3 Ввод проекта с использованием кода Verilog

В качестве примера разработки, мы будем использовать схему сумматора — вычитателя, показанную на рисунке 11. Схема может суммировать, вычитать и накапливать n-битные числа, используя двоичный код. Два главных входа имеют разрядность $A = a_{n-1}, a_{n-2}, ... a_0$ и $B = b_{n-1}, b_{n-2}, ... b_0$, а главный вывод имеет значение $Z = z_{n-1}, z_{n-2}, ... z_0$. Другой вход — это контрольный сигнал AddSub, выполняющий действие Z = A + B, если AddSub = 0, и Z = A - B, если AddSub = 1. Второй контрольный вход — Sel, используется для выбора режима работы аккумулятора. Если Sel = 0, то выполняются операции $Z = A \pm B$, а если Sel = 1, к текущему значению Z прибавляется/отнимается значение B. Если в результате сложения или вычитания появляется переполнение, устанавливается сигнал Переполнения.

Чтобы упростить обработку асинхронных входных сигналов, мы будем загружать их в триггеры — защёлки по нарастающему фронту тактовых импульсов. Т.о., входы A и В загружаются в регистры Areg и Breg, a Sel и AddSub загружаются в регистры SelR и AddSubR соответственно. Схема сумматора — вычитателя помещает результат в регистр Zreg.

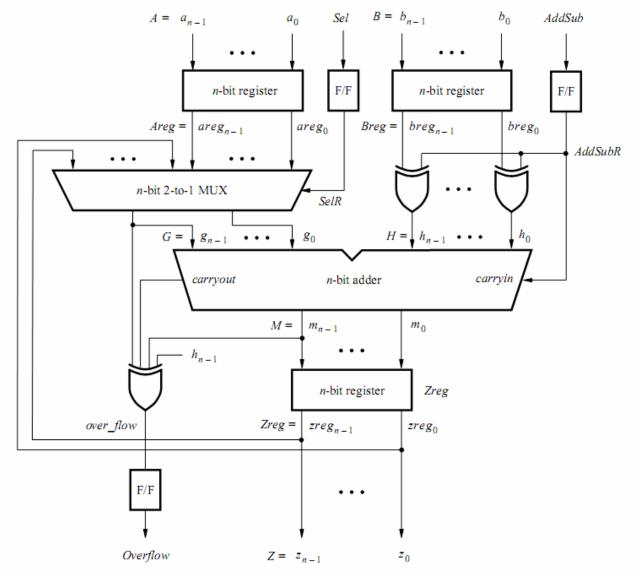


Figure 11: The adder/subtractor circuit.

Схема описывается на языке Verilog на рисунке 12. В ней используется 16-ти битная схема, следовательно, n = 16.

```
// Top-level module
module addersubtractor (A, B, Clock, Reset, Sel, AddSub, Z, Overflow);
 parameter n = 16;
 input [n-1:0] A, B;
 input Clock, Reset, Sel, AddSub;
 output [n-1:0] Z;
output Overflow;
 reg SelR, AddSubR, Overflow;
 reg [n-1:0] Areg, Breg, Zreg;
 wire [n-1:0] G, H, M, Z;
wire carryout, over_flow;
 // Define combinational logic circuit
 assign H = Breg ^ {n{AddSubR}};
 mux2to1 multiplexer (Areg, Z, SelR, G);
   defparam multiplexer.k = n;
 adderk nbit_adder (AddSubR, G, H, M, carryout);
 defparam nbit_adder.k = n;
assign over_flow = carryout ^ G[n-1] ^ H[n-1] ^ M[n-1];
 assign Z = Zreg;
  // Define flip-flops and registers
 always @ (posedge Reset or posedge Clock)
   if (Reset -- 1)
   begin
     Areg <= 0; Breg <= 0; Zreg <= 0;
     SelR <= 0; AddSubR <= 0; Overflow <= 0;
   else
   begin
     Areg <= A; Breg <= B; Zreg <= M;
     SelR <= Sel; AddSubR <= AddSub; Overflow <= over_flow;
endmodule
// k-bit 2-to-1 multiplexer
module mux2to1 (V, W, Sel, F);
 parameter k = 8;
 input [k-1:0] V, W;
 input Sel;
 output [k-1:0] F;
 reg [k-1:0] F;
 always @ (V or W or Sel)
   if (Sel -- 0) F - V;
    else F - W;
endmodule
... continued in Part b
```

Figure 12: Verilog code for the circuit in Figure 11 (Part a)

```
// k-bit adder
module adderk (carryin, X, Y, S, carryout);
parameter k = 8;
input [k-1:0] X, Y;
input carryin;
output [k-1:0] S;
output carryout;
reg [k-1:0] S;
reg carryout;

always @ (X or Y or carryin)
    {carryout, S} = X + Y + carryin;
endmodule
```

Figure 12: Verilog code for the circuit in Figure 11 (Part b).

Обратите внимание на то, что головной модуль назван *addersubtractor*, точно так же, как и на рисунке 4, где описано, как создавать проект. Этот код должен быть определенного типа, и не важно, используете ли вы при его вводе текстовый редактор ASCII или средства ввода текста Quartus II. Общим условием для всех разработчиков является использование того же имени для файла, что и имя головного модуля. Имя файла должно иметь расширение v., которое указывает на файл Verilog. Т.о., мы будем использовать файл *addersubtractor.v.*

Для удобства, мы разместили необходимый файл в директории altera/81/qdesigns/vhdl_verilog_tutorial. Скопируйте этот файл в папку проекта quartus tutorial.