Lucas Loscheider R. M.

Joao Guilherme Monteiro Guimarães

Acumulador multiplicador

Instrução com a finalidade de realizar a multiplicação de dois números quaisquer, obter o resultado e somar o resultado com o resultado da próxima multiplicação.

A instrução realizará a multiplicação toda vez que for identificado um sinal de subida no clock. Haverá também um sinal de de clear o qual terá a função de zerar o resultado das somas consecutivas caso o sinal seja 1, começando a somar do zero assim como no início.

Inicialmente teremos duas entradas de 8 bits para o multiplicador, o qual fará a multiplicação e entregará o resultado de tamanho 16 bits para o somador. No somador teremos outra entrada (além do resultado da multiplicação) que será a do resultado anterior já acumulado da soma (inicialmente zero), e assim realizará a soma e o resultado será entregue ao registrador. Em seguida será exibido esse resultado depois de entregue no registrador e sua saída será uma das entradas do somador, como dito anteriormente, fazendo assim um loop continuo. O clock e clear são recebidos no registrador, o qual identificará o momento certo de de multiplicar, somar e zerar o número registrado. O acumulador é o conjunto do somador, registrador e multiplicador.

Segue abaixo os códigos:

*//**somador***

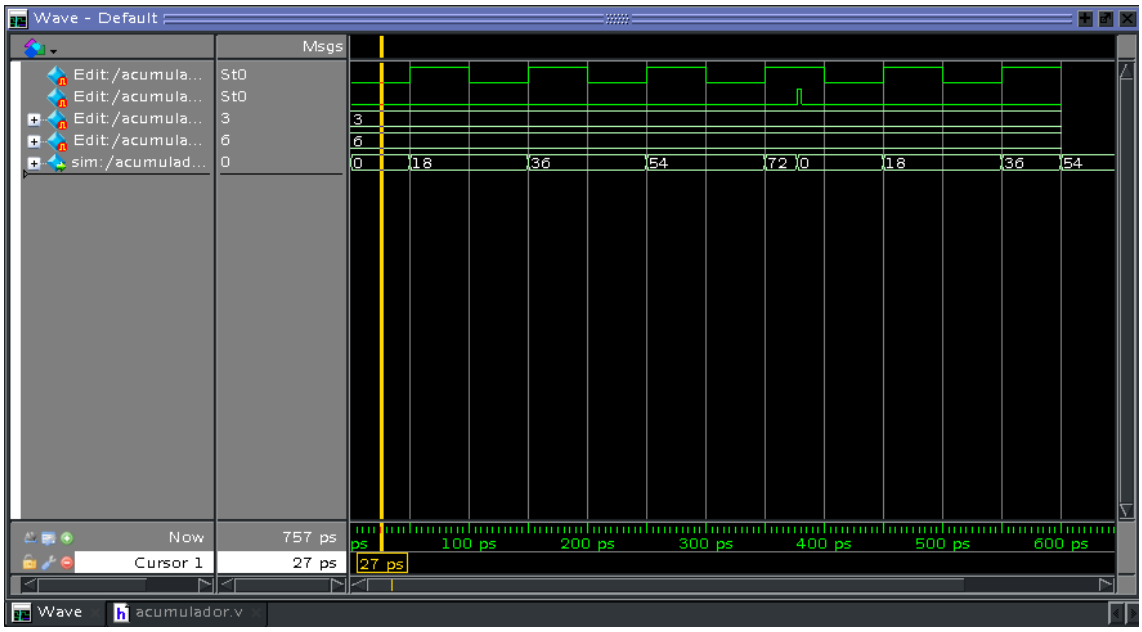
```
// modulo responsavel por realizar a soma ou a subtracao
// das entradas p_A e p_B, dependendo do valor de p_Controle
module somador (p_A, p_B, p_Controle, p_Output);
   input p_Controle;
   input [15:0] p_A, p_B;
   output reg [15:0] p_Output;

   // se p_Controle for igual a Zero, sera realizado
   // uma soma das entradas p_A e p_B, caso contrario,
   // sera realizado uma subtracao entre eles.
   always@(p_A, p_B, p_Controle)
```

```
    if(p_Controle)
        p_Output = p_A + p_B;
    else
        p_Output = p_A - p_B;

endmodule

//registrador
// modulo responsavel para simular o funcionamento de um
// flip-flop do tipo D.
module registrador (p_Input, p_Clock, p_WriteOn, p_Clear, p_Output);
  input p_Clock, p_WriteOn, p_Clear;
  input [15:0] p_Input;
  output reg [15:0] p_Output;

  // valor inicial do registrador
  initial
    p_Output = 16'b0;

  // caso p_Clear esteja ativo, a saida sera zerada indepente
  // do estado de p_WriteOn
  always@(posedge p_Clock, posedge p_Clear)
    if(p_Clear)
      p_Output = 16'b0;
    else if(p_WriteOn)
      p_Output = p_Input;

endmodule
```

```
//multiplicador
// modulo responsavel por realizar a multiplicacao das
// entradas p_A e p_B
module multiplicador (p_A, p_B, p_Output);
   input [7:0] p_A, p_B;
   output reg [15:0] p_Output;

   // a multilicacao so sera realizada novamente, se o
   // estado das entradas mudar
   always@(p_A, p_B)
      p_Output = p_A * p_B;

endmodule

//Acumulador
// modulo responsavel por realizar acumulacao da multiplicao
// das entradas p_A e p_B a cada borda de subida de p_Clock
module acumulador (p_A, p_B, p_Clock, p_Clear, p_Output);
   input p_Clock, p_Clear;
   input [7:0] p_A, p_B;
   output [15:0] p_Output;

   // fios de conexao entre os modulos
   wire [15:0] v_SumToReg, v_MulToSum;

   multiplicador mult (p_A, p_B, v_MulToSum);
```

```
// a entrada p_WriteOn foi setada como constante
registrador regs (v_SumToReg, p_Clock, 1'b1, p_Clear, p_Output);

// este modulo pode trabalhar com duas operacoes, soma ou
// subtracao, nesta pratica, vai ser utilizado somente a operacao
// soma, sendo assim, p_Controle foi setado como 1
somador sum (v_MulToSum, p_Output, 1'b1, v_SumToReg);

endmodule
```

Imagem do processamento:



Podemos observar que dada uma entrada 6 e 3 no multiplicador, no clock de subida é feita a multiplicação e somada com o resultado já armazenado no registrador.

Quando é dado um pulso no reset (segunda linha de cima para baixo) o valor do registrador é zerado.