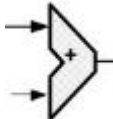


Identificación	Nombre	Símbolo	Descripción	Tabla de verdad	Simplificación	Funciones Lógicas	Código	Nombre del archivo																																
4	Sumador		Unidad que realiza la suma de dos números de 32 bits	Utiliza half Adders: <table><tr><th colspan="2">Entradas</th><th colspan="2">Salidas</th></tr><tr><th>A</th><th>B</th><th>S</th><th>C</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	Entradas		Salidas		A	B	S	C	0	0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	$S =$ <table><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr></table> $C =$ <table><tr><td>0</td><td>0</td><td>1</td><td>0</td></tr></table> La simplificación se da mediante el método de mapas de Karnaugh	0	1	0	1	0	0	1	0	$S = A \oplus B$ $C = A \cdot B$ Ya con las ecuaciones simplificadas se puede obtener las ecuaciones para los acarrees sin tener que esperar por el acarreo anterior de la siguiente manera. Basado en el hecho que una señal de acarreo será generada en el caso de que las entradas a_i y b_i son ambas 1 o cuando alguna de estas es 1 y el acarreo de entrada es uno. $c_{i+1} = a_i \cdot b_i + (a_i \oplus b_i) \cdot c_i$ $s_i = (a_i \oplus b_i) \oplus c_i$ donde el generador o G es: $G_i = a_i \cdot b_i$ y el propagador o P es: $P_i = a_i \oplus b_i$ $c_{i+1} = G_i + P_i \cdot c_i$ $s_i = P_i \oplus c_i$ Usando estas ecuaciones vemos como no es necesario esperar por la generación del acarreo anterior. Esto aplicado a un sumador de 4 bits se vería de la siguiente manera: $c1 = G_0 + P_0 \cdot c0$ $c2 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c0$ $c3 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot c0$ $c4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot c0$	<pre>module thirtyTwoBitsFullAdder(input [31:0] a, input [31:0] b, output [31:0] s, input c0, output cout); wire c1,c2,c3,c4,c5,c6,c7; fourbitFullAdder fbfa1 (.a(a[3:0]), .b(b[3:0]),.c0(c0),.c4(c1),.s(s[3:0])); fourbitFullAdder fbfa2 (.a(a[7:4]),.b(b[7:4]),.c0(c1),.c4(c2),.s(s[7:4])); fourbitFullAdder fbfa3 (.a(a[11:8]),.b(b[11:8]),.c0(c2),.c4(c3),.s(s[11:8])); fourbitFullAdder fbfa4 (.a(a[15:12]),.b(b[15:12]),.c0(c3),.c4(c4),.s(s[15:12])); fourbitFullAdder fbfa5 (.a(a[19:16]),.b(b[19:16]),.c0(c4),.c4(c5),.s(s[19:16])); fourbitFullAdder fbfa6 (.a(a[23:20]), .b(b[23:20]),.c0(c5),.c4(c6),.s(s[23:20])); fourbitFullAdder fbfa7 (.a(a[27:24]), .b(b[27:24]),.c0(c6),.c4(c7),.s(s[27:24])); fourbitFullAdder fbfa8 (.a(a[31:28]), .b(b[31:28]),.c0(c7),.c4(cout),.s(s[31:28])); endmodule</pre>	Sumador32bits_ChavarriaOrtegaOrtizVillalobos.v
Entradas		Salidas																																						
A	B	S	C																																					
0	0	0	0																																					
0	1	1	0																																					
1	0	1	0																																					
1	1	0	1																																					
0	1	0	1																																					
0	0	1	0																																					

