P3.1 VSμP – ALU Design

Lluís Terés

Instituto de Microelectrónica de Barcelona, IMB-CNM (CSIC)

Universitat Autònoma de Barcelona (UAB)

CONTENTS

P3.1

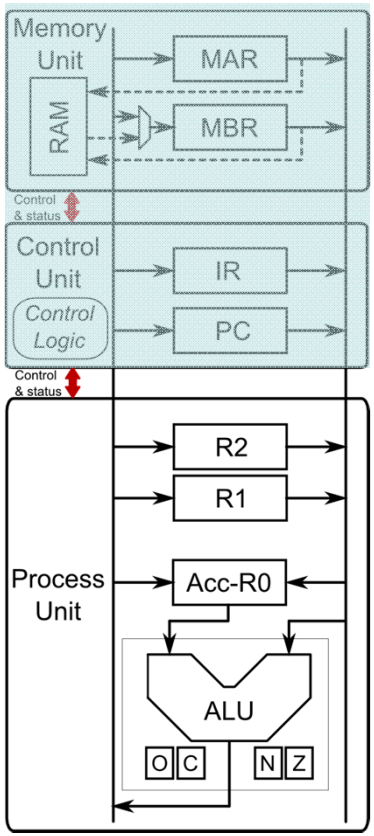
UAB Universitat Autònoma de Barcelona

ALU Capabilities and General Structure

P3.1

UAB Universitat Autònoma de Barcelona

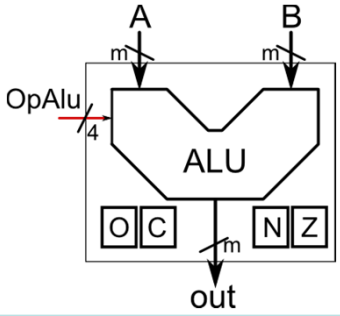
VSµP ALU Design: Processing unit & updated ALU



Registers

R2, R1: General purpose registers.
Acc = R0: Special register useful as general purpose register but strongly related with ALU as it provides one of the operands.

ALU



ALU: performs a lot of arithmetic, logical and relational operations using the two input operands, A & B, and leaving the result on the bus "Out" which is connected to Bus-A ready to be stored in any internal register. Also the ALU flags (O,C,N,Z) are being updated by each operation.

O,C,N,Z flags: "Overflow", "Carry", "Negative" and "Zero" flags from ALU:

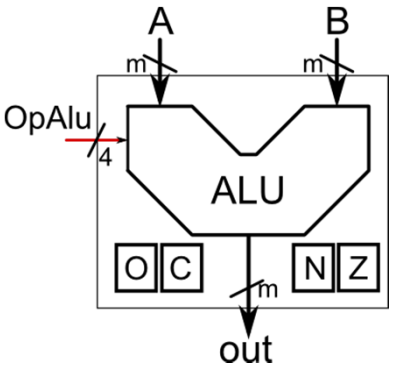
- Overflow: the result requires more bits than the representation capability of "Out" bus
- Carry: once operation done there is a carry bit after the most significant bit of "Out"
- Negative: the result in "Out" is a negative number
- Zero: the result in "Out" is "0…0"

ALU Capabilities and General Structure

P3.1

UAB Universitat Autònoma de Barcelona

VSμP : ALU Operations and Flags



Out <= A OpAlu B;
O <= '1' when "Overflow"
C <= "1" when "Carry=1";
N <= "1" when "Out negative";
Z <= "1" when "Out=0…0";

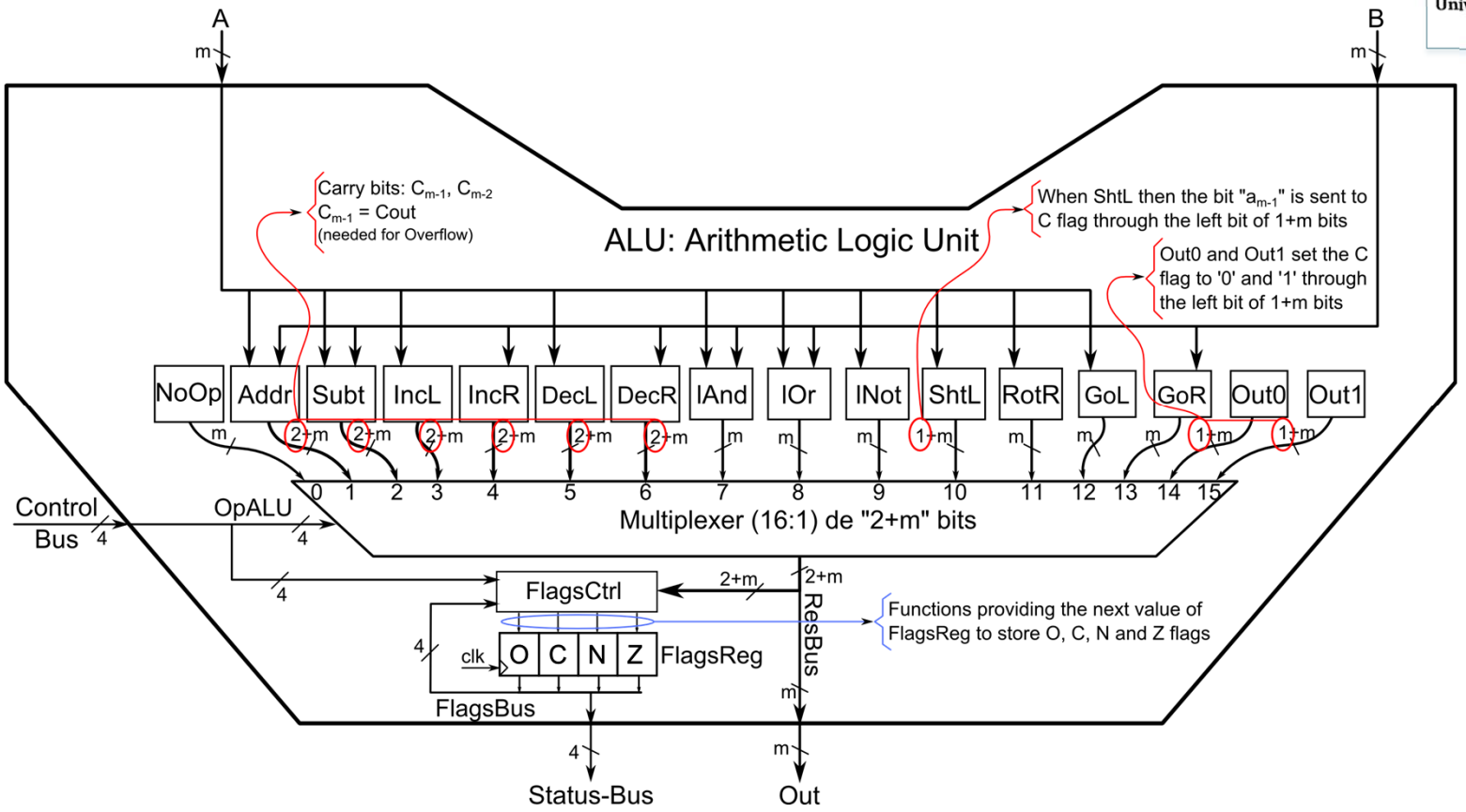
#	OpALU	Mnemonic	Behaviour	Flags	Comments	
1	0000	NoOp	No oper.: OUT<="0…0"	Keep	Any operation is done	
2	0001	Addr	OUT <= A + B	O, C, N, Z	Addition	Arithmetic
3	0010	Subt	OUT <= A - B	O, C, N, Z	Subtraction	Arithmetic
4	0011	IncL	OUT <= A + '1'	O, C, N, Z	Increment left operand A	Arithmetic
5	0100	IncR	OUT <= B + '1'	O, C, N, Z	Increment right operand B	Arithmetic
6	0101	DecL	OUT <= A – '1'	O, C, N, Z	Decrement left operand A	Arithmetic
7	0110	DecR	OUT <= B – '1'	O, C, N, Z	Decrement right operand B	Arithmetic
8	0111	lAnd	OUT <= A and B	N, Z	Logical "and" of A & B	Logic
9	1000	lOr	OUT <= A or B	N, Z	Logical "or" of A & B	Logic
10	1001	lNot	OUT <= not A	N, Z	Logical "not" of A	Logic
11	1010	ShtL	OUT <= ShiftLeft A	C,N, Z	Shift A one position to left	Shift
12	1011	RotR	OUT <= RotateRight A	N, Z	Rotate A one position to right	Shift
13	1100	GoL	OUT <= A	Keep	Left operand A to output	Data transfer
14	1101	GoR	OUT <= B	Keep	Right operand B to output	Data transfer
15	1110	Out0	OUT <= "0…0"	N, Z	Put all output bits at '0'	Data transfer
16	1111	Out1	OUT <= "1…1"	C, N, Z	Put all output bits at '1'	Data transfer

VSμP : ALU Components

P3.1

UAB Universitat Autònoma de Barcelona

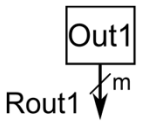
ALU: G[al] Structure and Components



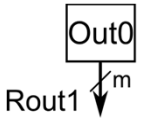
VSμP : ALU Components

UAB Universitat Autònoma de Barcelona

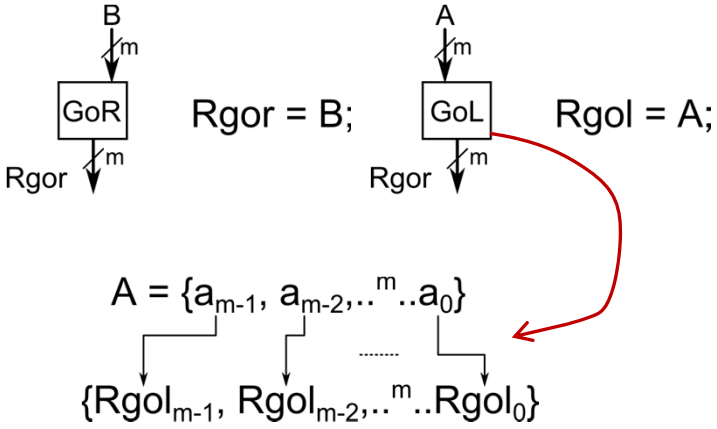
ALU: Non Arithmetic Blocks

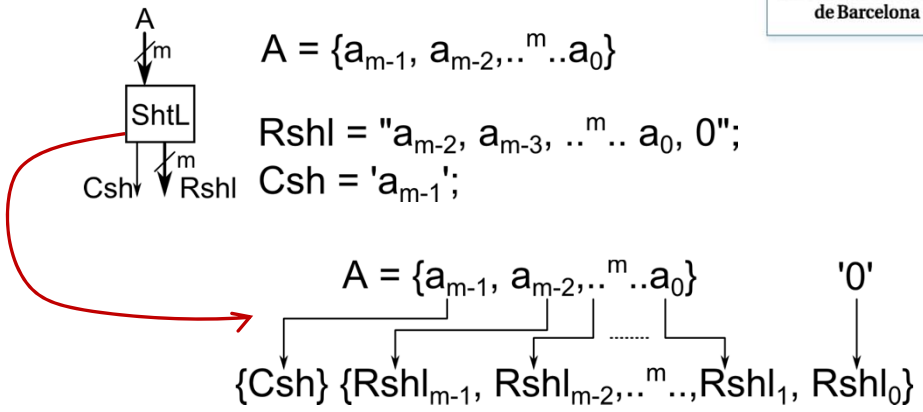


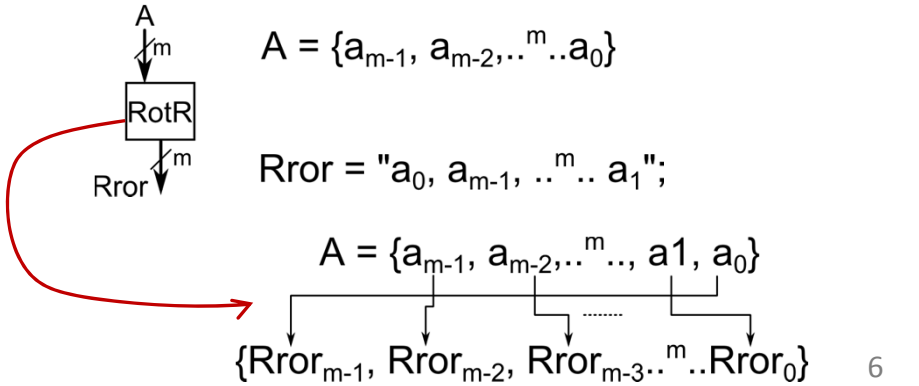
Rout1 = "1..^{m}..1";
Flag C = '1';



Rout0 = "0..^{m}..0";
Flag C = '0';

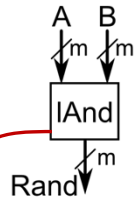






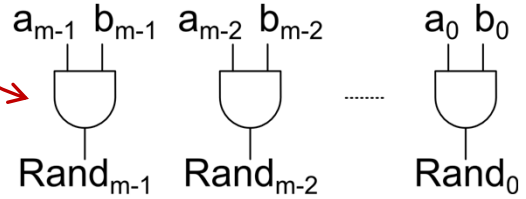
VSμP : ALU Components

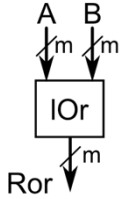
ALU: Logic Operators



$A = \{a_{m-1}, a_{m-2},..^{m}..a_0\}$
$B = \{b_{m-1}, b_{m-2},..^{m}..b_0\}$

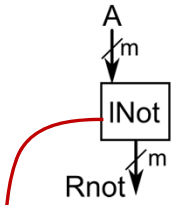
$Rand = "a_{m-1}\cdot b_{m-1},\ a_{m-2}\cdot b_{m-2},\ ..^{m}..\ a_0\cdot b_0";$





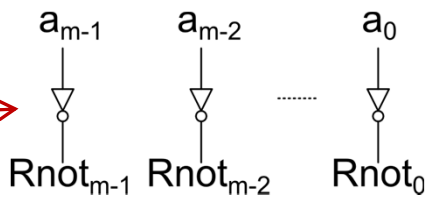
$A = \{a_{m-1}, a_{m-2},..^{m}..a_0\}$
$B = \{b_{m-1}, b_{m-2},..^{m}..b_0\}$

$Ror = "a_{m-1}+b_{m-1},\ a_{m-2}+b_{m-2},\ ..^{m}..\ a_0+b_0";$

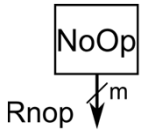


$A = \{a_{m-1}, a_{m-2},..^{m}..a_0\}$

$Rnot = "\overline{a_{m-1}},\ \overline{a_{m-2}},\ ..^{m}..\ \overline{a_0}";$



ALU: No Operation



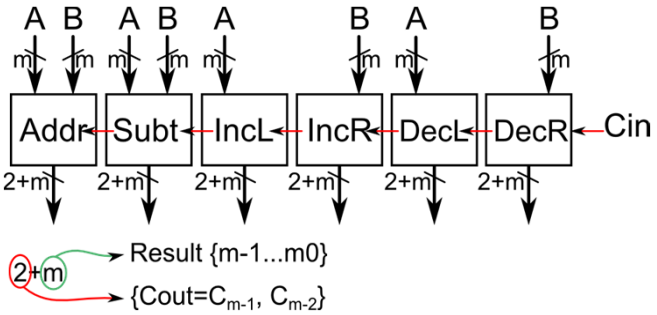
$Rnop = "0..^{m}..0";$

Keep unchanged the ALU flags

VSμP : ALU Components

UAB
Universitat Autònoma de Barcelona

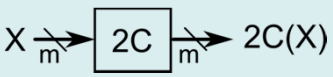
ALU: Arithmetic Operators



- Six different operations using operands of "m" bits
- Each involving one adder or one subtractor
- Outputs of each module:
  - result of "m" bits
  - "1" carry out (Cout=C_{m-1}) bit → 1+m bits
  - "1" carry bit (C_{m-2}) to compute "Overflow = C_{m-1} orex C_{m-2}"
  - Total output bits: 2+m

ALU: Signed numbers

- This CPU will use 2's Complement (2C) representation for signed numbers
  - 2C(+X) = -X; 2C(-X) = +X
  - X - Y = X + 2C(Y); X - 1 = X + 2C(1); Y - 1 = Y + 2C(1)
  - X_{m-1} contains the sign of X (0: positive, 1: negative)
- ***All the ALU arith operations could be done by just "adding" the right 2C operands***
- 1's Complement (1C) is the complementary value bit by bit.
- 2C computation:
  - 2C(X) = 1C(X) + 1 (arithmetic "+")
  - 2C(X) by exploring right (LSB) to left (MSB) the X bits



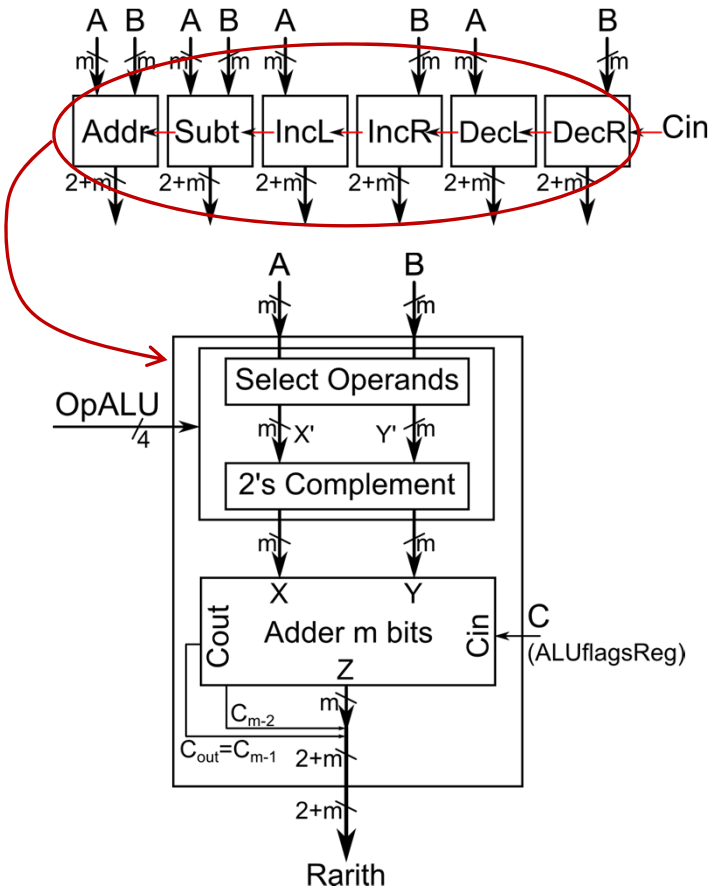

```
Toggle = 0;
For i = 0 to m-1 do
  If Toggle=0 then 2C(Xi)=Xi
  else 2C(Xi)=not(Xi);
  If Xi=1 then Toggle=1;
End for;
```

VSμP : ALU Components

P3.1

UAB Universitat Autònoma de Barcelona

ALU: Arithmetic Operators: Configurable arithmetic block

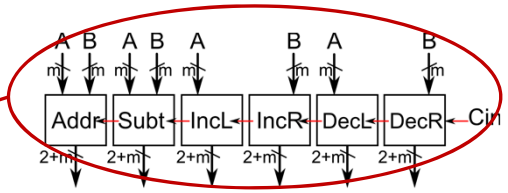


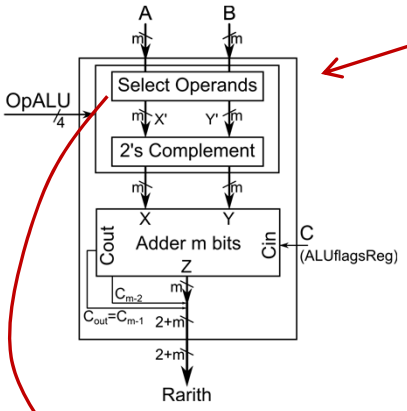
OpALU (Op3 Op2 Op1 Op0)	Nemonic	Operation	OperationC2	X', Y'	C2in	X	Y
0 0 0 0	NoOp	No op.	No op.	-, -	-	-	-
0 0 0 1	Addr	A + B	A + B	A, B	-	X'	Y'
0 0 1 0	Subt	A - B	A + (-B)	A, B	C2(B)→ C2(Y')	X'	C2(Y')
0 0 1 1	IncL	A + "1"	A + 1	A, 1	-	X'	Y'
0 1 0 0	IncR	B + "1"	1 + B	1, B	-	X'	Y'
0 1 0 1	DecL	A - "1"	A + (- 1)	A, 1	C2(1)→ C2(Y')	X'	C2(Y')
0 1 1 0	DecR	B - "1"	(- 1) + B	1, B	C2(1)→ C2(X')	C2(X')	Y'
0 1 1 1	lAnd	A and B	A and B	- -	-	-	-
1 0 0 0	lOr	A or B	A or B	- -	-	-	-
1 0 0 1	lNot	not A	not A	- -	-	-	-
1 0 1 0	ShtL	ShtL A	ShtL A	- -	-	-	-
1 0 1 1	RotR	RotR A	RotR A	- -	-	-	-
1 1 0 0	GoL	A	A	- -	-	-	-
1 1 0 1	GoR	B	B	- -	-	-	-
1 1 1 0	Out0	"0…0"	"0…0"	- -	-	-	-
1 1 1 1	Out1	"1…1"	"1…1"	- -	-	-	-

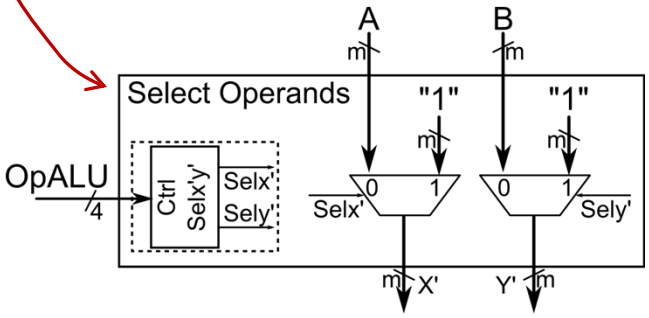
VSμP : ALU Components

UAB
Universitat Autònoma de Barcelona

ALU: Arithmetic Operators







$$SelX' = \overline{Op_1} \cdot \overline{Op_0} + Op_2 \cdot Op_1$$

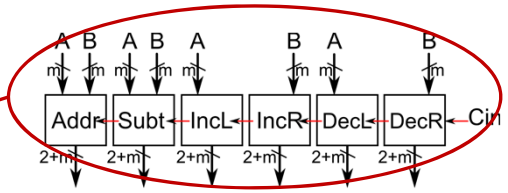
$$SelY' = Op_2 \cdot Op_0 + Op_1 \cdot Op_0$$

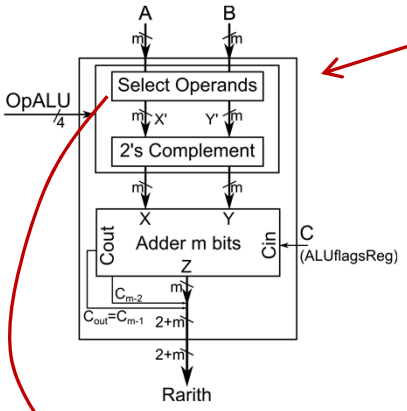
OpALU Op3 Op2 Op1 Op0	Nemonic	Operation	OperationC2	X', Y'		C2in	X	Y	
0 0 0 0	NoOp	No op.	No op.	- -		-	-	-	
0 0 0 1	Addr	A + B	A + B	A, B		-	X'	Y'	
0 0 1 0	Subt	A - B	A + (-B)	A, B		C2(B)→ C2(Y')	X'	C2(Y')	
0 0 1 1	IncL	A + 1	A + 1	A, 1		-	X'	Y'	
0 1 0 0	IncR	B + 1	1 + B	1, B		-	X'	Y'	
0 1 0 1	DecL	A - 1	A + (- 1)	A, 1		C2(1)→ C2(Y')	X'	C2(Y')	
0 1 1 0	DecR	B - 1	(- 1) + B	1, B		C2(1)→ C2(X')	C2(X')	Y'	
0 1 1 1	lAnd	A and B	A and B	- -		-	-	-	
1 0 0 0	lOr	A or B	A or B	- -		-	-	-	
1 0 0 1	lNot	not A	not A	- -		-	-	-	
1 0 1 0	ShtL	ShtL A	ShtL A	- -		-	-	-	
1 0 1 1	RotR	RotR A	RotR A	- -		-	-	-	
1 1 0 0	GoL	A	A	- -		-	-	-	
1 1 0 1	GoR	B	B	- -		-	-	-	
1 1 1 0	Out0	"0...0"	"0...0"	- -		-	-	-	
1 1 1 1	Out1	"1...1"	"1...1"	- -		-	-	-	

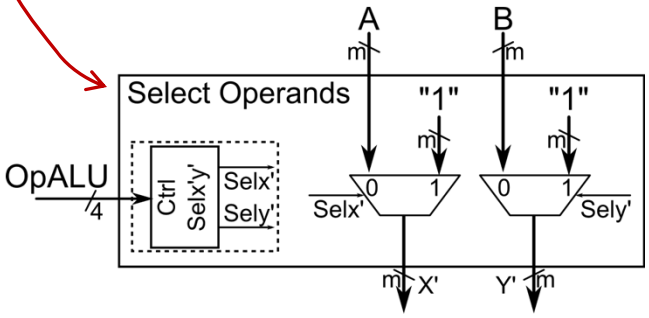
VSμP : ALU Components

UAB
Universitat Autònoma de Barcelona

ALU: Arithmetic Operators







$$SelX' = \overline{Op_1} \cdot \overline{Op_0} + Op_2 \cdot Op_1$$

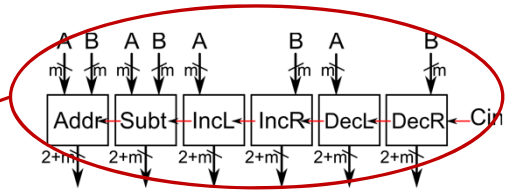
$$SelY' = Op_2 \cdot Op_0 + Op_1 \cdot Op_0$$

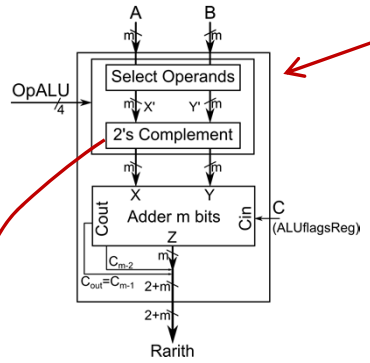
OpALU Op3 Op2 Op1 Op0	Nemonic	Operation	OperationC2	X', Y'	SelX' SelY'	C2in	X	Y	SelC2 SelX SelY
0 0 0 0	NoOp	No op.	No op.	- -	- -	-	-	-	- - -
0 0 0 1	Addr	A + B	A + B	A, B	0 0	-	X'	Y'	- 0 0
0 0 1 0	Subt	A - B	A + (-B)	A, B	0 0	C2(B)→ C2(Y')	X'	C2(Y')	0 0 1
0 0 1 1	IncL	A + 1	A + 1	A, 1	0 1	-	X'	Y'	- 0 0
0 1 0 0	IncR	B + 1	1 + B	1, B	1 0	-	X'	Y'	- 0 0
0 1 0 1	DecL	A - 1	A + (- 1)	A, 1	0 1	C2(1)→ C2(Y')	X'	C2(Y')	0 0 1
0 1 1 0	DecR	B - 1	(- 1) + B	1, B	1 0	C2(1)→ C2(X')	C2(X')	Y'	1 1 0
0 1 1 1	lAnd	A and B	A and B	- -	- -	-	-	-	- - -
1 0 0 0	lOr	A or B	A or B	- -	- -	-	-	-	- - -
1 0 0 1	lNot	not A	not A	- -	- -	-	-	-	- - -
1 0 1 0	ShtL	ShtL A	ShtL A	- -	- -	-	-	-	- - -
1 0 1 1	RotR	RotR A	RotR A	- -	- -	-	-	-	- - -
1 1 0 0	GoL	A	A	- -	- -	-	-	-	- - -
1 1 0 1	GoR	B	B	- -	- -	-	-	-	- - -
1 1 1 0	Out0	"0…0"	"0…0"	- -	- -	-	-	-	- - -
1 1 1 1	Out1	"1…1"	"1…1"	- -	- -	-	-	-	- - -

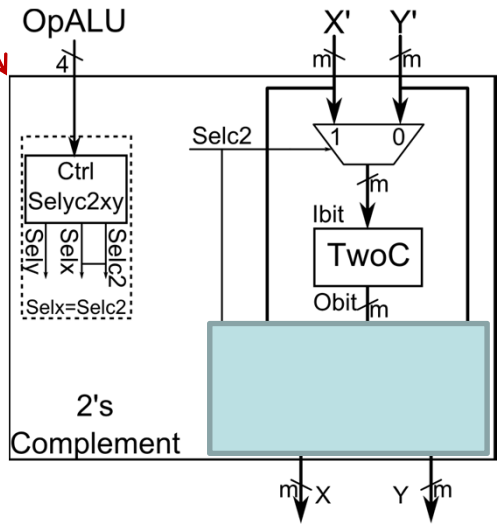
VSμP : ALU Components

UAB Universitat Autònoma de Barcelona

ALU: Arithmetic Operators







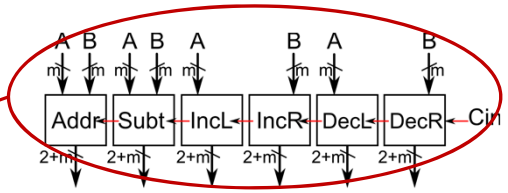
OpALU Op3 Op2 Op1 Op0	Nemonic	Operation	OperationC2	X', Y'	SelY' SelX'	C2in	X	Y
0 0 0 0	NoOp	No op.	No op.	- -	- -	-	-	-
0 0 0 1	Addr	A + B	A + B	A, B	0 0	-	X'	Y'
0 0 1 0	Subt	A - B	A + (-B)	A, B	0 0	C2(B)→ C2(Y')	X'	C2(Y')
0 0 1 1	IncL	A + 1	A + 1	A, 1	0 1	-	X'	Y'
0 1 0 0	IncR	B + 1	1 + B	1, B	1 0	-	X'	Y'
0 1 0 1	DecL	A - 1	A + (- 1)	A, 1	0 1	C2(1)→ C2(Y')	X'	C2(Y')
0 1 1 0	DecR	B - 1	(- 1) + B	1, B	1 0	C2(1)→ C2(X')	C2(X')	Y'
0 1 1 1	lAnd	A and B	A and B	- -	- -	-	-	-
1 0 0 0	lOr	A or B	A or B	- -	- -	-	-	-
1 0 0 1	lNot	not A	not A					
1 0 1 0	ShtL	ShtL A	ShtL A					
1 0 1 1	RotR	RotR A	RotR A					
1 1 0 0	GoL	A	A					
1 1 0 1	GoR	B	B					
1 1 1 0	Out0	"0…0"	"0…0"	- -	- -	-	-	-
1 1 1 1	Out1	"1…1"	"1…1"	- -	- -	-	-	-

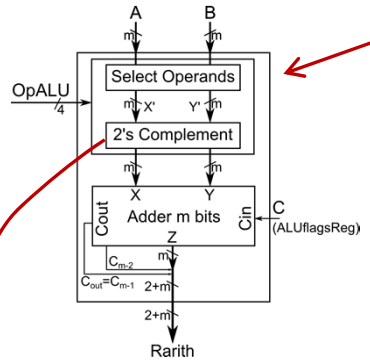
Selc2 $\stackrel{?}{=}$ Selx = ?
Selx = ?
Sely = ?

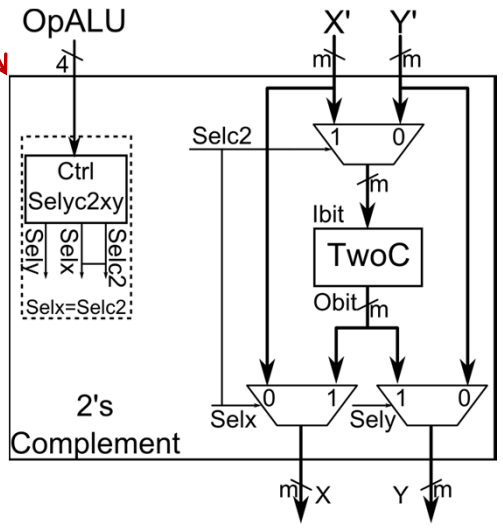
VSμP : ALU Components

UAB Universitat Autònoma de Barcelona

ALU: Arithmetic Operators







OpALU Op3 Op2 Op1 Op0	Nemonic	Operation	OperationC2	X', Y'	SelX' SelY'	C2in	X	Y	SelC2 SelX SelY
0 0 0 0	NoOp	No op.	No op.	- -	- -	-	-	-	- - -
0 0 0 1	Addr	A + B	A + B	A, B	0 0	-	X'	Y'	- 0 0
0 0 1 0	Subt	A - B	A + (-B)	A, B	0 0	C2(B)→ C2(Y')	X'	C2(Y')	0 0 1
0 0 1 1	IncL	A + 1	A + 1	A, 1	0 1	-	X'	Y'	- 0 0
0 1 0 0	IncR	B + 1	1 + B	1, B	1 0	-	X'	Y'	- 0 0
0 1 0 1	DecL	A - 1	A + (- 1)	A, 1	0 1	C2(1)→ C2(Y')	X'	C2(Y')	0 0 1
0 1 1 0	DecR	B - 1	(- 1) + B	1, B	1 0	C2(1)→ C2(X')	C2(X')	Y'	1 1 0
0 1 1 1	lAnd	A and B	A and B	- -	- -	-	-	-	- - -
1 0 0 0	lOr	A or B	A or B	- -	- -	-	-	-	- - -
1 0 0 1	lNot	not A	not A						- - -
1 0 1 0	ShtL	ShtL A	ShtL A						- - -
1 0 1 1	RotR	RotR A	RotR A						- - -
1 1 0 0	GoL	A	A						- - -
1 1 0 1	GoR	B	B						- - -
1 1 1 0	Out0	"0…0"	"0…0"	- -	- -	-	-	-	- - -
1 1 1 1	Out1	"1…1"	"1…1"	- -	- -	-	-	-	- - -

Selc2 $\stackrel{?}{=}$ Selx = ?
Selx = ?
Sely = ?

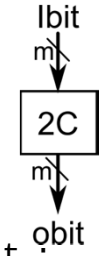
VSμP : ALU Components

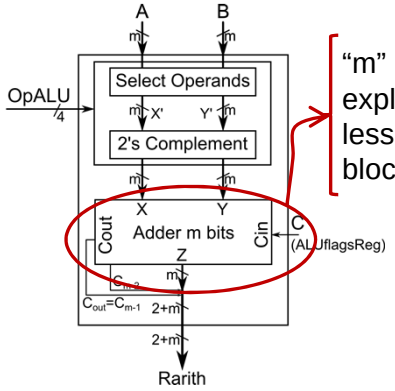
P3.1

ALU: Two's Complement Block

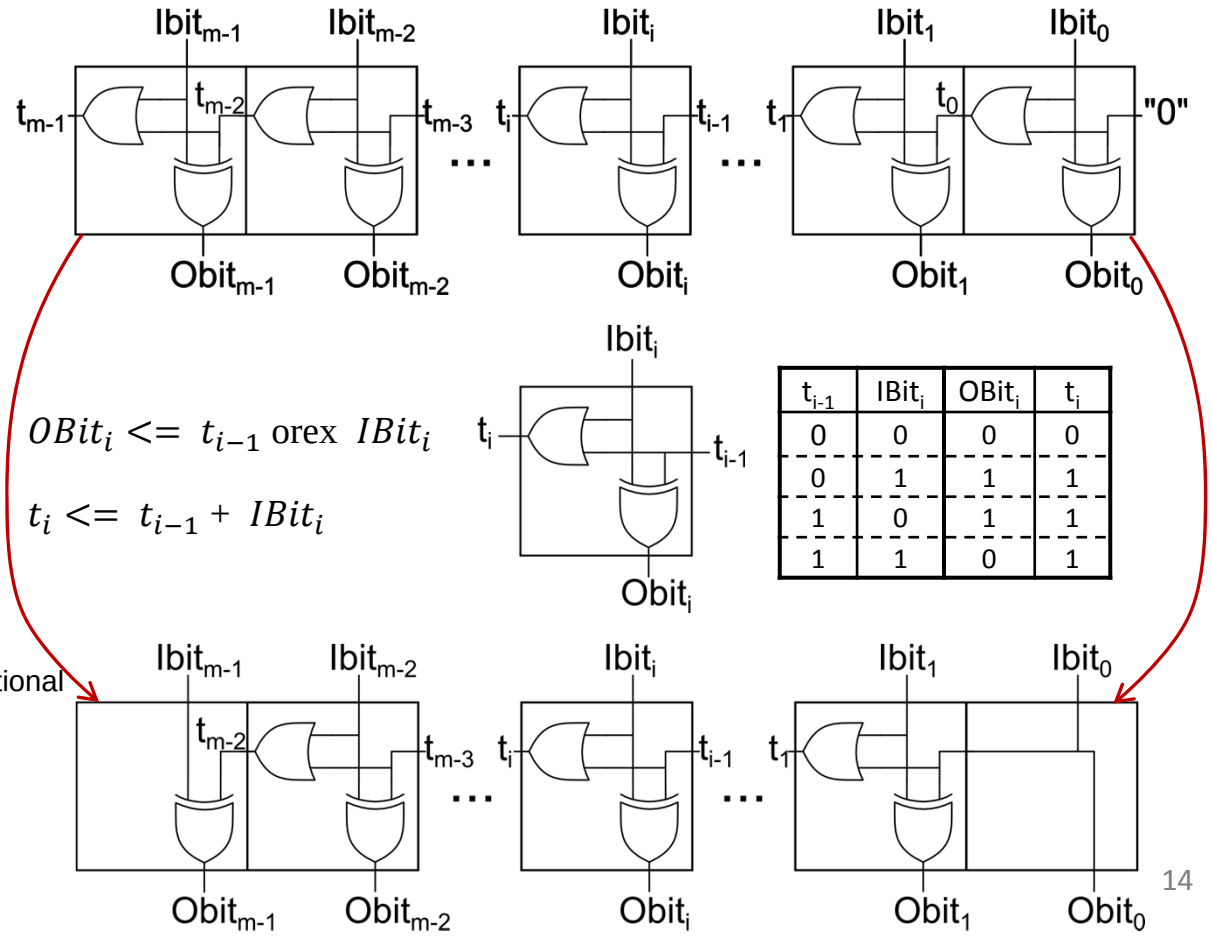
2's Complement (2C)

```
T-1 = 0;
For i = 0 to m-1 do
  If Ti-1=0 then
      OBiti = IBiti;
    else
      OBiti = not Ibiti;
  End if;
  Ti = Ibiti or Ti-1;
End for;
```





"m" bits Adder already explained in previous lessons as a combinational block.



$OBit_i <= t_{i-1}$ orex $IBit_i$

$t_i <= t_{i-1} + IBit_i$

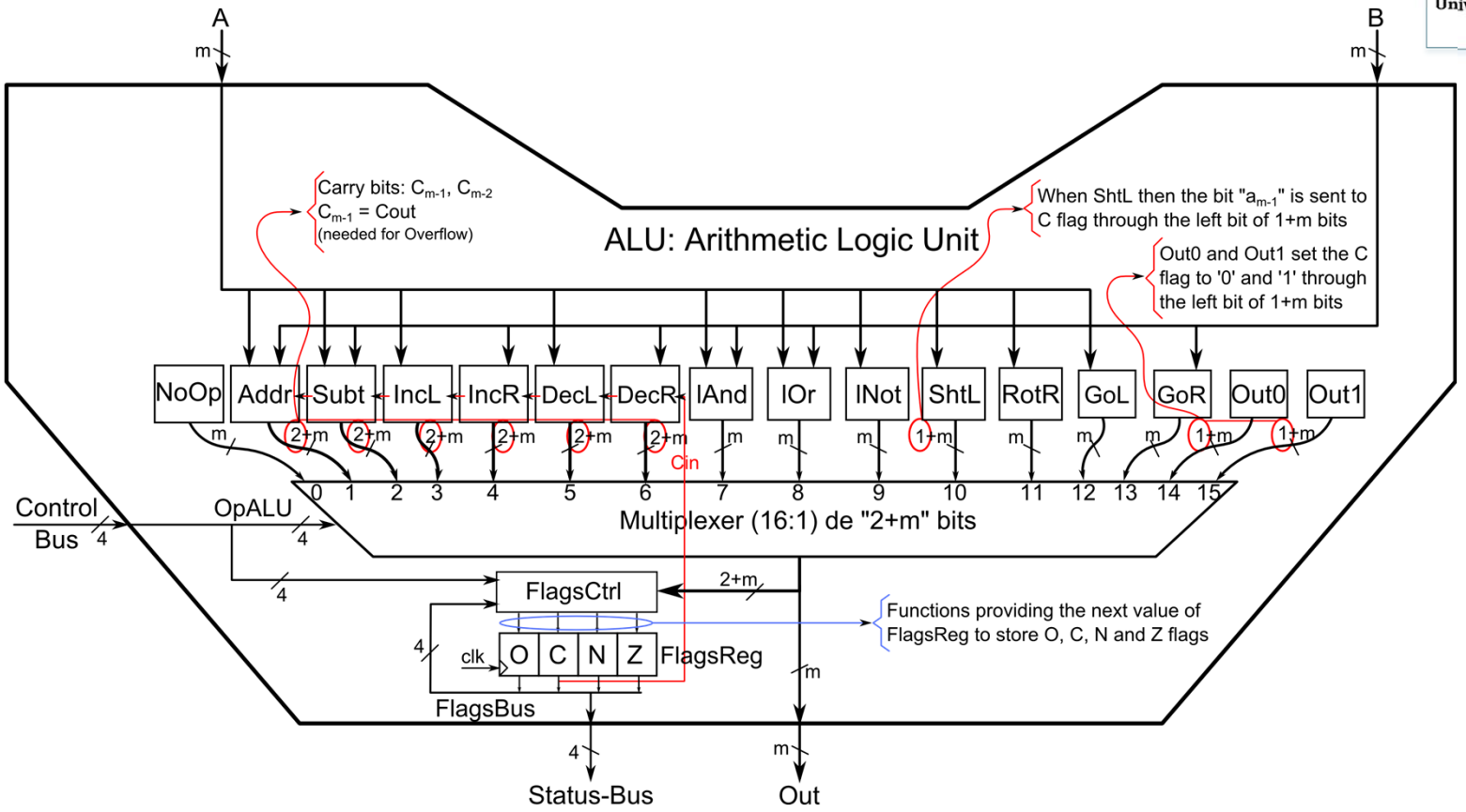
t_{i-1}	$IBit_i$	$OBit_i$	t_i
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

VSμP : ALU Components

P3.1

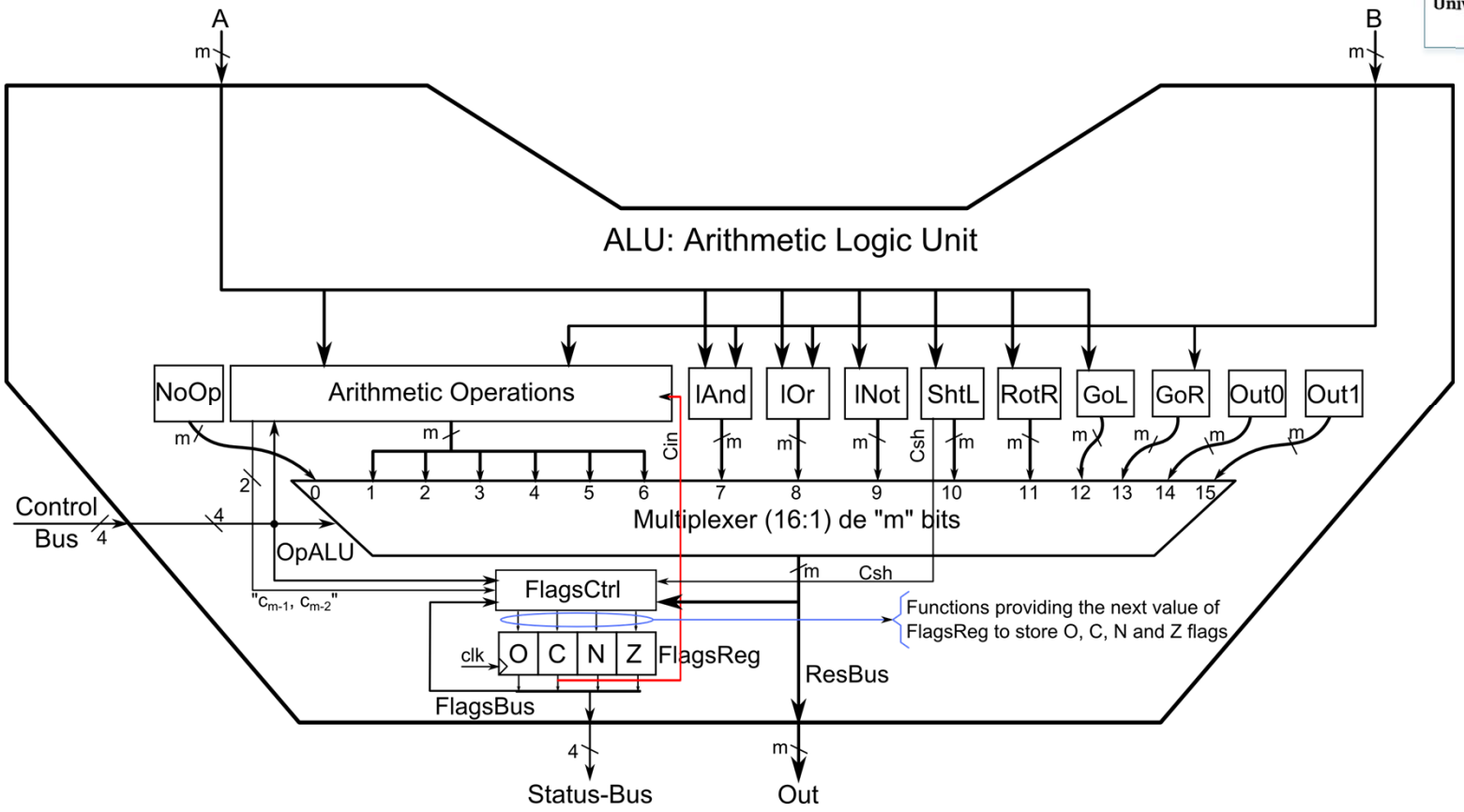
UAB Universitat Autònoma de Barcelona

ALU: G[al] Structure and Components (previous version)



VSμP : ALU Components

ALU: G[al] Structure and Components (optimized)



Summary

ALU: Design Summary

Session summary

- PU and ALU have been updated: new flag “Overflow” and operations mnemonics
- ALU has been progressively designed at different levels:
  - ✓ Blocks, logic functions and gates
  - ✓ Hardware description pseudo-codes
- Non arithmetic operations in brief
- Arithmetic operations in 2C optimized into a configurable block:
  - ✓ Operands selection
  - ✓ 2’s complement (2C) application
  - ✓ All operations using 2C and a Full-Adder
- The final ALU with optimized arithmetic operations