CSU22022 Computer Architecture I

Fourteenth Lecture - Function Unit

Michael Manzke

2023-2024

Trinity College Dublin

$G = A + Y + C_{in}$

S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	$all0's$	$G = A$	$G = A + 1$
0	1	B	$G = A + B$	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$
1	1	$all1's$	$G = A - 1$	$G = A$

Table 1: The arithmetic micro-ops can be implemented using the carry-in C_{in} and two select inputs S_1, S_0, which condition the B input to deliver Y to the full-adder computing: $G = A + Y + C_{in}$

Arithmetic Circuit

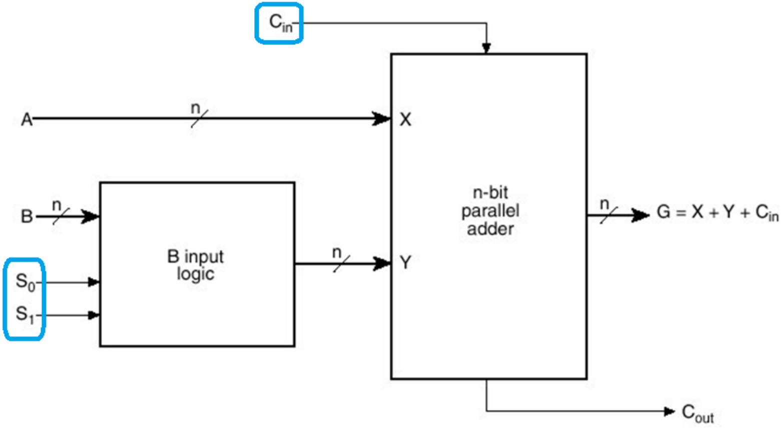
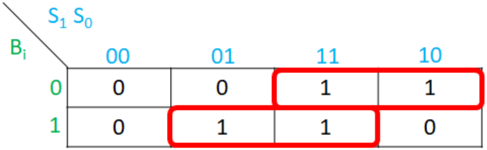


Figure 1: The signals S_1, S_0, C_{in} determine the arithmetic micro-ops accoding to Table 1

Karnaugh Map $Y_i = S_0 B_i + S_1 \overline{B_i}$

S_1	S_0	B_i	Y_i	
0	0	0	0	$all 0's$
0	0	1	0	
0	1	0	0	B
0	1	1	1	
1	0	0	1	$\overline{B}$
1	0	1	0	
1	1	0	1	$all 1's$
1	1	1	1	



B_i \ $S_1 S_0$	00	01	11	10
0	0	0	1	1
1	0	1	1	0

One Bit Slice $Y_i = S_0 B_i + S_1 \overline{B_i}$

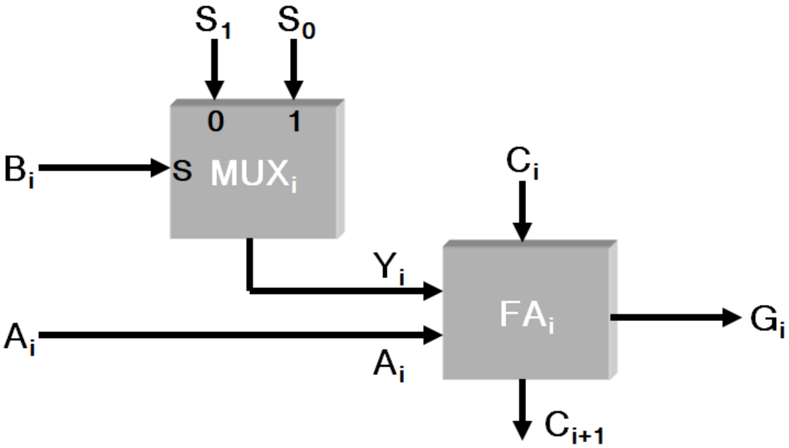


Figure 2: Thus a 2:1 MUX controlled by B_i can efficiently generates Y_i

One Bit Slice Blogic

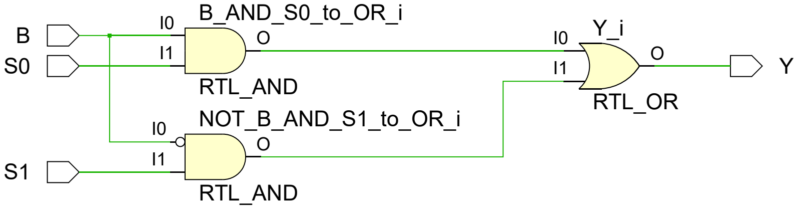


Figure 3: This logic implements the 2:1 MUX

8-bit BLogic

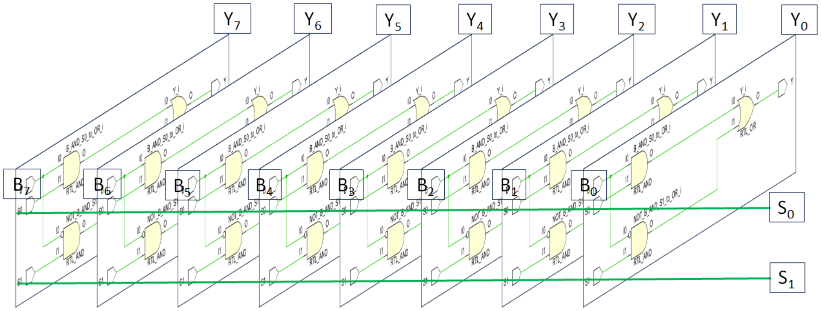


Figure 4: This example show 8 instantiations of the B-Logic, but we need 32-bit.

8-bit Ripple Carry Adder

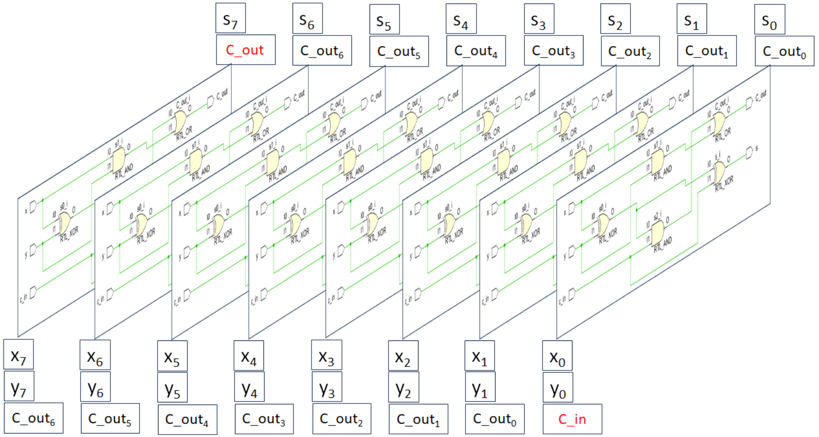
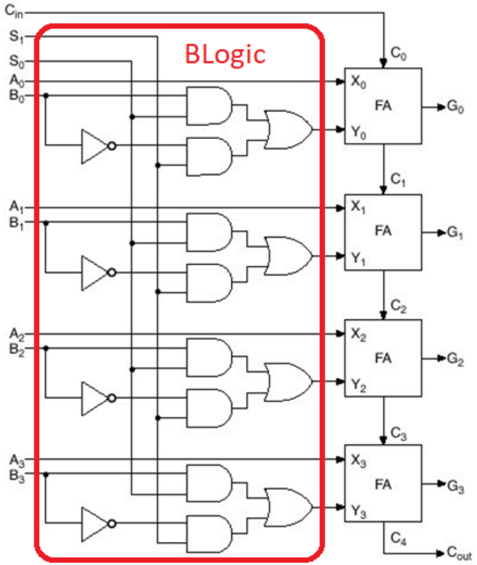
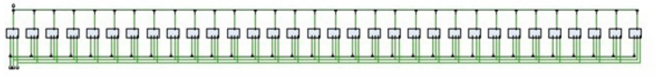


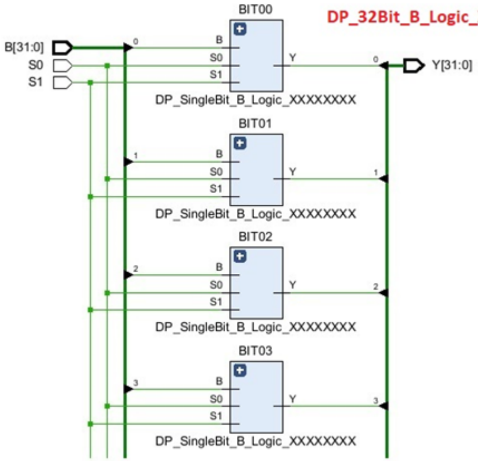
Figure 5: This example show 8 instantiations of the Full-Adder, but we need 32-bit.

4-Bit Arithmetic Circuit

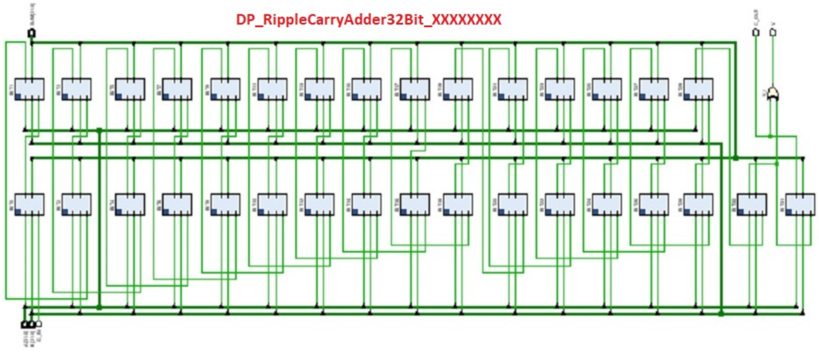


32-Bit B-Logic

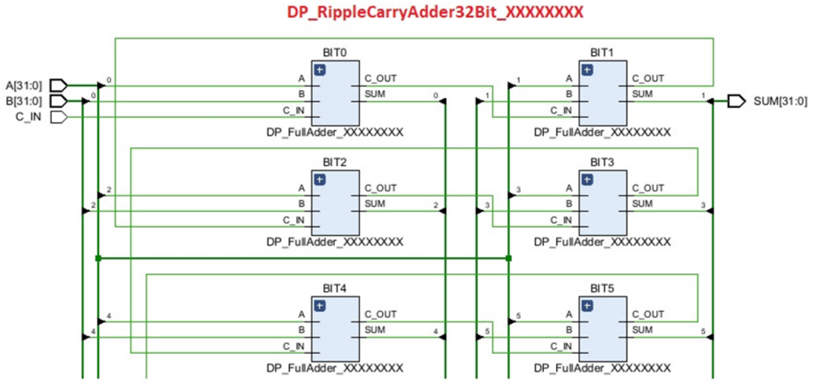




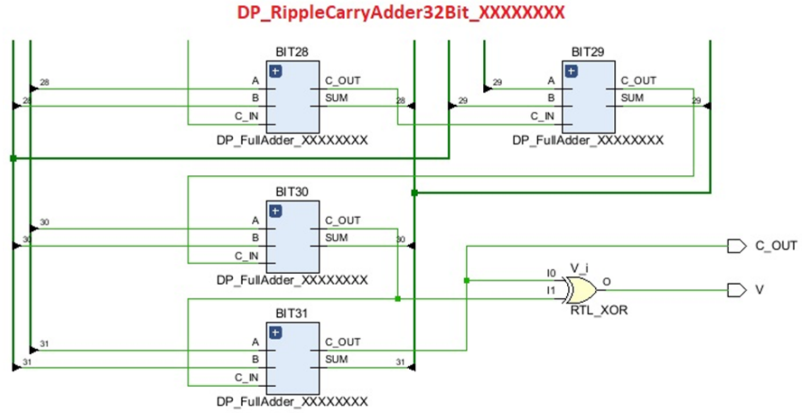
32-Bit Ripple Carry Adder



32-Bit Ripple Carry Adder



32-Bit Ripple Carry Adder



Logic Circuit

S_1	S_0		
0	0	G=A$\wedge$B	AND
0	1	G=A$\vee$B	OR
1	0	G=A$\oplus$B	XOR
1	1	G=$\overline{A}$	NOT

Table 2: Also, the logic function are selected by input S_1 and S_0

Logic Circuit Implemented with a 4:1 MUX

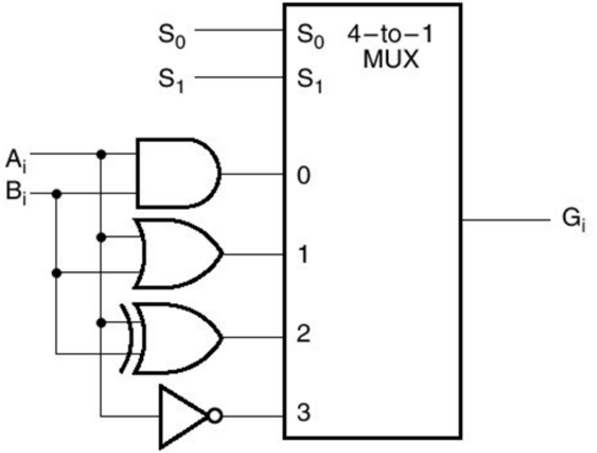


Figure 6: One bit slice of the logic unit

DP_SingleBit_LogicCircuit_XXXXXXXX

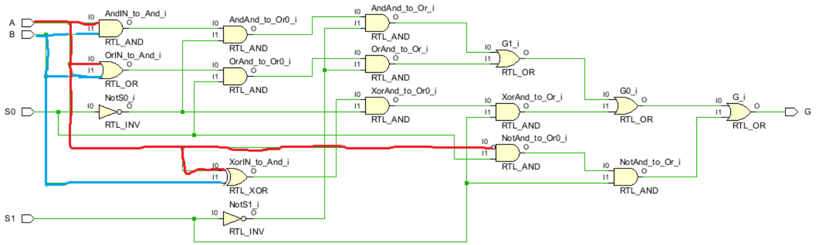


Figure 7: One bit slice of the logic unit

8-bit Logic Unit

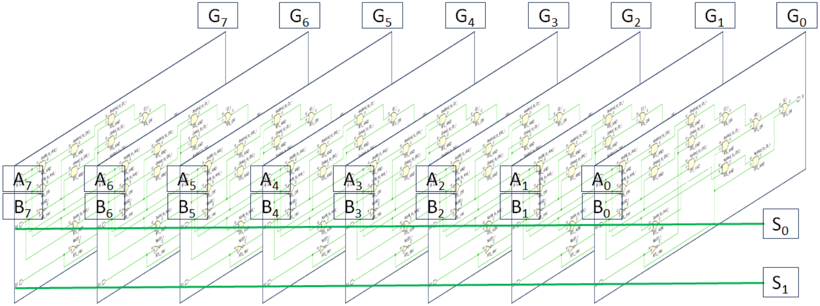
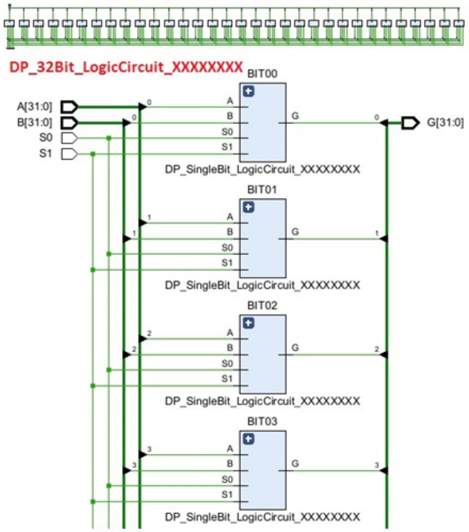


Figure 8: This example show 8 instantiations of the one bit logic unit slice, but we need 32-bit.

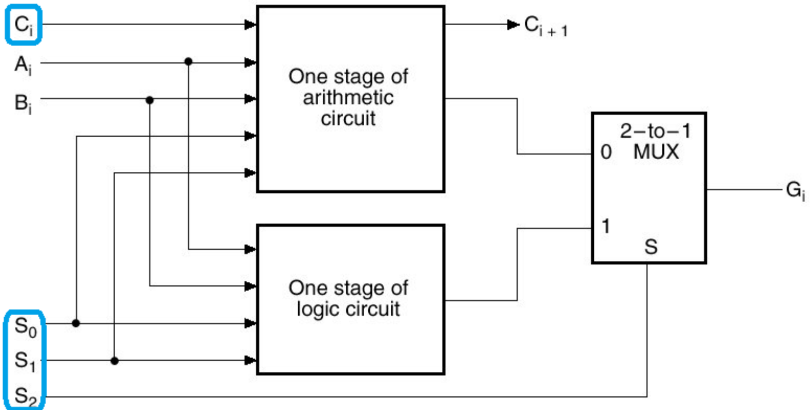
Logic Circuit Implementation with a 4:1 MUX


DP_32Bit_LogicCircuit_XXXXXXXX
A[31:0]
B[31:0]
S0
S1
G[31:0]
BIT00
BIT01
BIT02
BIT03
A
B
S0
S1
G
DP_SingleBit_LogicCircuit_XXXXXXXX

ALU (Arithmetic/Logic)

We next use an additional 2:1 MUX controlled by S_2 to select either the arithmetic output bit or the logic output bit as shown on the next slide.

One Bit slice ALU



ALU Operations

S_2	S_1	S_0	C_{in}		
0	0	0	0	G=A	TRANSFER
0	0	0	1	G=A+1	INCREMENT
0	0	1	0	G=A+B	ADD
0	0	1	1	G=A+B+1	ADD WITH C
0	1	0	0	G=A$\overline{B}$	A plus 1's COMPLEMENT
0	1	0	1	G=A$\overline{B}$+1	SUBTRACT
0	1	1	0	G=A-1	DECREMENT
0	1	1	1	G=A	TRANSFER
1	0	0	-	G=A$\wedge$B	AND
1	0	1	-	G=A$\vee$B	OR
1	1	0	-	G=A$\oplus$B	XOR
1	1	1	-	G=$\overline{A}$	NOT

4-Bit SR/SL Shifter Unit

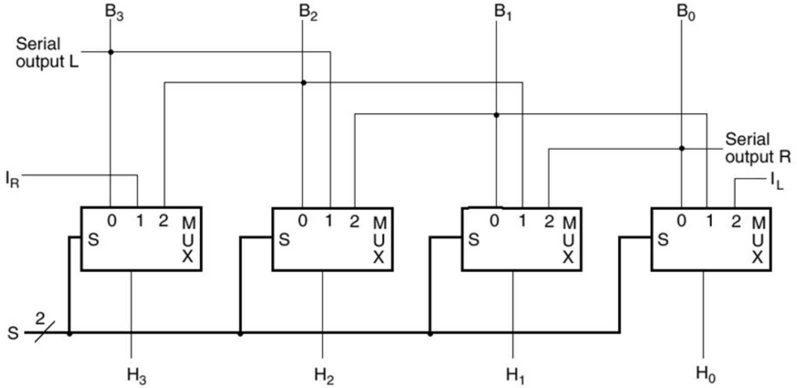
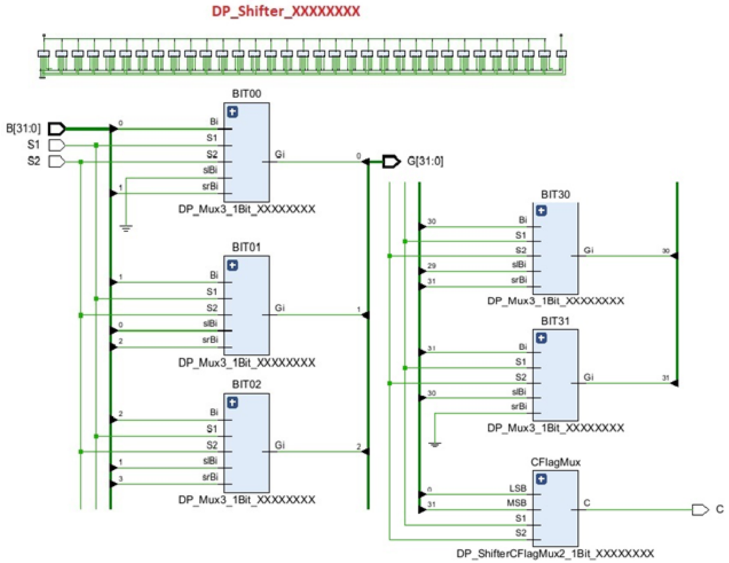


Figure 9: For speed of execution the shifter unit is always implemented as a combinational circuit based on a MUX.

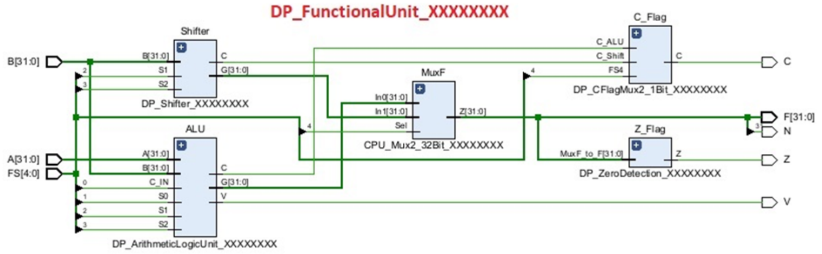
SR/SL Shifter Operations

S_2	S_1	
0	0	F=B
0	1	F=srB
1	0	F=slB

32-Bit SR/SL Shifter Unit (Including C-Flag)



Functional Unit VHDL Schematic


DP_FunctionalUnit_XXXXXXXX
B[31:0]
A[31:0]
FS[4:0]
Shifter
B[31:0]
S1
S2
C
G[31:0]
DP_Shifter_XXXXXXXX
ALU
A[31:0]
B[31:0]
C_IN
S0
S1
S2
C
G[31:0]
V
DP_ArithmeticLogicUnit_XXXXXXXX
MuxF
In0[31:0]
In1[31:0]
Sel
Z[31:0]
CPU_Mux2_32Bit_XXXXXXXX
C_Flag
C_ALU
C_Shift
FS4
C
DP_CFlagMux2_1Bit_XXXXXXXX
Z_Flag
MuxF_to_F[31:0]
Z
DP_ZeroDetection_XXXXXXXX
C
F[31:0]
N
Z
V