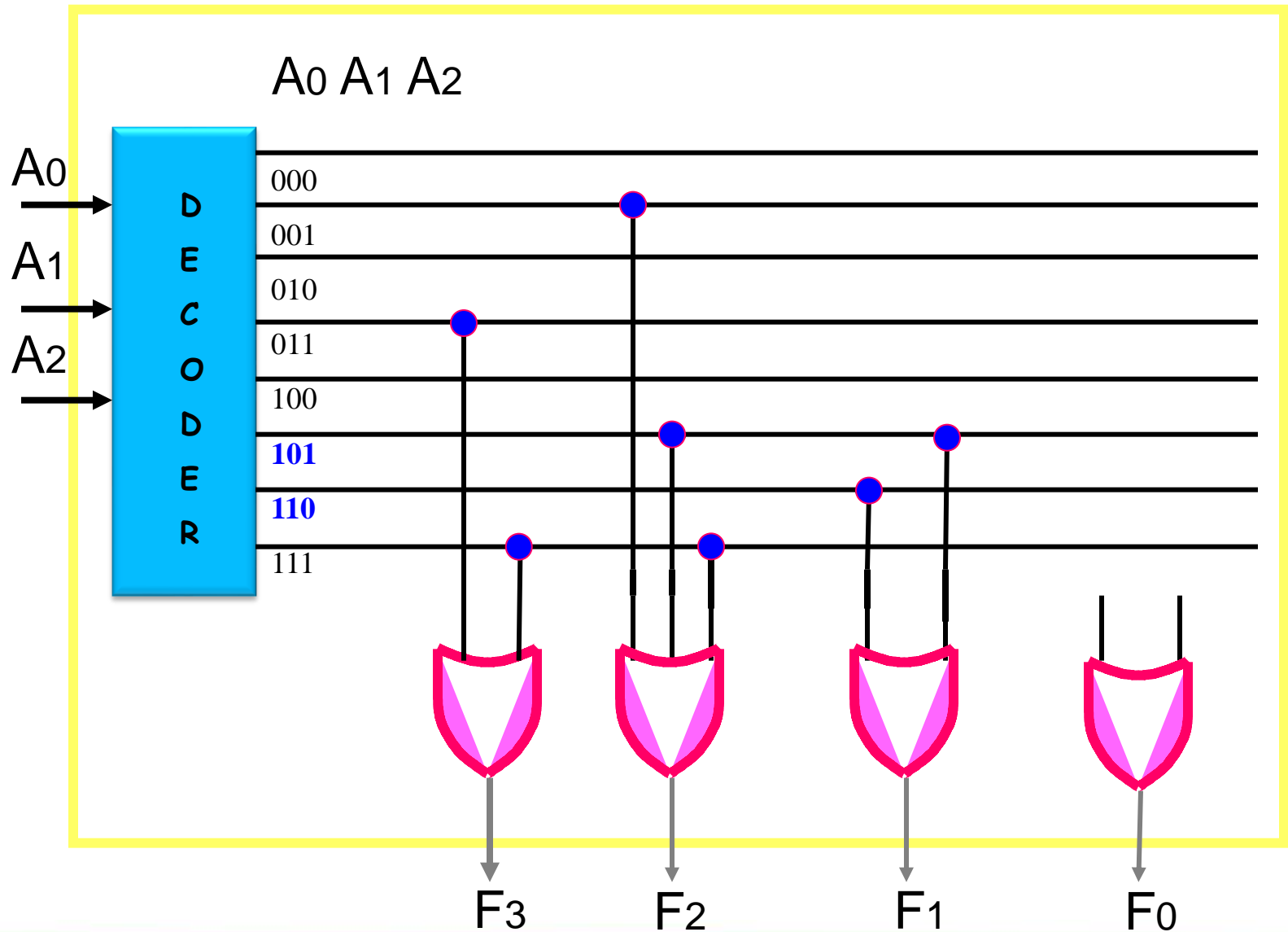
$$F_3 = \bar{A}_0 A_1 A_2 + A_0 A_1 A_2$$



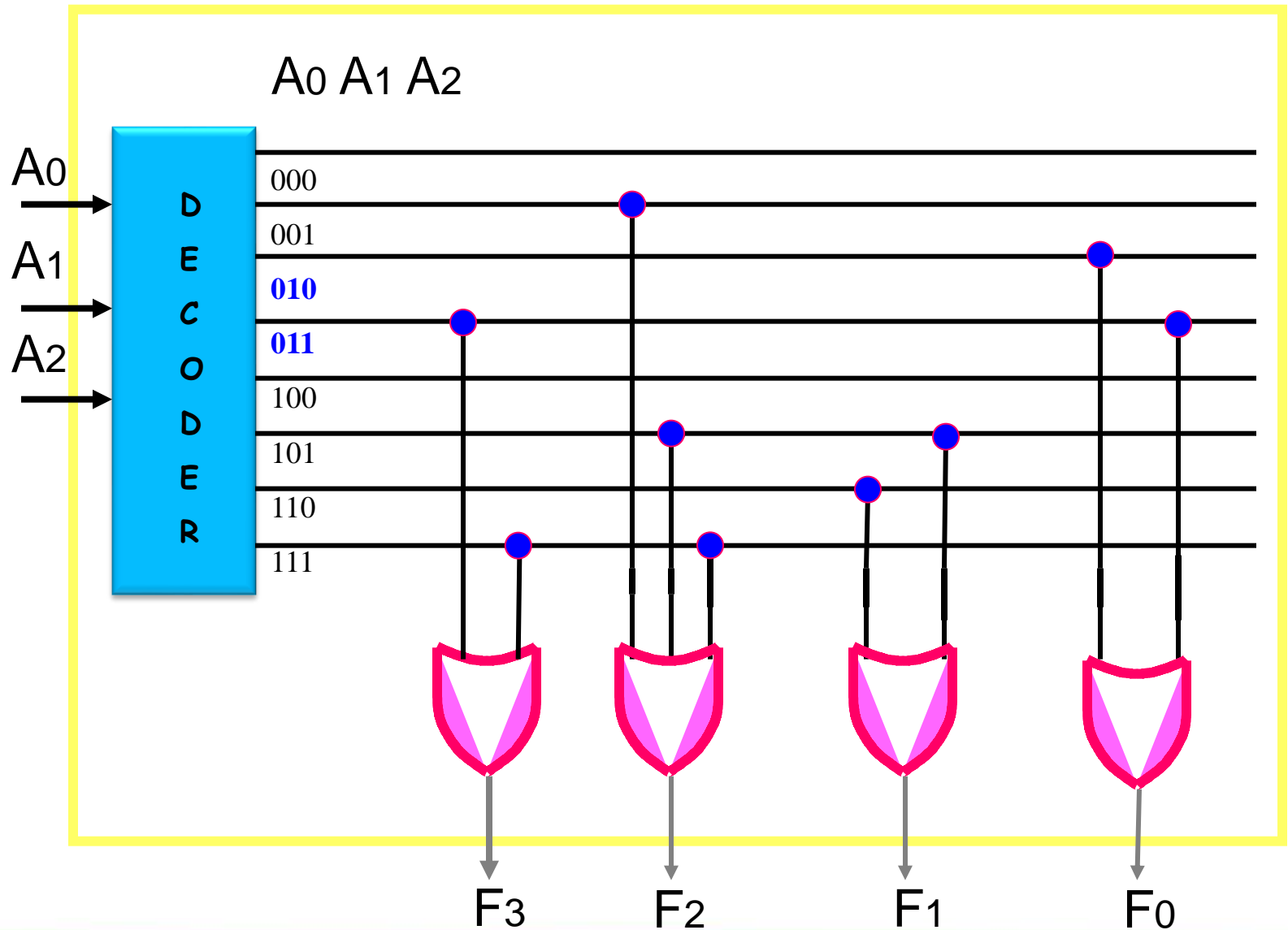
$$F_2 = \overline{A_0} \overline{A_1} A_2 + A_0 \overline{A_1} A_2 + A_0 A_1 A_2$$



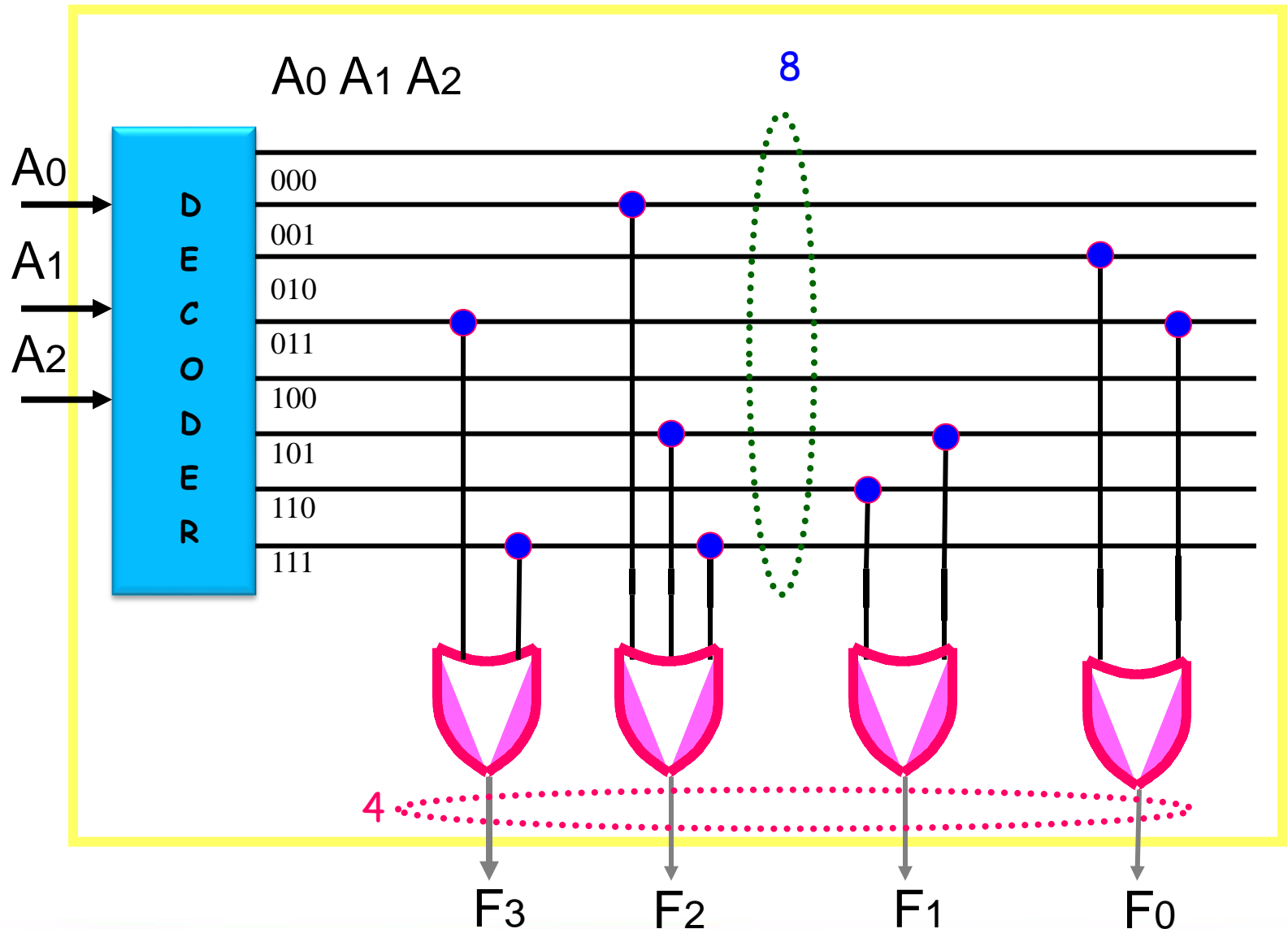
$$F_1 = A_0 A_1 \bar{A}_2 + A_0 \bar{A}_1 A_2$$



$$F_0 = \bar{A}_0 A_1 \bar{A}_2 + \bar{A}_0 A_1 A_2$$

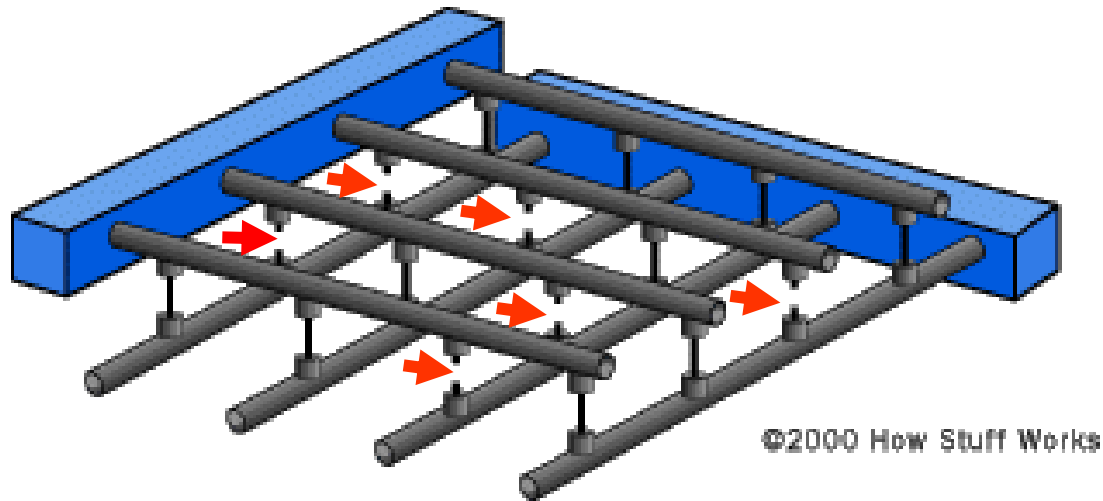


Programmed ROM ... (32 bits)



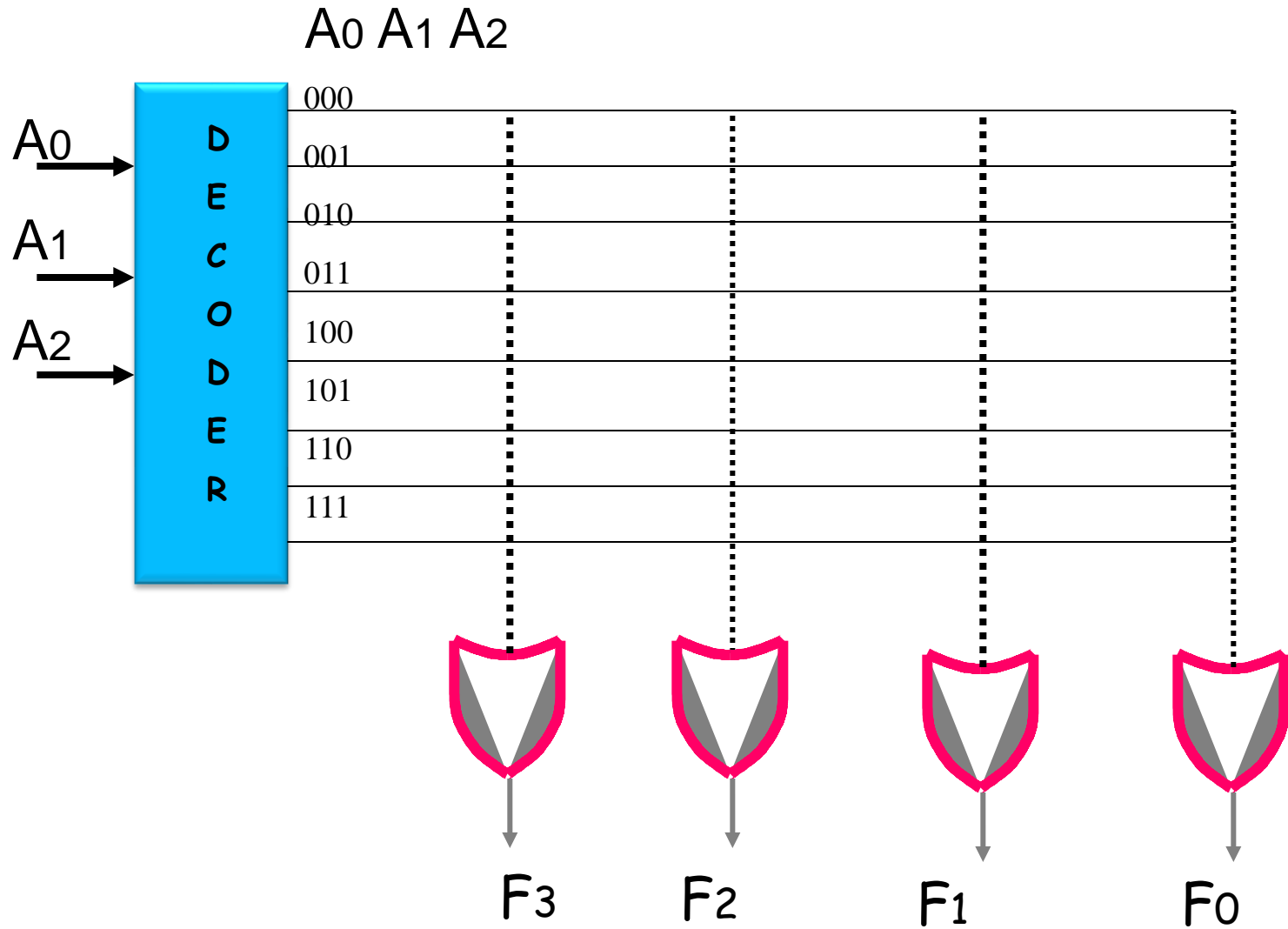
$$4 \times 8 = 32$$

Programmable Read Only Memory (PROM)... Burning

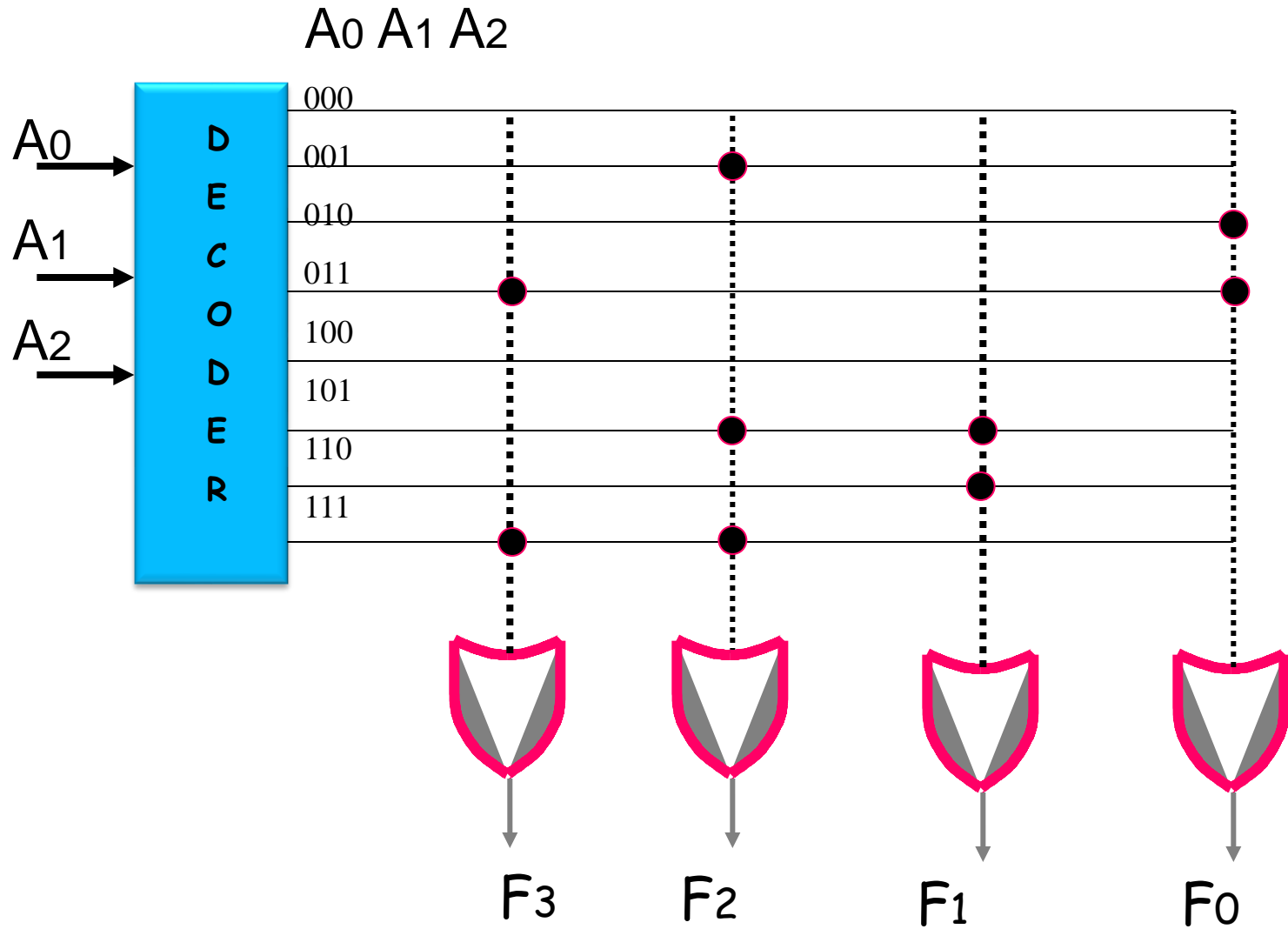


PROM

A simplified way

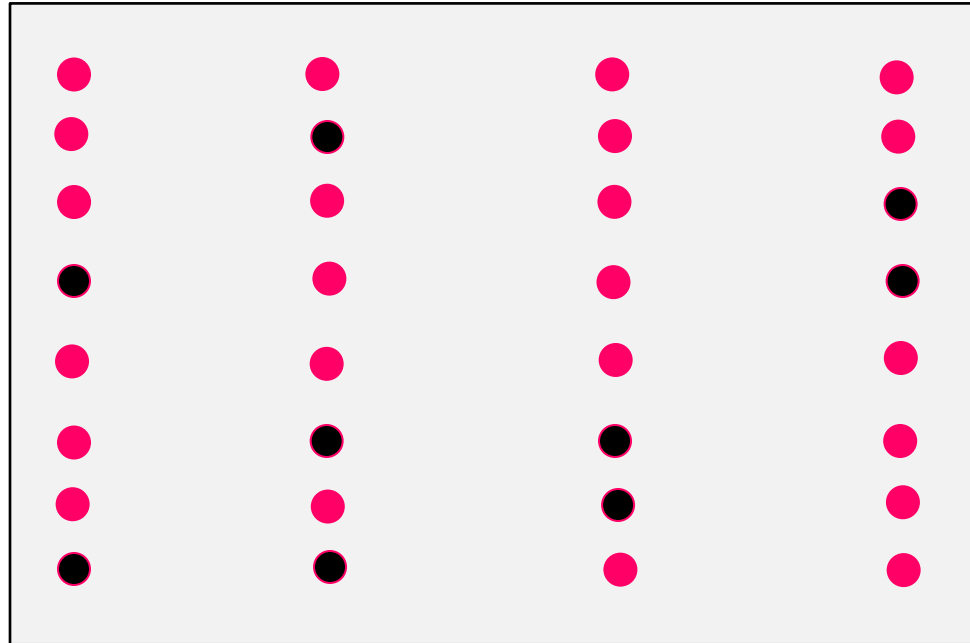


A simplified way



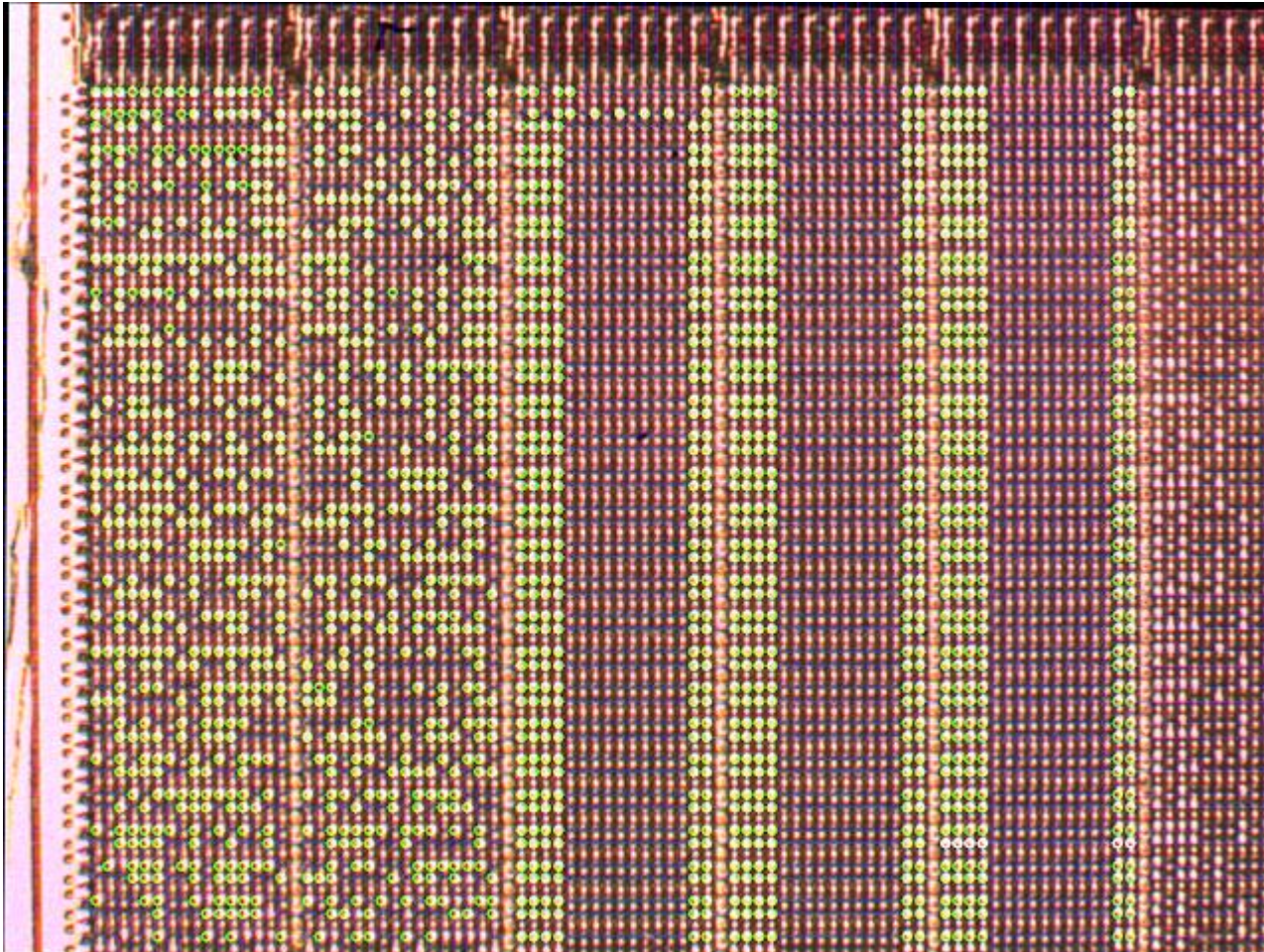
$$8^*4 = 32\text{-bits (32-dots total)}$$

8



4

Masked ROMs - Atmel MARC4

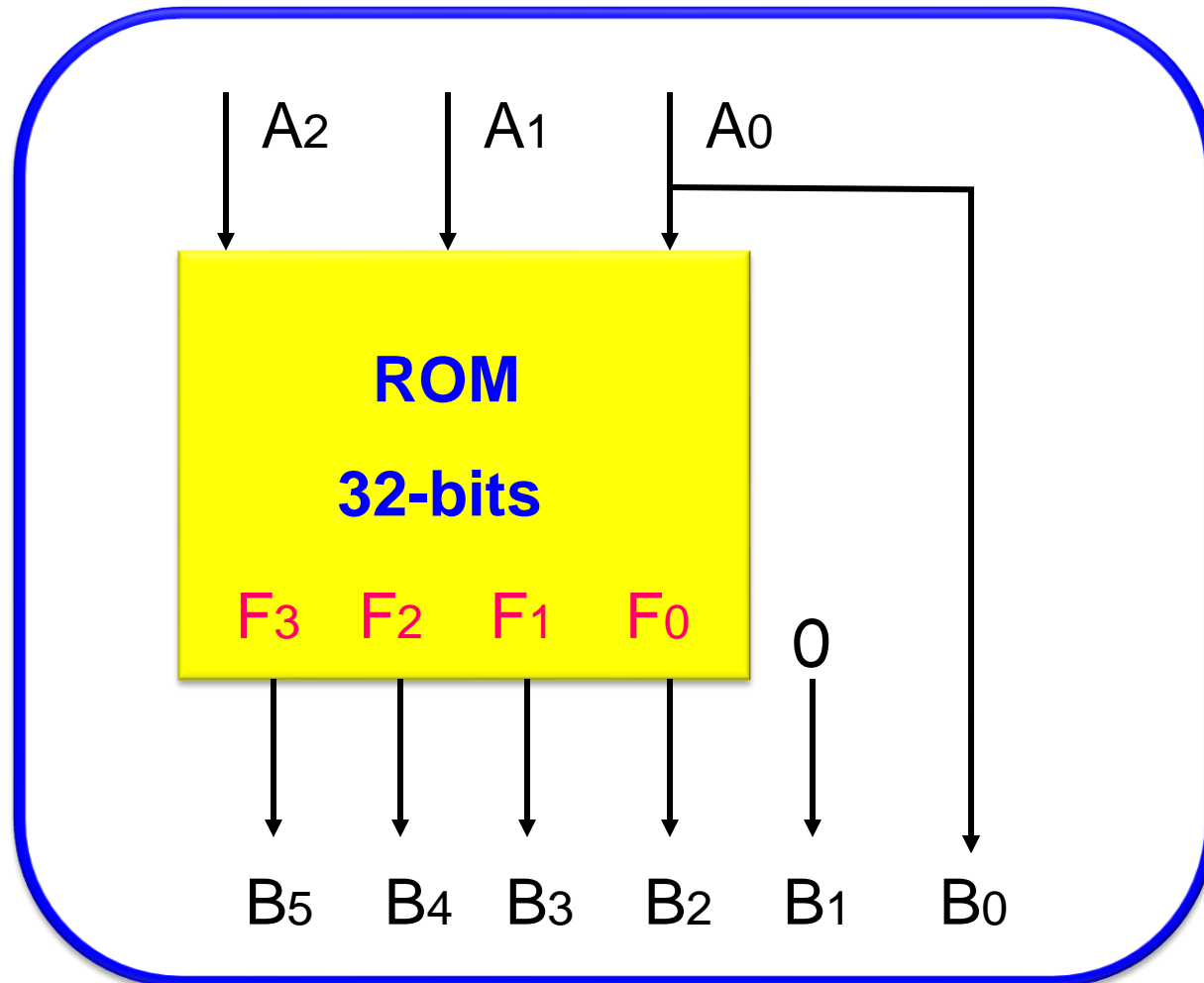


Masked ROMs - Atmel MARC4

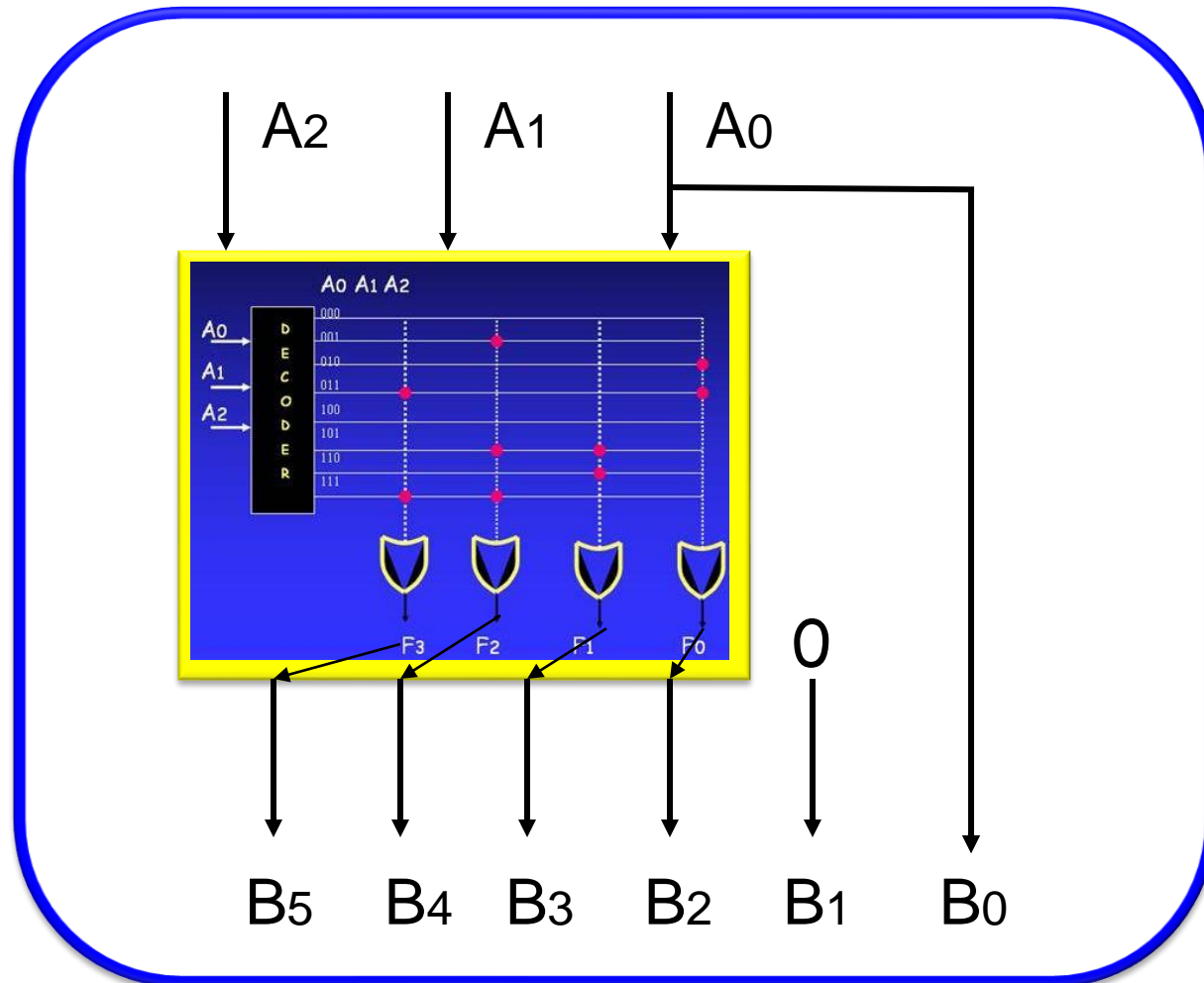


Final ROM

B0 = A0 and **B1 = 0**



Final ROM with Decoder and Gates



ARM (ROM)



ROM (Read Only Memory) is also used in permanent storage of data within an **SoC** (System on Chip) or other integrated circuit applications.