

Aluno do Embarcotech\_37 no IFMA

Nome: Manoel Felipe Costa Furtado

Matrícula: 20251RSE.MTC0086

Residência Profissional em FPGA – Turma DELTA - 2025.2

Atividade – Referente ao capítulo 02 da unidade 02

Tema do Capítulo – Fundamentos de FPGA

Prazo dia 07/09/2025 as 23:59

Objetivo: Desenvolver um código utilizando Icarus Verilog que represente um subtrator completo.

Enunciado:

Desenvolva um código utilizando o Icarus Verilog que represente um subtrator completo (“subtratorcompleto”). A aplicação deverá ser composta por três entradas (a, b, cin) e duas saídas (s, cout). Faça a compilação, simulação e visualização das formas de onda do projeto “subtratorcompleto” no VSCode, utilizando o Icarus Verilog e verifique os resultados da simulação no GTKWave. A tabela 1 mostra os valores de entrada e saída do circuito subtrator completo.

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Tabela 1. Tabela Verdade do circuito digital Subtrator Completo  
Fonte: Próprio Autor (2025)

A Figura 1 mostra o circuito digital do subtrator completo com as indicações das entradas “A”, “B” e “Cin” e as saídas “S” e “Cout”

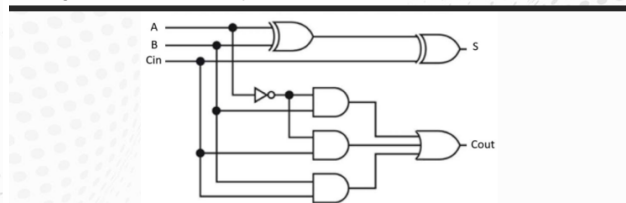


Figura 1. Circuito digital do subtrator completo

Fonte: Adaptado de Eletrônica Digital – Aula 08 para Faculdade (2023), Slideshare.

Link da Figura 01: <https://pt.slideshare.net/slideshow/eletronica-digital-aula-08-para-faculdade/271949497>

## Instruções:

Seguir os passos descritos no ebook (“etapas”):

- 1) Escreva Os códigos do subtratorcompleto e subtratorcompleto\_tb (bancada de teste) em arquivos de texto no VSCode e salve-os com extensão verilog (.v);
- 2) No terminal do VSCode, certifique-se de que a linha de diretório está apontando para a pasta na qual foram salvos os arquivos “.v” (Módulo Principal e Bancada de Teste);
- 3) Faça a Compilação dos arquivos subtratorcompleto (módulo principal) e subtratorcompleto\_tb (bancada de teste) com:
  - iverilog -o subtratorcompleto.vvp subtratorcompleto.v
  - iverilog -o subtratorcompleto\_tb.vvp subtratorcompleto\_tb.v
- 4) Execute o comando dir no terminal do VSCode para verificar se os arquivos de saída .vvp foram gerados;
- 5) No terminal do VSCode, faça a Simulação do projeto com o arquivo subtratorcompleto\_tb.vvp (arquivo de saída da compilação) com:
  - vvp subtratorcompleto\_tb.vvp
- 6) Execute o comando dir no terminal do VSCode para verificar se o arquivo de saída .vcd foi gerado;
- 7) No terminal do VSCode, execute o GTKWave para a Visualização das Formas de Ondas a partir do arquivo .vcd (arquivo de saída da simulação) com:
  - gtkwave subtratorcompleto.vcd
- 8) Maximize o programa GTKWave aberto na tela;
- 9) No Canto superior esquerdo, faça a expansão de subtratorcompleto\_tb, clicando no + e então selecione uut;
- 10) À esquerda e ao centro, selecione simultaneamente a, b, cin, s, cout;
- 11) À esquerda, na parte inferior, selecione o botão insert;
- 12) Na parte superior esquerda, clique continuamente no botão zoom - (lupa -) até que as formas de onda fiquem visíveis;
- 13) Por fim, clique em cada combinação de entradas e saídas na área da forma de onda e será apresentada a combinação das entradas e saídas para análise do projeto e identificação de eventuais inconsistências.

## Solução

- GitHub: [Segunda Fase FPGA/Unidade 02/Cap 02](#)
- Link do Vídeo: <https://youtu.be/pp68TilUHVg>
- Link do Código Completo: [Unid 02 Cap 02.zip](#)

### 1) Explicação da justificativa do nome "Subtrator Completo"

O nome "Subtrator Completo" (em inglês, Full Subtractor) vem da sua capacidade de ser um bloco de construção "completo" para subtrair números com múltiplos bits. Para entender isso, precisamos compará-lo com um "Meio-Subtrator" (Half Subtractor).

#### Meio-Subtrator (Half Subtractor):

- Este é o circuito mais simples possível para subtração. Ele tem apenas **duas** entradas (A e B) e duas saídas (Diferença e Empréstimo).
- Ele consegue calcular  $A - B$ .
- Problema: Ele não tem uma entrada para considerar um "empréstimo" vindo de uma operação anterior. Por exemplo, ao subtrair  $1010 - 0111$ , quando vamos calcular o segundo bit da direita ( $1 - 1$ ), precisamos considerar que o primeiro bit ( $0 - 1$ ) "pediu emprestado". O Meio-Subtrator não consegue lidar com esse "empréstimo que vem de antes".

#### Subtrator Completo (Full Subtractor):

- Ele é "completo" porque resolve o problema do Meio-Subtrator. Ele possui três entradas: A, B e Cin (*Carry In*, que aqui funciona como **Borrow In** ou Empréstimo de Entrada).
- Essa terceira entrada, Cin, é exatamente o elo que faltava. Ela representa o empréstimo solicitado pela coluna de bits anterior (menos significativa).
- Por causa disso, podemos conectar ("cascatear") vários Subtratores Completos para subtrair números de qualquer tamanho (4 bits, 8 bits, 32 bits, etc.). Cada Cout (Empréstimo de Saída) de um estágio se conecta ao Cin do próximo estágio, formando uma "corrente de empréstimo" (*borrow chain*).

A operação fundamental que o circuito realiza é: **Diferença = A - B - Cin.**

- S (Soma/Diferença) é o bit de resultado dessa operação.
- Cout (Empréstimo de Saída) é 1 se precisarmos "pedir emprestado" da próxima coluna (mais significativa) para realizar a operação. A regra é simples: **Cout será 1 sempre que o que estamos subtraindo (B + Cin) for maior que A.**

## 2) Tabela Verdade e Reduzindo o circuito

A	B	Cin	Cálculo (A - B - Cin)	S (Diferença)	Cout (Empréstimo)	Justificativa
0	0	0	$0 - 0 - 0 = 0$	0	0	<b>Cout=0</b> porque A (0) não é menor que B+Cin (0). A diferença é 0.
0	0	1	$0 - 0 - 1 = -1$	1	1	<b>Cout=1</b> porque A (0) é menor que B+Cin (1). Para fazer a conta, pegamos emprestado (nosso A vira 10, ou 2 em decimal). $2 - 1 = 1$ . A diferença é 1
0	1	0	$0 - 1 - 0 = -1$	1	1	<b>Cout=1</b> porque A (0) é menor que B+Cin (1). Pegamos emprestado (A vira 10). $2 - 1 = 1$ . A diferença é 1.
0	1	1	$0 - 1 - 1 = -2$	0	1	<b>Cout=1</b> porque A (0) é menor que B+Cin (2). Pegamos emprestado (A vira 10). $2 - 1 - 1 = 0$ . A diferença é 0.
1	0	0	$1 - 0 - 0 = 1$	1	0	<b>Cout=0</b> porque A (1) não é menor que B+Cin (0). A diferença é 1.
1	0	1	$1 - 0 - 1 = 0$	0	0	<b>Cout=0</b> porque A (1) não é menor que B+Cin (1). A diferença é 0.
1	1	0	$1 - 1 - 0 = 0$	0	0	<b>Cout=0</b> porque A (1) não é menor que B+Cin (1). A diferença é 0.
1	1	1	$1 - 1 - 1 = -1$	1	1	<b>Cout=1</b> porque A (1) é menor que B+Cin (2). Para fazer a conta $1 - 1$ dá 0, mas depois $0 - 1$ requer um empréstimo. O resultado final é 1.

	C	0	1
AB			
00		0 0	1 1
01		1 2	0 3
11		0 6	1 7
10		1 4	0 5

Para a saída S:  $A' B' C + A' B C' + A B' C' + A B C$

Pelo Mapa de Karnaugh não foi possível reduzir o circuito contudo, podemos simplificar usando álgebra de boole.

Passo 1: Reorganizar os termos para agrupar fatores comuns

$$s = (A' B' Cin + A' B Cin') + (A B' Cin' + A B Cin)$$

Passo 2: Fatorar os termos comuns (A' e A)

Colocar A' em evidência no primeiro grupo e A em evidência no segundo grupo.

$$s = A' (B' Cin + B Cin') + A (B' Cin' + B Cin)$$

### Passo 3: Identificar as expressões de XOR e XNOR

- Porta XOR (OU Exclusivo): A definição de  $A \oplus B$  é  $A' B + A B'$ .
- Porta XNOR (NÃO-OU Exclusivo): A definição de  $(A \oplus B)'$  é  $A' B' + A B$ .
- O termo  $(B' Cin + B Cin')$  é exatamente a definição de  $B \oplus Cin$ .
- O termo  $(B' Cin' + B Cin)$  é exatamente a definição da porta XNOR, ou seja,  $(B \oplus Cin)'$ .

### Passo 4: Substituir as expressões de XOR e XNOR na equação

- $s = A' (B \oplus Cin) + A (B \oplus Cin)'$

### Passo 5: Reconhecer a forma final da porta XOR

Seja  $Y = (B \oplus Cin)$ , então  $Y' = (B \oplus Cin)'$ .

- $s = A'Y + AY'$

### Passo 6: Substituir de volta e chegar à conclusão

- $s = A \oplus Y \Rightarrow s = A \oplus (B \oplus Cin)$
- $s = A \oplus B \oplus Cin$

Em Verilog, o operador para XOR é o  $\wedge$ , então a expressão final é  $s = a \wedge b \wedge cin$ ;

Desenho mostrado na figura 1 do enunciado.

Para a saída Cout:

		C	
		0	1
AB	00	0 0	1 1
	01	1 2	1 3
11	11	0 6	1 7
	10	0 4	0 5

$$A' B + A' Cin + B Cin$$

Pelo Mapa de Karnaugh foi possível reduzir o circuito.

Em Verilog,  $cout = (\sim a \& b) \mid (\sim a \& cin) \mid (b \& cin)$ ;

Desenho mostrado na figura 1 do enunciado.

### 3) Código em Verilog – Etapa 01

```
/*
 * Atividade_Uni_02_Cap_02
 * @file subtratorcompleto.v
 * @version 1.0
 * @date 05/09/2025
 * @author Manoel Felipe Costa Furtado
 * @copyright 2025 Manoel Furtado (MIT License) (veja LICENSE.md)
 * @brief Psistema de alarme simplificado
-----

Módulo: subtratorcompleto
Descrição: Representa um subtrator completo de 1 bit.
Entradas: a, b, cin (borrow-in)
Saídas: s (diferença), cout (borrow-out)
-----
*/
module subtratorcompleto(
    input a,
    input b,
    input cin,
    output s,
    output cout
);

    // Lógica para a saída de diferença (S)
    assign s = a ^ b ^ cin;

    // Lógica para a saída de empréstimo (Cout)
    // Cout é '1' quando precisamos "emprestar" do próximo bit.
    // Isso ocorre se (não A e B) ou (não A e Cin) ou (B e Cin).
    assign cout = (~a & b) | (~a & cin) | (b & cin);

endmodule
```

### 4) Código para a Simulação – Etapa 2

```
/*
 * Atividade_Uni_02_Cap_02
 * @file subtratorcompleto_tb.v
 * @version 1.0
 * @date 05/09/2025
 * @author Manoel Felipe Costa Furtado
 * @copyright 2025 Manoel Furtado (MIT License) (veja LICENSE.md)
 * @brief Subtrator Completo
-----

Módulo: tb_subtratorcompleto
Descrição: Testbench para o módulo subtratorcompleto.
```

```

-----
*/

`include "subtratorcompleto.v"

`timescale 1ns / 1ps // Define a unidade de tempo da simulação

module tb_subtratorcompleto;

    // 1. Declaração de sinais
    // Entradas para o DUT (Device Under Test) são do tipo 'reg'
    reg a_tb;
    reg b_tb;
    reg cin_tb;

    // Saídas do DUT são do tipo 'wire'
    wire s_tb;
    wire cout_tb;

    // 2. Instanciação do Módulo a ser testado (DUT)
    // Conecta os sinais do testbench às portas do módulo
    subtratorcompleto DUT (
        .a(a_tb),
        .b(b_tb),
        .cin(cin_tb),
        .s(s_tb),
        .cout(cout_tb)
    );

    // 3. Geração dos estímulos e controle da simulação
    initial begin
        // Configuração para gerar o arquivo de forma de onda (VCD)
        $dumpfile("subtratorcompleto.vcd");
        $dumpvars(0, tb_subtratorcompleto);

        // Mensagem inicial no console
        $display("Iniciando a simulacao do Subtrator Completo...");
        $display("A B Cin | S Cout");
        $display("-----");

        // Casos de teste baseados na tabela verdade
        {a_tb, b_tb, cin_tb} = 3'b000; #10;
        {a_tb, b_tb, cin_tb} = 3'b001; #10;
        {a_tb, b_tb, cin_tb} = 3'b010; #10;
        {a_tb, b_tb, cin_tb} = 3'b011; #10;
        {a_tb, b_tb, cin_tb} = 3'b100; #10;
        {a_tb, b_tb, cin_tb} = 3'b101; #10;
        {a_tb, b_tb, cin_tb} = 3'b110; #10;
        {a_tb, b_tb, cin_tb} = 3'b111; #10;
    end

```

```

// Finaliza a simulação após o último caso de teste
$display("Simulacao concluida.");

$finish;

end

// Monitora as mudanças nos sinais e imprime no console
// a cada mudança, formatando para se parecer com a tabela verdade.
always @(a_tb, b_tb, cin_tb, s_tb, cout_tb) begin
    $monitor("%b %b %b   | %b %b", a_tb, b_tb, cin_tb, s_tb, cout_tb);
end

endmodule

```

## 5) Compilação – Etapas 3 a 6

- iverilog -o subtratorcompleto.vvp subtratorcompleto.v
- iverilog -o subtratorcompleto\_tb.vvp subtratorcompleto\_tb.v
- vvp subtratorcompleto\_tb.vvp

```

1 /*
2  * Atividade_Uni_02_Cap_02
3  * @file subtratorcompleto_tb.v
4  * @version 1.0
5  * @date 05/09/2025
6  * @author Manoel Felipe Costa Furtado
7  * @copyright 2025 Manoel Furtado (MIT License) (veja LICENSE.md)
8  * @brief Subtrator Completo
9
10 -----
11 Módulo: tb_subtratorcompleto
12 Descrição: Testbench para o módulo subtratorcompleto.
13 -----
14 */

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

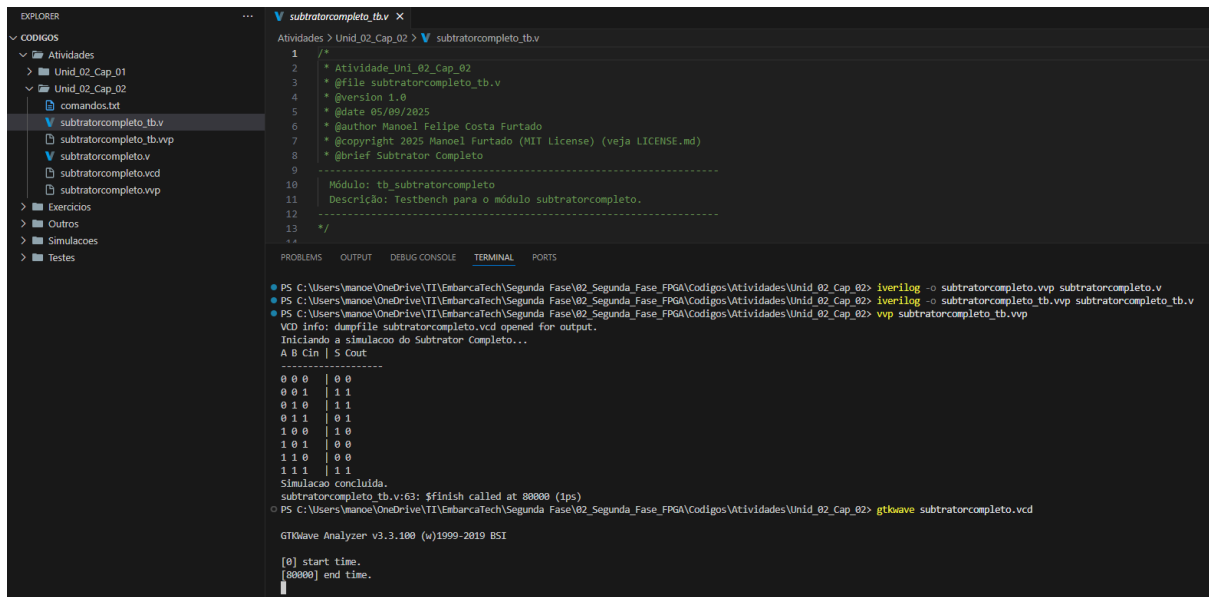
PS C:\Users\maano\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda_Fase_FPGA\Codigos\Atividades\Unid_02_Cap_02> iverilog -o subtratorcompleto.vvp subtratorcompleto.v
PS C:\Users\maano\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda_Fase_FPGA\Codigos\Atividades\Unid_02_Cap_02> iverilog -o subtratorcompleto_tb.vvp subtratorcompleto_tb.v
PS C:\Users\maano\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda_Fase_FPGA\Codigos\Atividades\Unid_02_Cap_02> vvp subtratorcompleto_tb.vvp
VCD info: dumpfile subtratorcompleto.vcd opened for output.
Iniciando a simulacao do Subtrator Completo...
A B Cin | S Cout
-----
0 0 0 | 0 0
0 0 1 | 1 1
0 1 0 | 1 1
0 1 1 | 0 1
1 0 0 | 1 0
1 0 1 | 0 0
1 1 0 | 0 0
1 1 1 | 1 1
Simulacao concluida.
subtratorcompleto_tb.v:63: $finish called at 80000 (tps)
PS C:\Users\maano\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda_Fase_FPGA\Codigos\Atividades\Unid_02_Cap_02>

```



## 6) Simulação – Etapas 7 a 13

- gtkwave subtratorcompleto.vcd



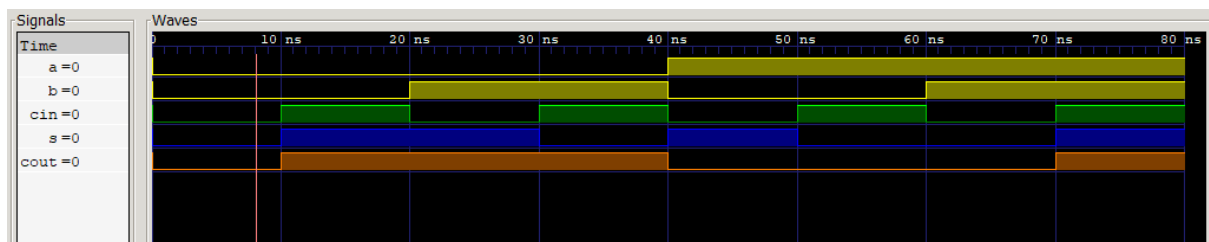
```
1 /*
2  * Atividade_Uni_02_Cap_02
3  * @file subtratorcompleto.tb.v
4  * @version 1.0
5  * @date 05/09/2025
6  * @author Manoel Felipe Costa Furtado
7  * @copyright 2025 Manoel Furtado (MIT License) (veja LICENSE.md)
8  * @brief Subtrator Completo
9  */
10
11 M6dulo: tb_subtratorcompleto
12   Descri76o: Testbench para o m6dulo subtratorcompleto.
13
14 */
```

```
PS C:\Users\manoe\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda Fase_FPGA\C6digos\Atividades\Unid_02_Cap_02> iverilog -o subtratorcompleto.vvp subtratorcompleto.v
PS C:\Users\manoe\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda Fase_FPGA\C6digos\Atividades\Unid_02_Cap_02> iverilog -o subtratorcompleto.tb.vvp subtratorcompleto.tb.v
VCD info: dumpfile subtratorcompleto.vcd opened for output.
Iniciando a simulac6o do Subtrator Completo...
A B Cin | S Cout
-----
0 0 0 | 0 0
0 0 1 | 1 1
0 1 0 | 1 1
0 1 1 | 0 1
1 0 0 | 1 0
1 0 1 | 0 0
1 1 0 | 0 0
1 1 1 | 1 1
Simulac6o concluida.
subtratorcompleto.tb.v:63: $finish called at 80000 (1ps)
PS C:\Users\manoe\OneDrive\TI\EmbarcaTech\Segunda Fase\02_Segunda Fase_FPGA\C6digos\Atividades\Unid_02_Cap_02> gtkwave subtratorcompleto.vcd

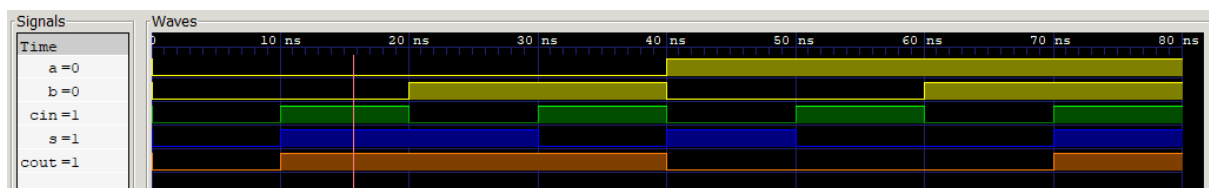
GTKWave Analyzer v3.3.100 (w)1999-2019 BSI

[0] start time.
[80000] end time.
```

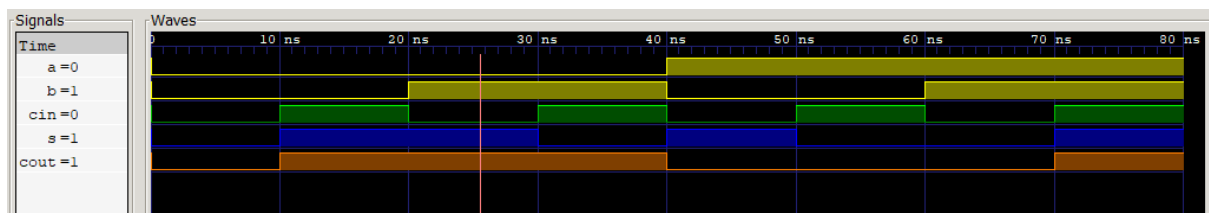
A = 0; B = 0; Cin = 0; S = 0; Cout = 0



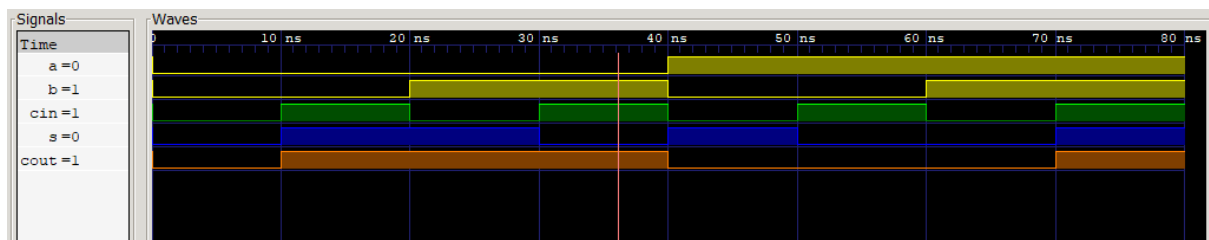
A = 0; B = 0; Cin = 1; S = 1; Cout = 1



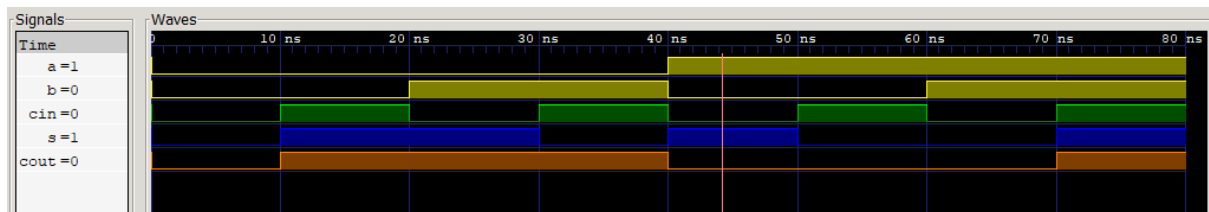
A = 0; B = 1; Cin = 0; S = 1; Cout = 1



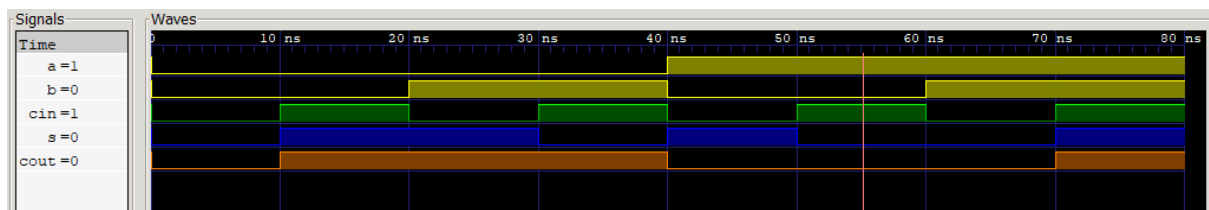
A = 0; B = 1; Cin = 1; S = 0; Cout = 1



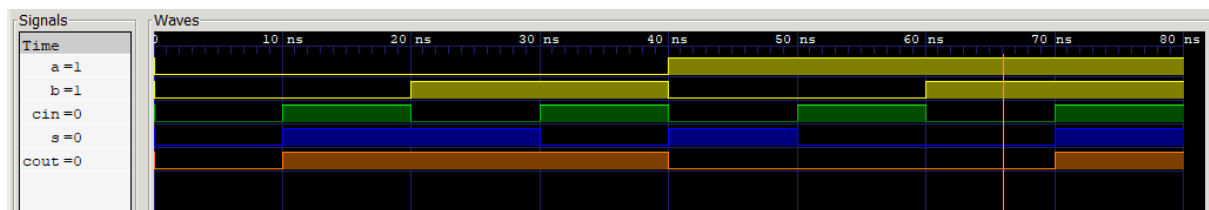
A = 1; B = 0; Cin = 0; S = 1; Cout = 0



A = 1; B = 0; Cin = 1; S = 0; Cout = 0



A = 1; B = 1; Cin = 0; S = 0; Cout = 0



A = 1; B = 1; Cin = 1; S = 1; Cout = 1

