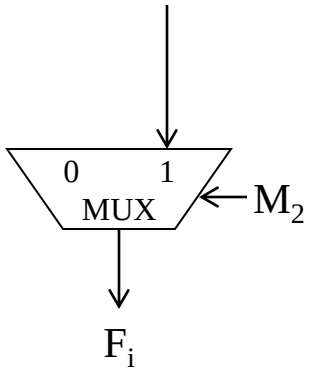
- Design a 4-bit ALU that implements the following set of operations with only the following components (assume 2's complement number representation, no need to implement overflow circuit)
  - 1-bit Full-Adders (FA)
  - 2-input AND/OR/XOR gates
  - Inverters
  - 2:1 MUX

M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B

M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B

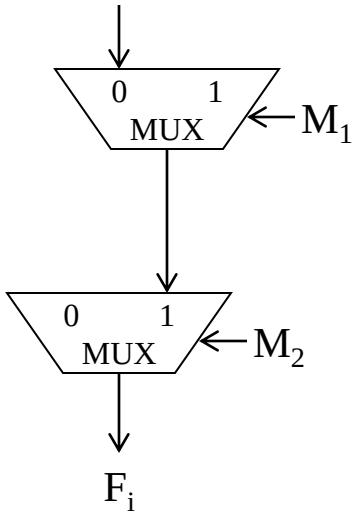
M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B



M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B

A * 2 = left-shift
e.g. 3 * 2 = “011” * 2 = “110” = 6

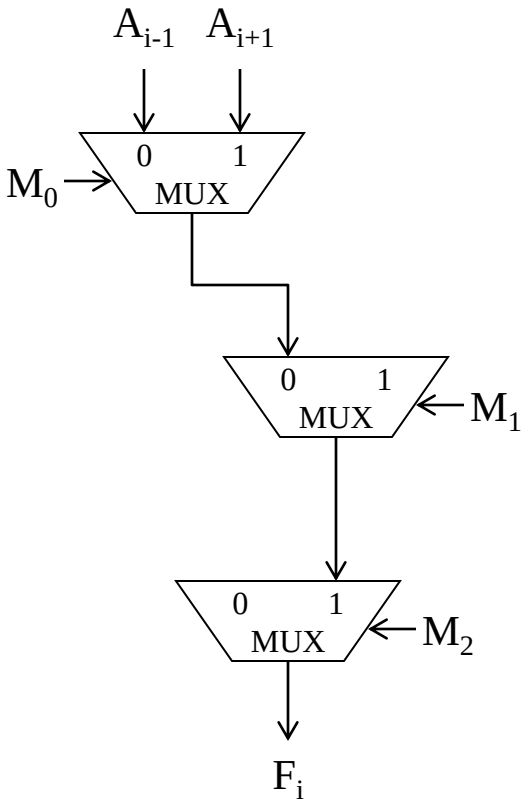
A / 2 = right-shift
e.g. 3 / 2 = “011” / 2 = “001” = 1



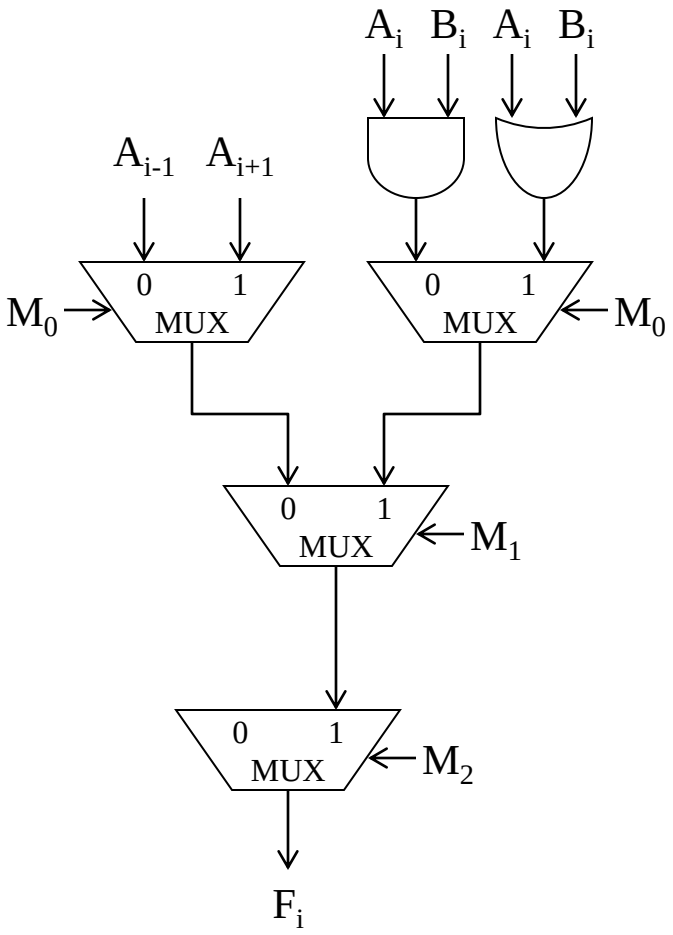
M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B

A * 2 = left-shift
e.g. 3 * 2 = “011” * 2 = “110” = 6

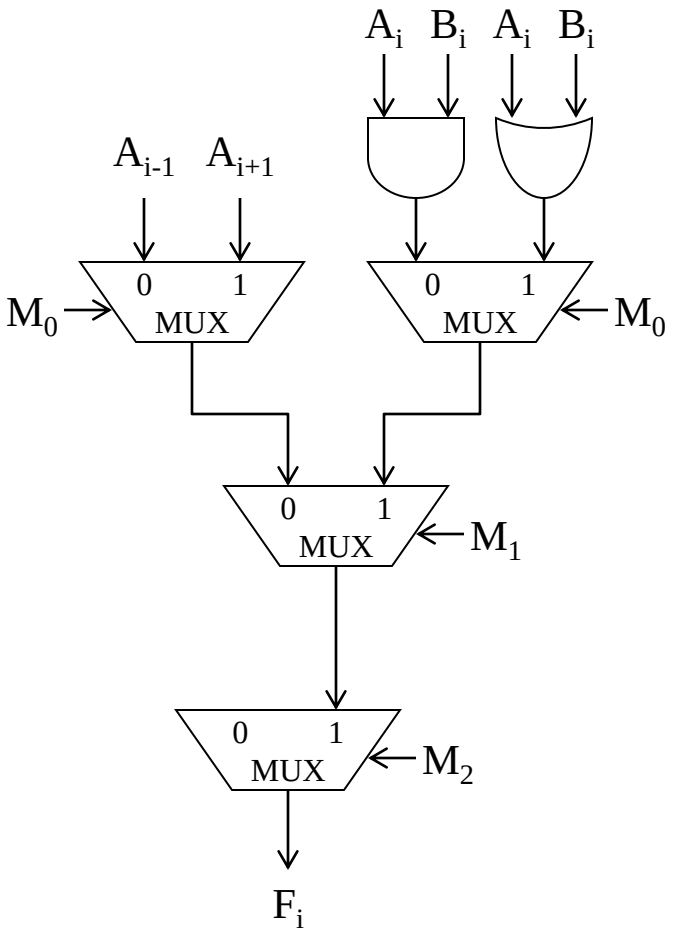
A / 2 = right-shift
e.g. 3 / 2 = “011” / 2 = “001” = 1



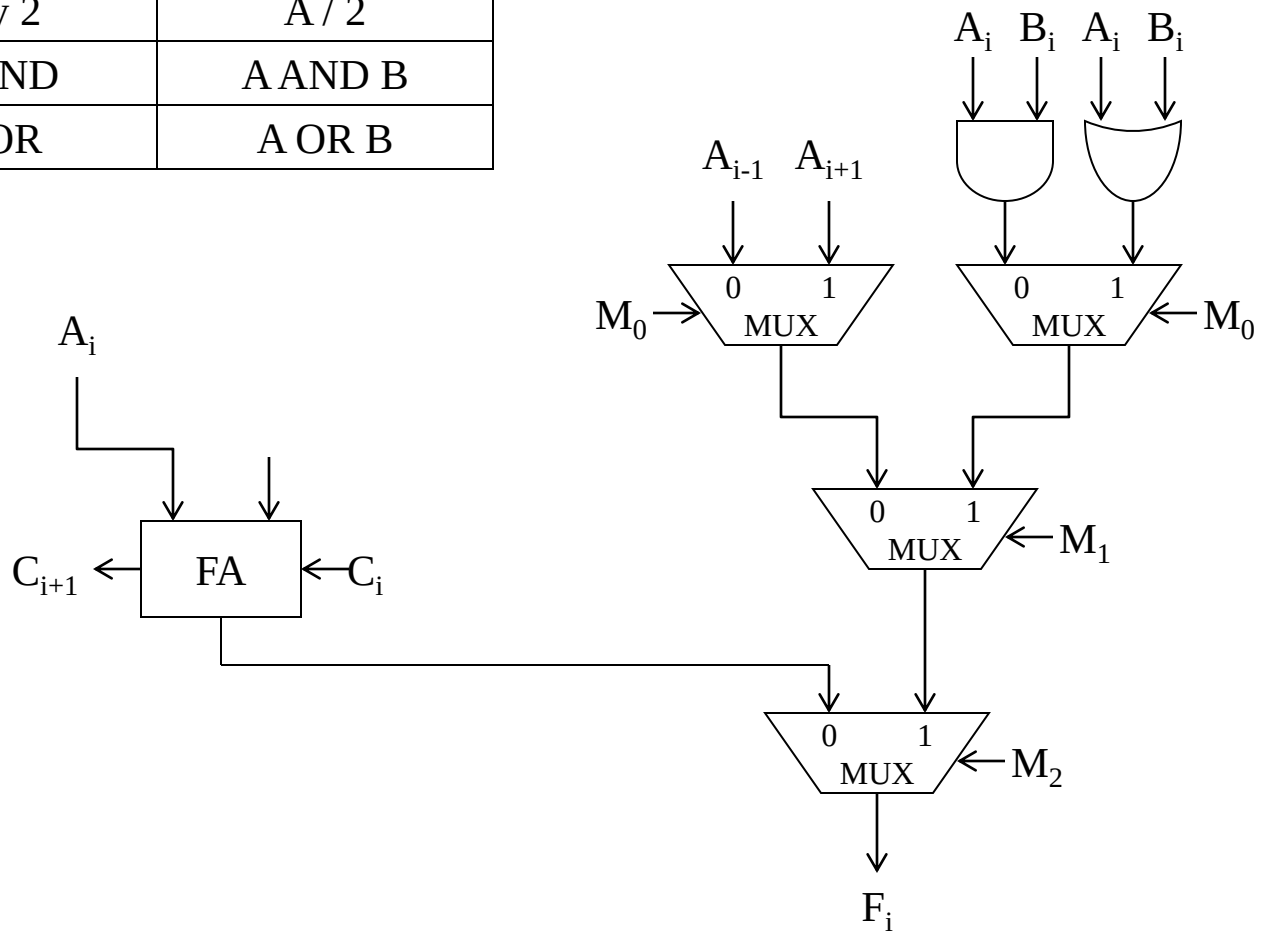
M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B



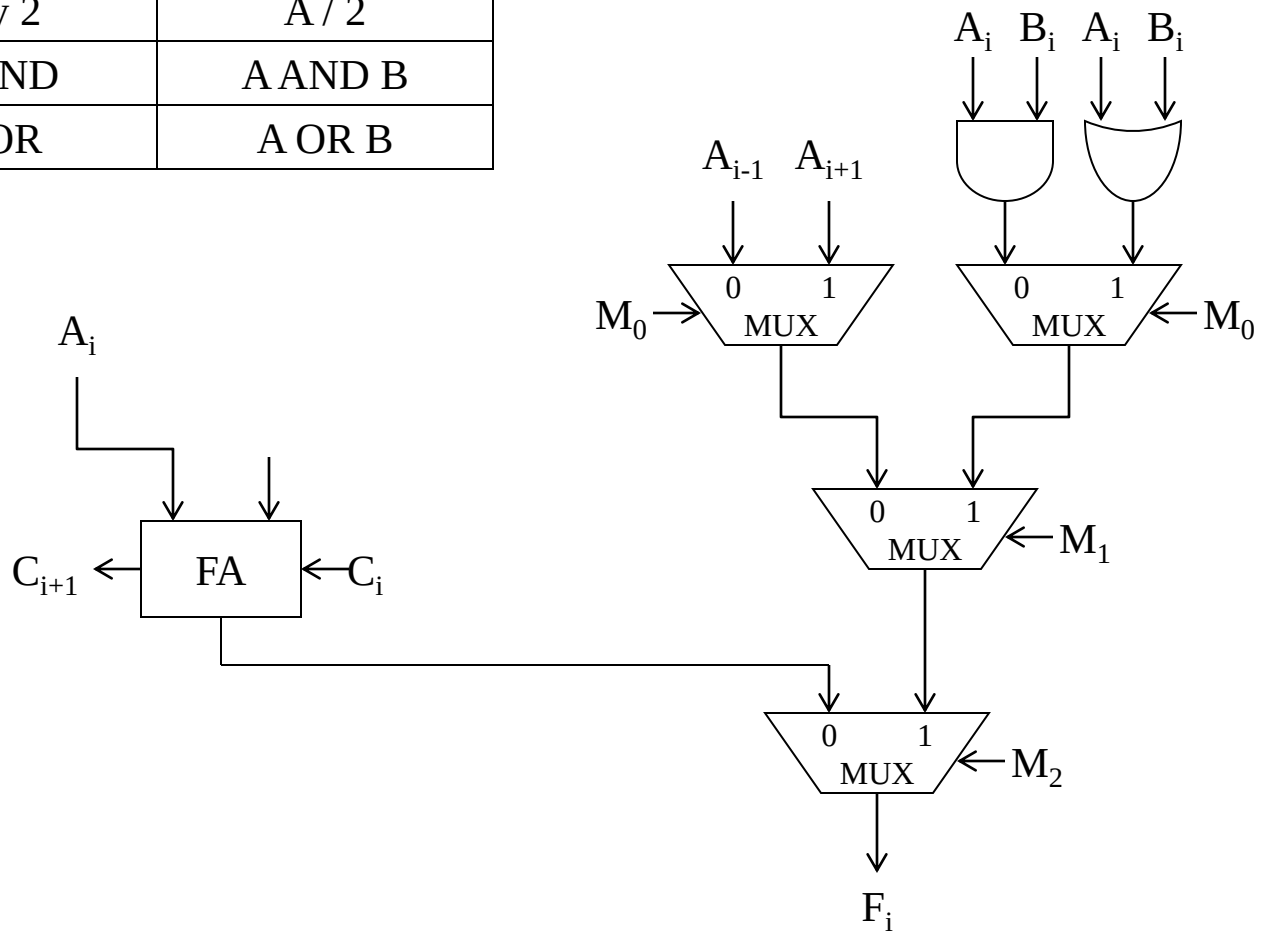
M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B



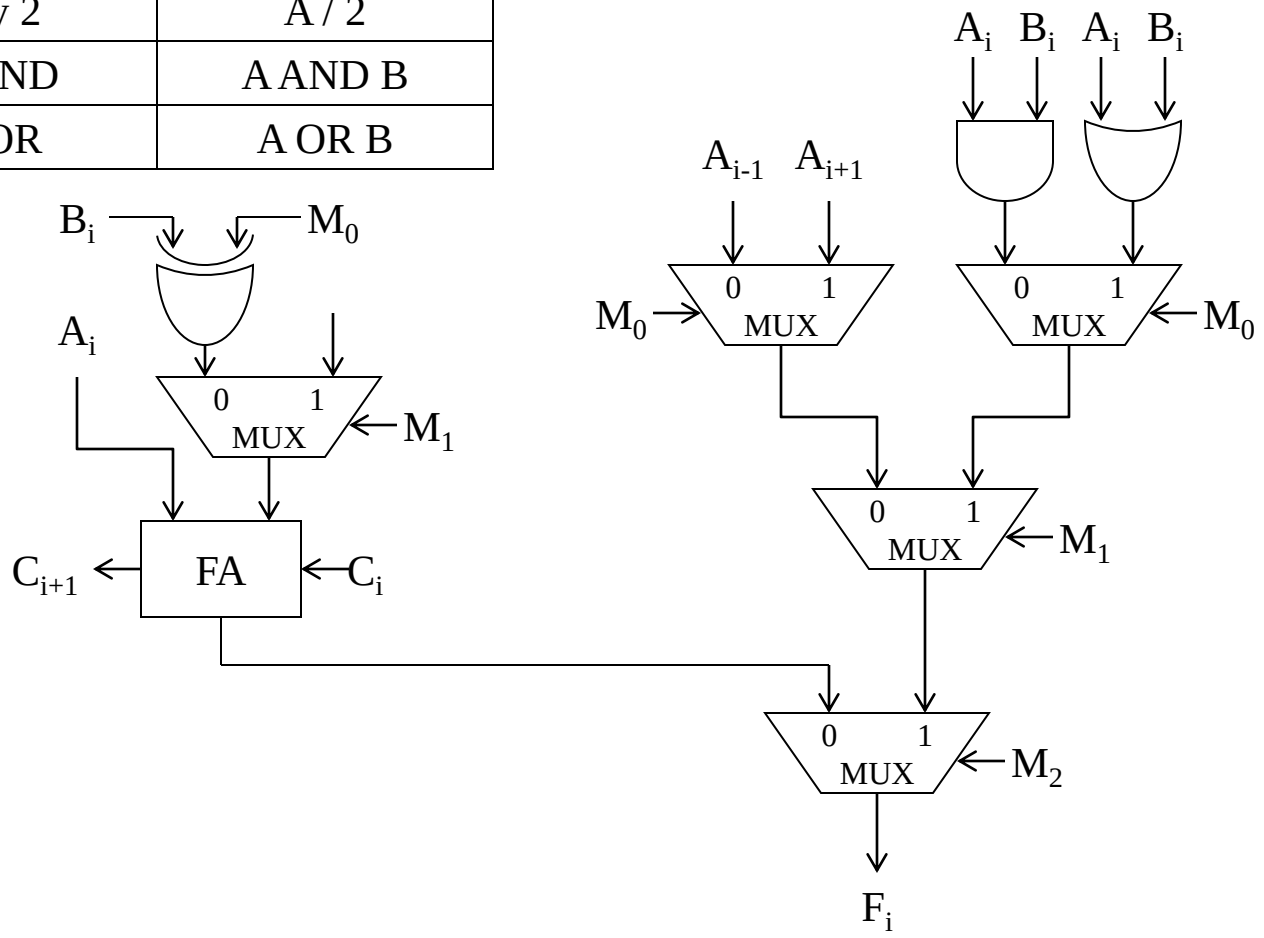
M_2	M_1	M_0	Function Name	F =
0	0	0	Add	A + B
0	0	1	Subtract	A – B
0	1	0	Increment	A + 1
0	1	1	Decrement	A – 1
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B



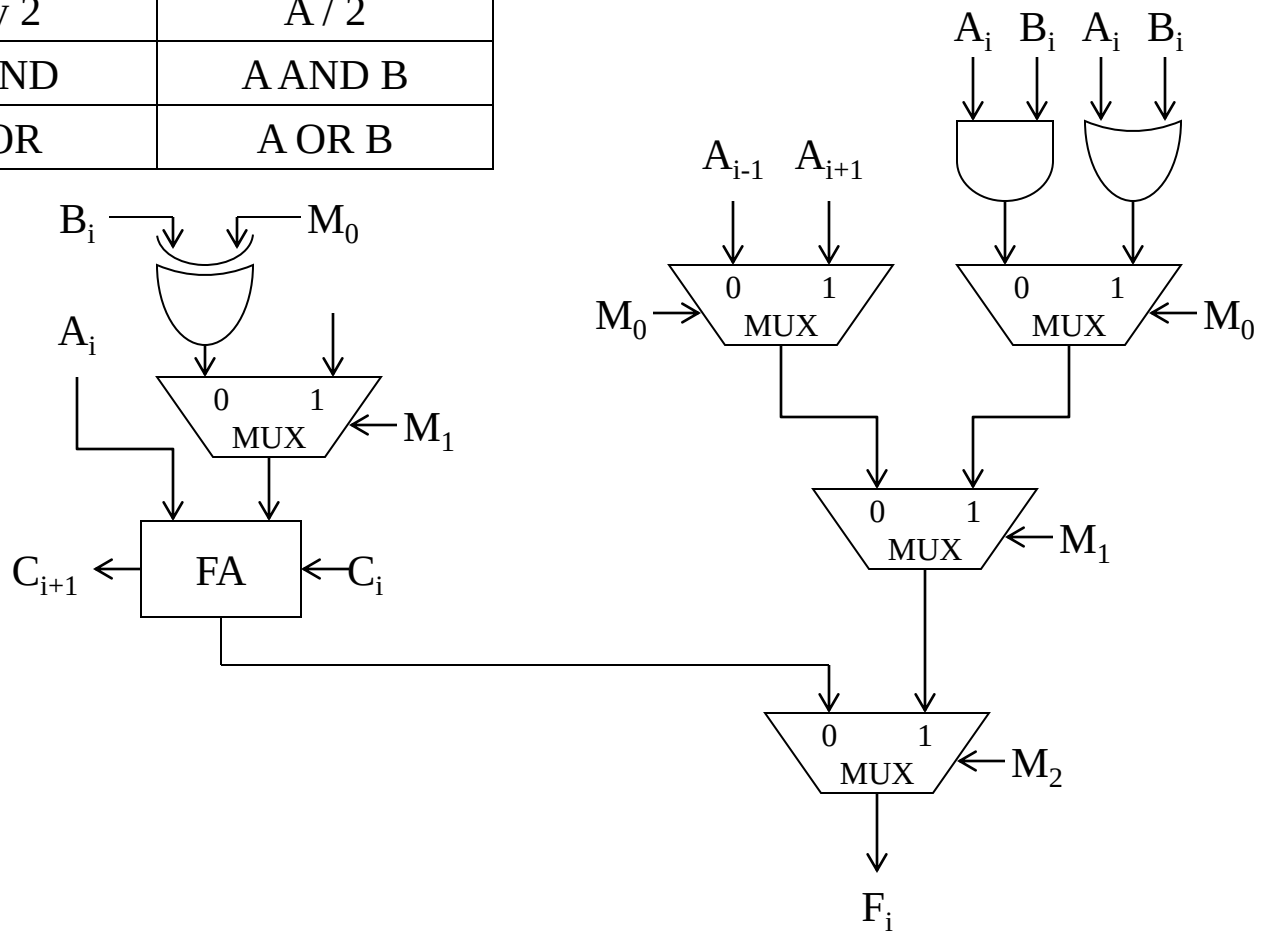
M_2	M_1	M_0	Function Name	F =		C_0
0	0	0	Add	A + B	B_i	0
0	0	1	Subtract	A – B	NOT(B_i)	1
0	1	0	Increment	A + 1	"0"	1
0	1	1	Decrement	A – 1	"1"	0
1	0	0	Multiply by 2	A * 2		
1	0	1	Divide by 2	A / 2		
1	1	0	Bitwise-AND	A AND B		
1	1	1	Bitwise-OR	A OR B		



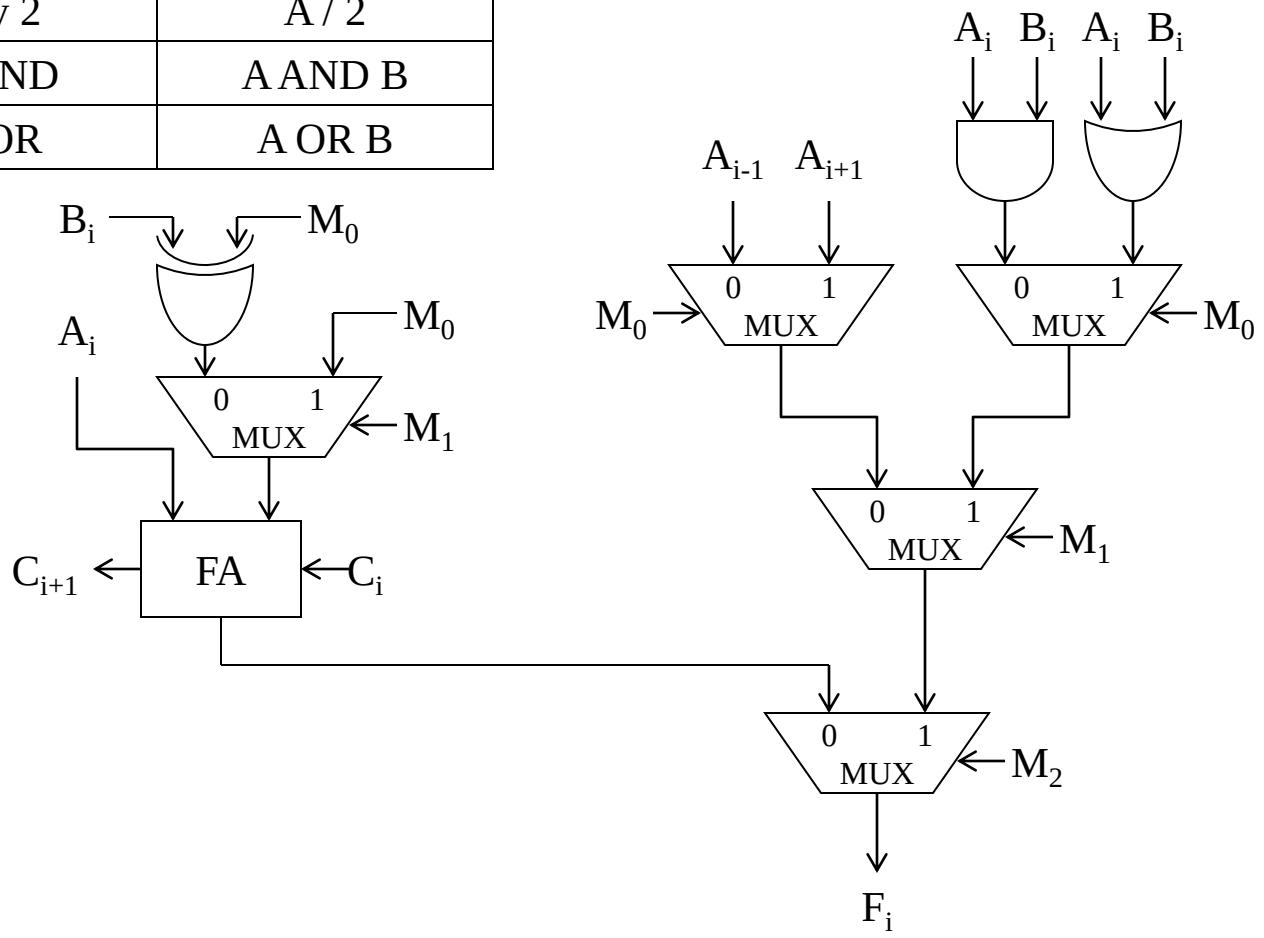
M_2	M_1	M_0	Function Name	F =		C_0
0	0	0	Add	A + B	B_i	0
0	0	1	Subtract	A – B	NOT(B_i)	1
0	1	0	Increment	A + 1	"0"	1
0	1	1	Decrement	A – 1	"1"	0
1	0	0	Multiply by 2	A * 2		
1	0	1	Divide by 2	A / 2		
1	1	0	Bitwise-AND	A AND B		
1	1	1	Bitwise-OR	A OR B		



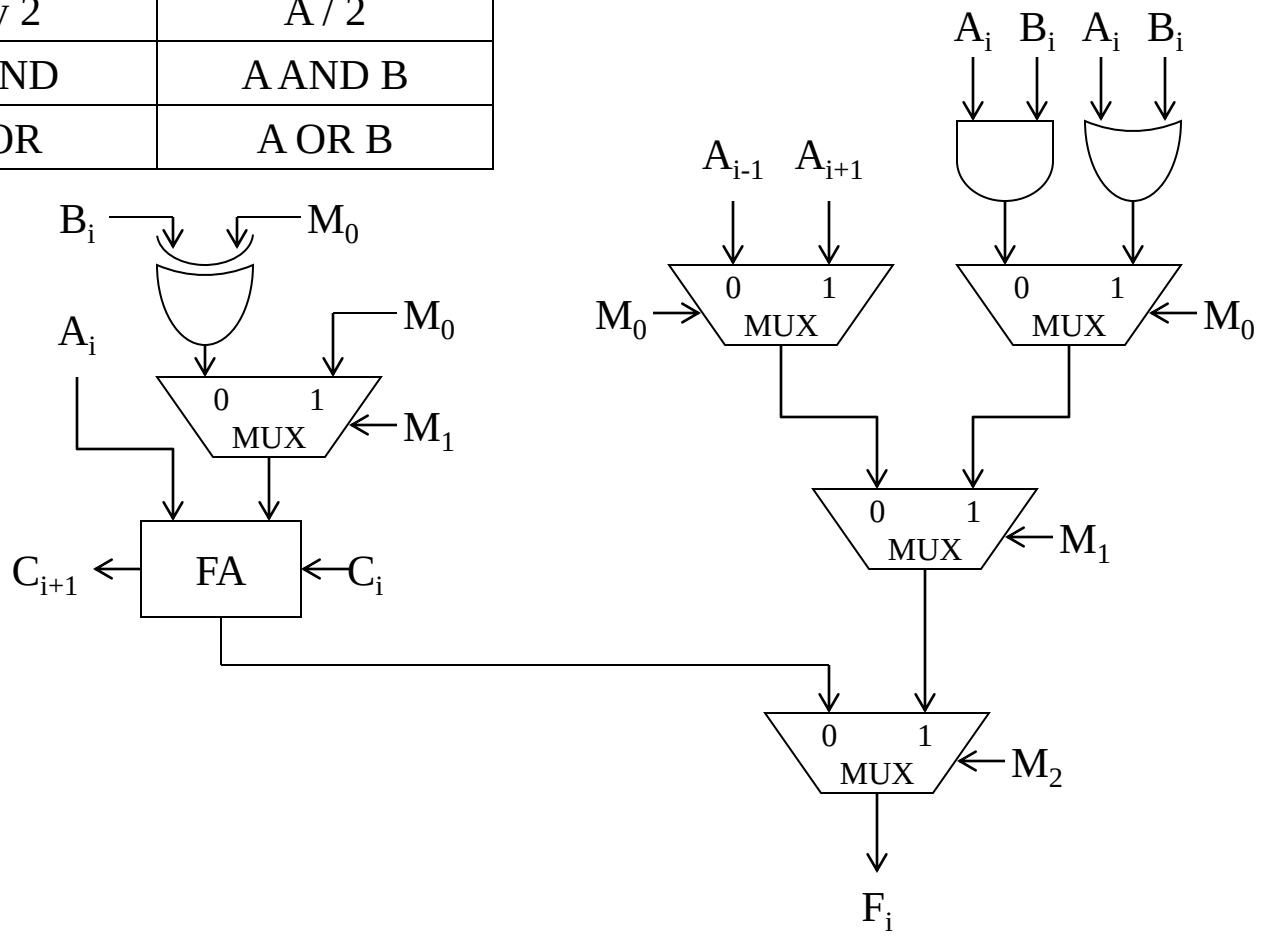
M_2	M_1	M_0	Function Name	F =		C_0
0	0	0	Add	A + B	B_i	0
0	0	1	Subtract	A – B	NOT(B_i)	1
0	1	0	Increment	A + 1	"0"	1
0	1	1	Decrement	A – 1	"1"	0
1	0	0	Multiply by 2	A * 2		
1	0	1	Divide by 2	A / 2		
1	1	0	Bitwise-AND	A AND B		
1	1	1	Bitwise-OR	A OR B		



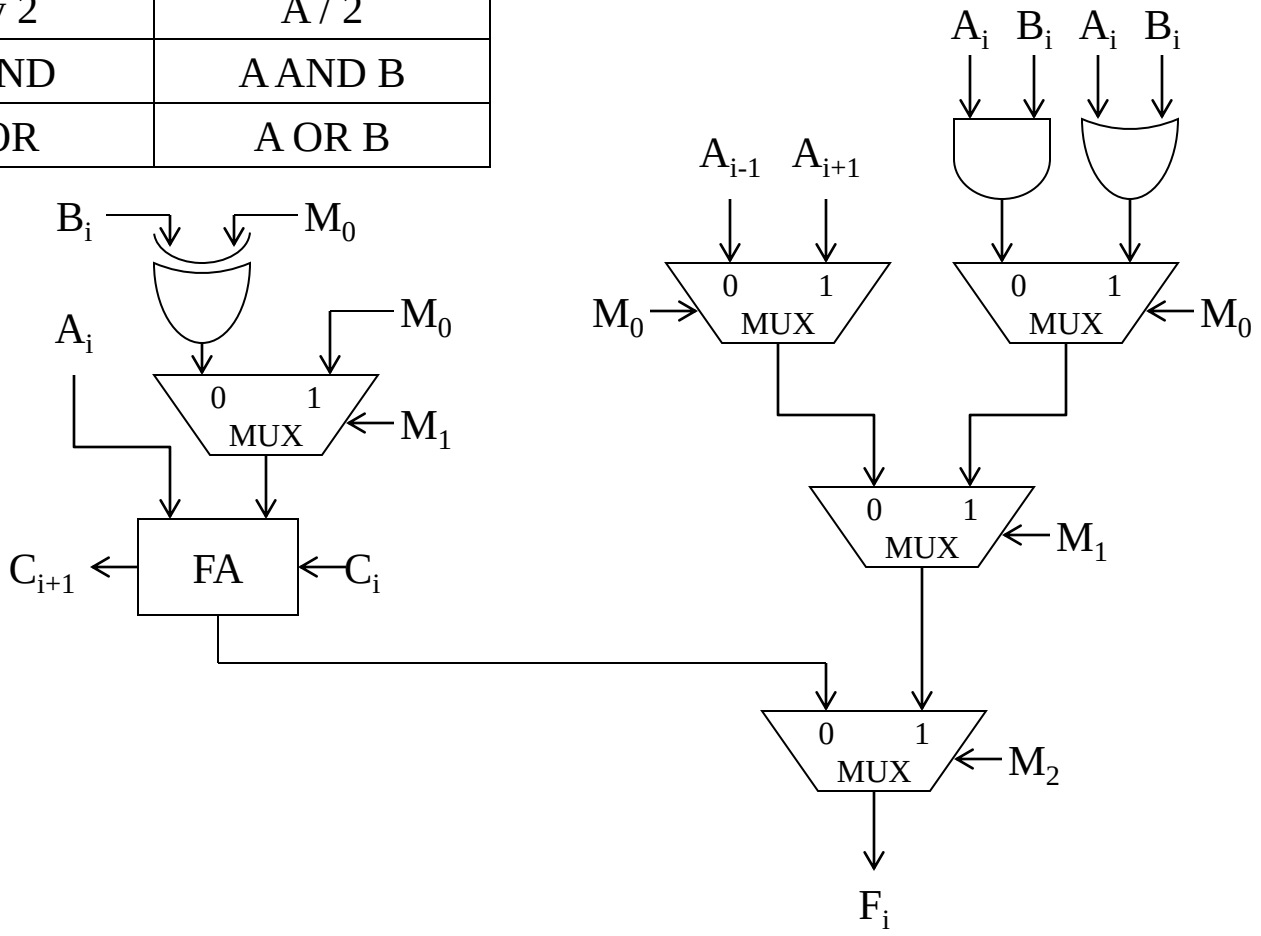
M_2	M_1	M_0	Function Name	F =		C_0
0	0	0	Add	A + B	B_i	0
0	0	1	Subtract	A – B	NOT(B_i)	1
0	1	0	Increment	A + 1	"0"	1
0	1	1	Decrement	A – 1	"1"	0
1	0	0	Multiply by 2	A * 2		
1	0	1	Divide by 2	A / 2		
1	1	0	Bitwise-AND	A AND B		
1	1	1	Bitwise-OR	A OR B		



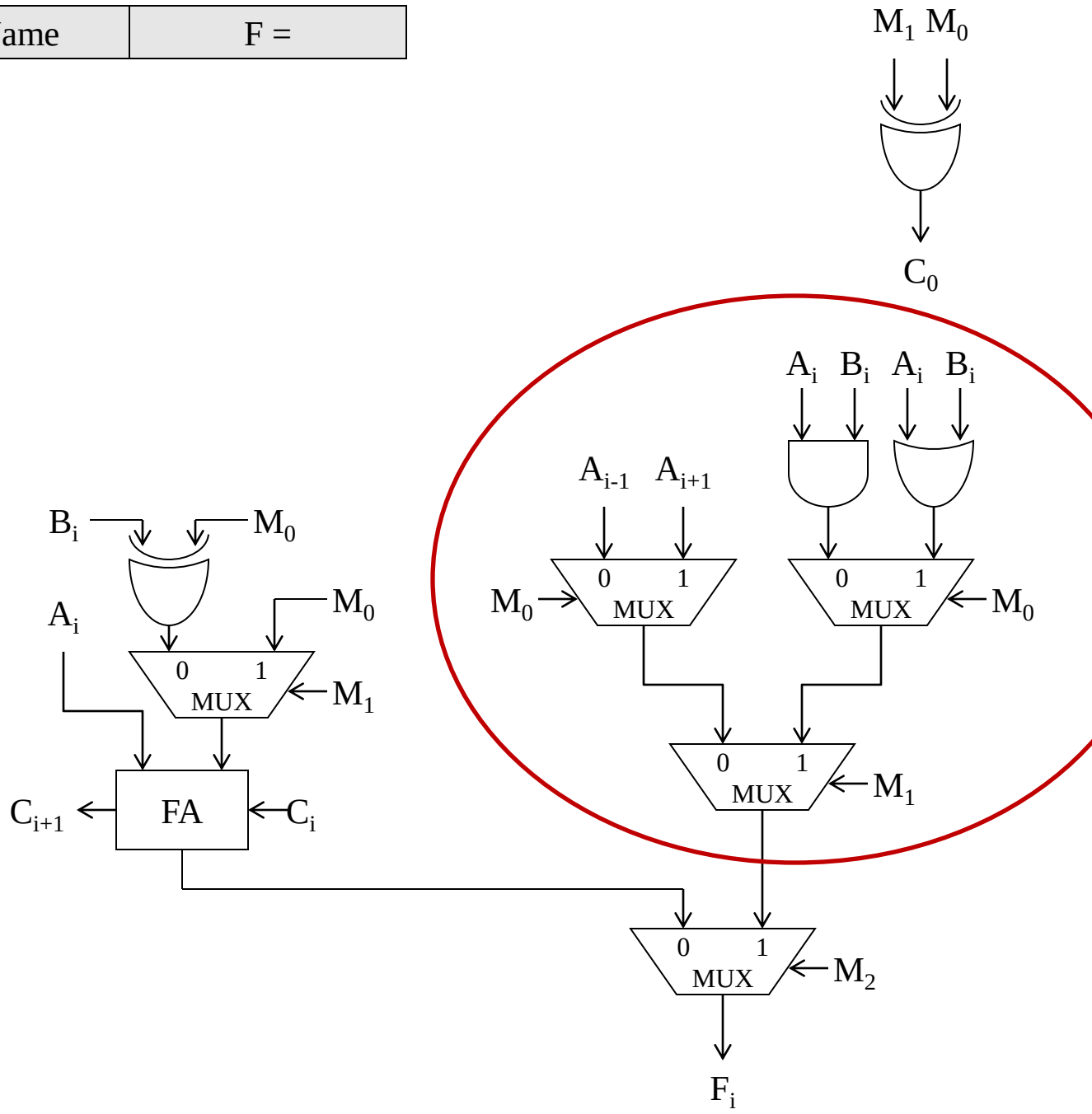
M_2	M_1	M_0	Function Name	F =		C_0
0	0	0	Add	A + B	B_i	0
0	0	1	Subtract	A – B	NOT(B_i)	1
0	1	0	Increment	A + 1	"0"	1
0	1	1	Decrement	A – 1	"1"	0
1	0	0	Multiply by 2	A * 2		
1	0	1	Divide by 2	A / 2		
1	1	0	Bitwise-AND	A AND B		
1	1	1	Bitwise-OR	A OR B		



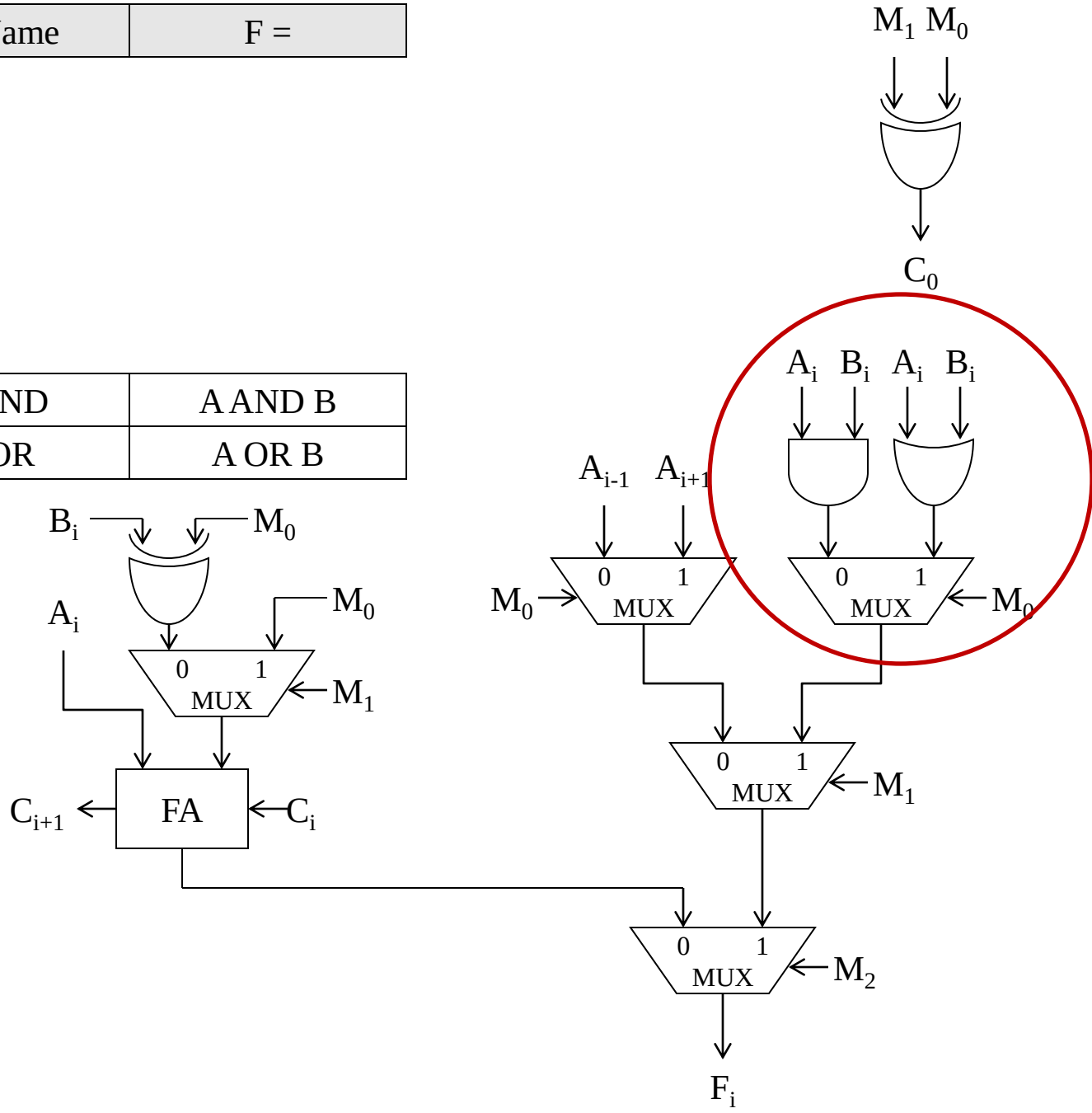
M_2	M_1	M_0	Function Name	F =		C_0
0	0	0	Add	A + B	B_i	0
0	0	1	Subtract	A – B	NOT(B_i)	1
0	1	0	Increment	A + 1	"0"	1
0	1	1	Decrement	A – 1	"1"	0
1	0	0	Multiply by 2	A * 2		
1	0	1	Divide by 2	A / 2		
1	1	0	Bitwise-AND	A AND B		
1	1	1	Bitwise-OR	A OR B		



M_2	M_1	M_0	Function Name	F =
0				
0				
0				
0				
1				
1				
1				
1				



M_2	M_1	M_0	Function Name	F =
0				
0				
0				
0				
1				
1				
1	1	0	Bitwise-AND	A AND B
1	1	1	Bitwise-OR	A OR B



M_2	M_1	M_0	Function Name	F =
0				
0				
0				
0				
1	0	0	Multiply by 2	A * 2
1	0	1	Divide by 2	A / 2
1				
1