Структура экзамена

1. Задача по написанию устройства, его моделирование и проверка работы на макете;
2. Три теоретических вопроса (из лекций и лабораторных работ);
3. Ограничение по времени: 5 часов;
4. Пользоваться нельзя только соседом;
5. Слайды лекций и методички будут на сайте kprf.mipt.ru

Разделы задач

1. Счетчики (Лабораторная 401);
2. Генераторы сигнала (Лабораторная 402);
3. Измерители сигнала (Лабораторная 403);
4. Вычислители (Лабораторная 409).

Разбор задач

Разбор примеров типичных задач разработки цифровой электроники Часы с будильником

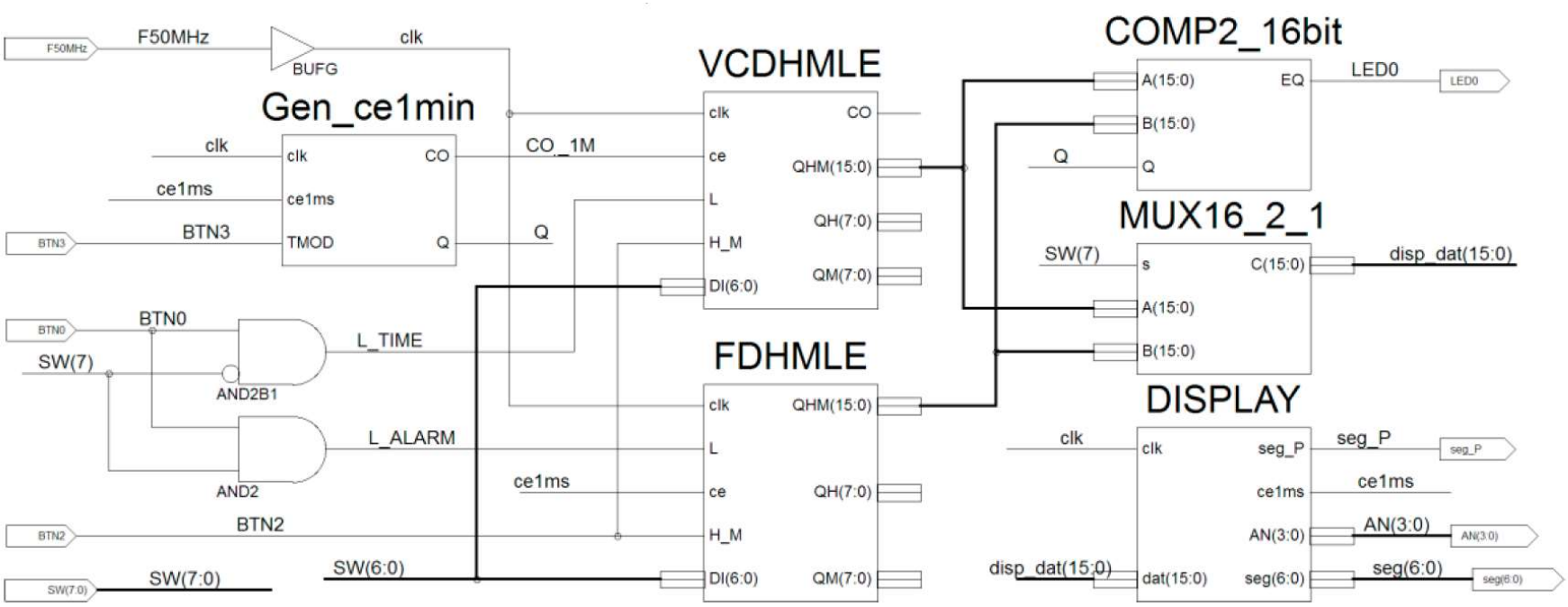
Техническое задание:

Разработать часы с будильником.

- Диапазон часов 0-23;
- Диапазон минут 0-59;
- Отображение текущего времени и времени будильника на семисегментном индикаторе;
- Переключение часы/будильник с помощью переключателя №7;
- Ввод значения переключателями 0-6;
- Установка значения кнопкой 0;
- Выбор установки часов/минут кнопкой 2;
- Отключение сработавшего будильника кнопкой 3;
- Отображение сработавшего сигнала будильника светодиодом 0.

- Разработать секундомер с дискретностью 1/100 и максимальным значением 99.99
- Разработать таймер с дискретностью 1 сек и максимальным значением 99.59
- Разработать календарь

Разбор примеров типичных задач разработки цифровой электроники
Часы с будильником



Разбор примеров типичных задач разработки цифровой электроники
Часы с будильником

Модуль счетчика минут

```
module VCDMLE(
input clk, output wire [7:0] QM, //Минуты
input ce, output wire CO, //"Конец" часа
input [6:0] DI, output reg [3:0]cd_1M=0, //Счетчик единиц минут
input L, output reg [3:0]cb_10M=0); //Счетчик десятков минут
wire CO10M = (cd_1M==9) & ce ; //"Конец" десятка минут
assign CO = CO10M & (cb_10M==5); //"Конец" часа
assign QM = {cb_10M,cd_1M}; //Число минут
always @ (posedge clk) begin
cd_1M <= L? DI[3:0] : CO10M? 0 : ce? cd_1M+1 : cd_1M ;// Счет минут
cb_10M <= L? DI[6:4] : CO? 0 : CO10M? cb_10M+1 : cb_10M ;// Счет часов
end
endmodule
```

Модуль счетчика часов

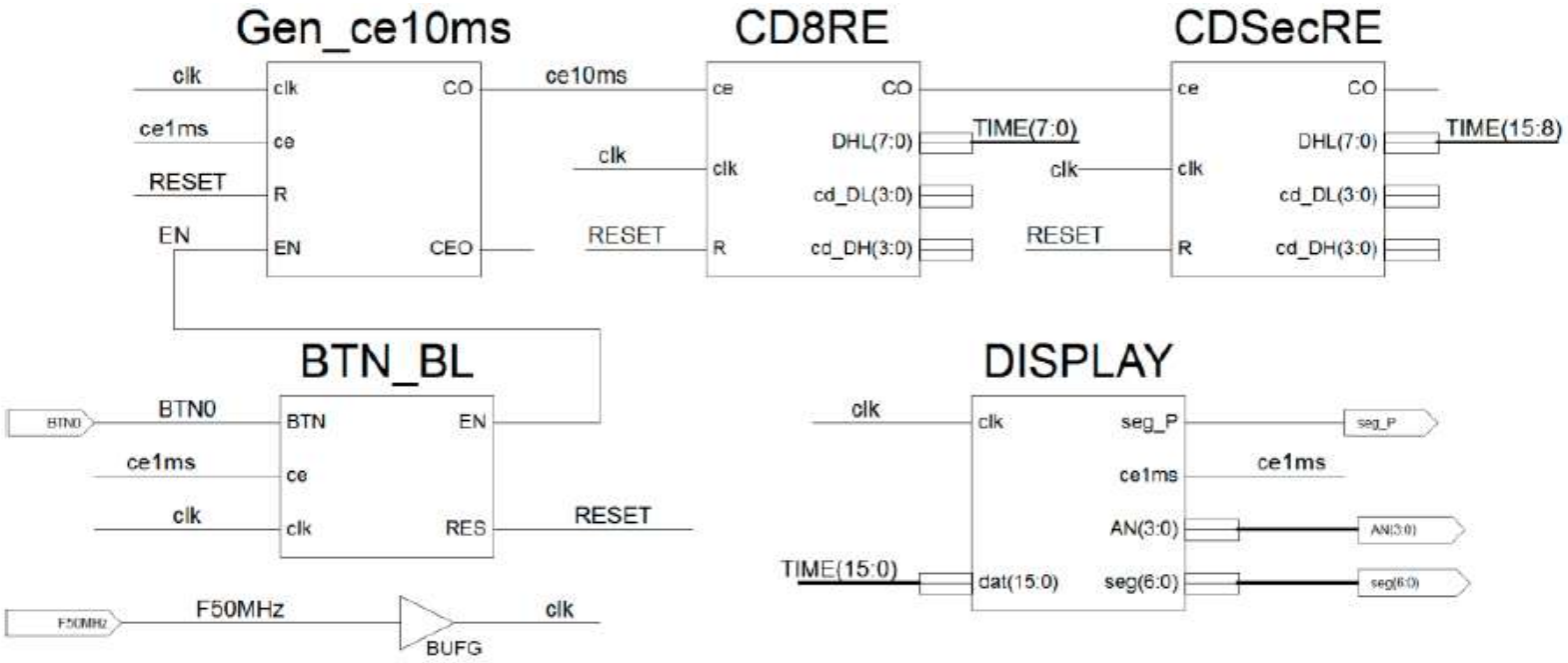
```
module VCDHLE(
input clk, output [7:0] QH,//Часы
input ce, output CO,//"Конец" суток
input [6:0] DI, output reg [3:0] cd_1H=0, //Счетчик единиц часов
input L, output reg [3:0] cb_10H=0);//Счетчик десятков часов

wire CO10H = (cd_1H==9) & ce ;//"Конец" суток
assign CO= ce & (cb_10H==2) & (cd_1H==3);
assign QH = {cb_10H,cd_1H};
always @ (posedge clk) begin
cd_1H <= L? DI[3:0] : (CO10H | CO)? 0 : ce? cd_1H+1 : cd_1H ;
cb_10H <= L? DI[6:4] : CO? 0 : CO10H? cb_10H+1 : cb_10H ;
end
endmodule
```

Разбор примеров типичных задач разработки цифровой электроники Часы с будильником

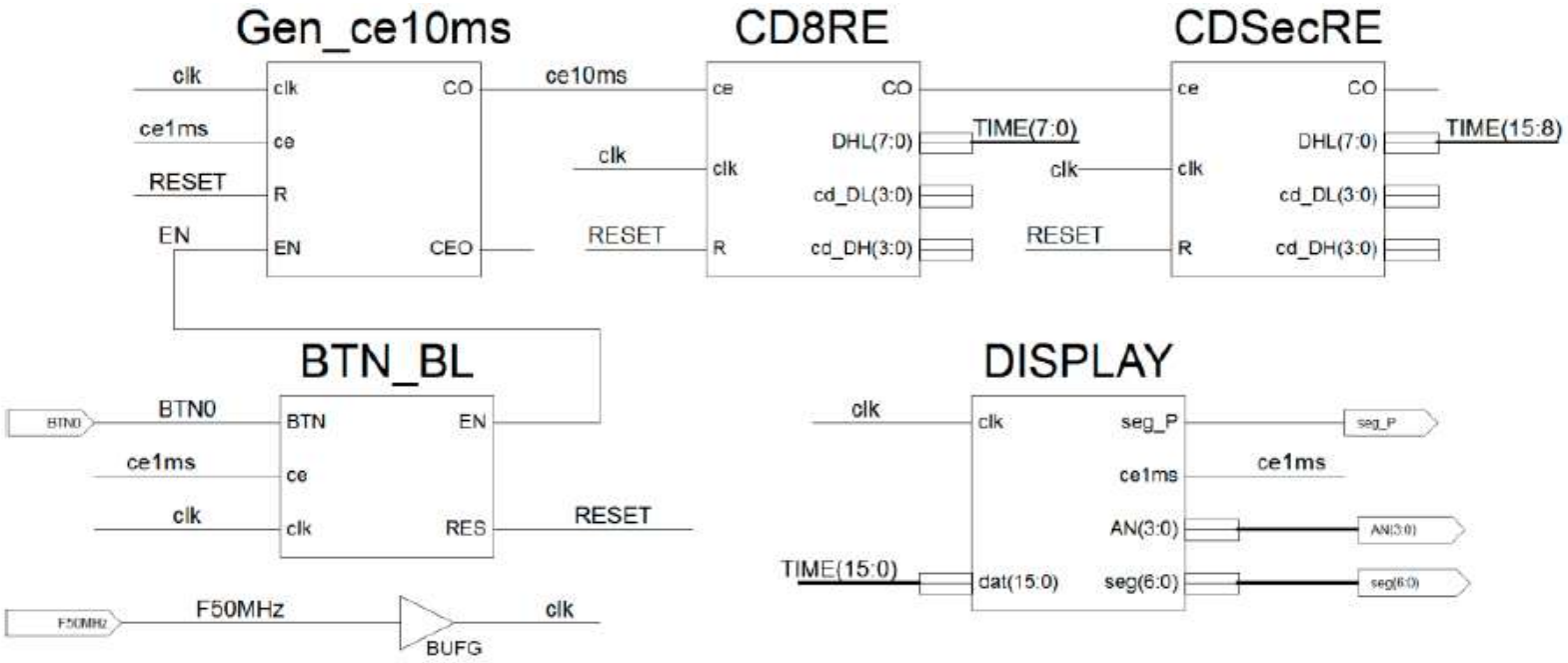
Модуль счетчика минут и часов

```
module VCDHMLE(
input clk, output wire [15:0] QHM, //Часы минуты
input ce, output wire [7:0] QH, //Часы
input [6:0] DI, output wire [7:0] QM, //Минуты
input L, output wire CO, //”Конец” суток
input H_M);//Управление загрузкой
wire COmin, COh ;
wire Lm = L & H_M ; //Загрузка минут
wire Lh = L & !H_M ; //Загрузка часов
assign QHM = {QH,QM} ; //Часы минуты
assign CO = COmin & COh; //”Конец” суток

//Счетчик минут
VCDMLE DD1 (.clk(clk), .QM(QM),
.ce(ce), .CO(COmin),
.DI(DI),
.L(Lm));
//Счетчик часов
VCDHLE DD2 ( .clk(clk), .QH(QH),
.ce(COmin), .CO(COh),
.DI(DI),
.L(Lh));
endmodule
```

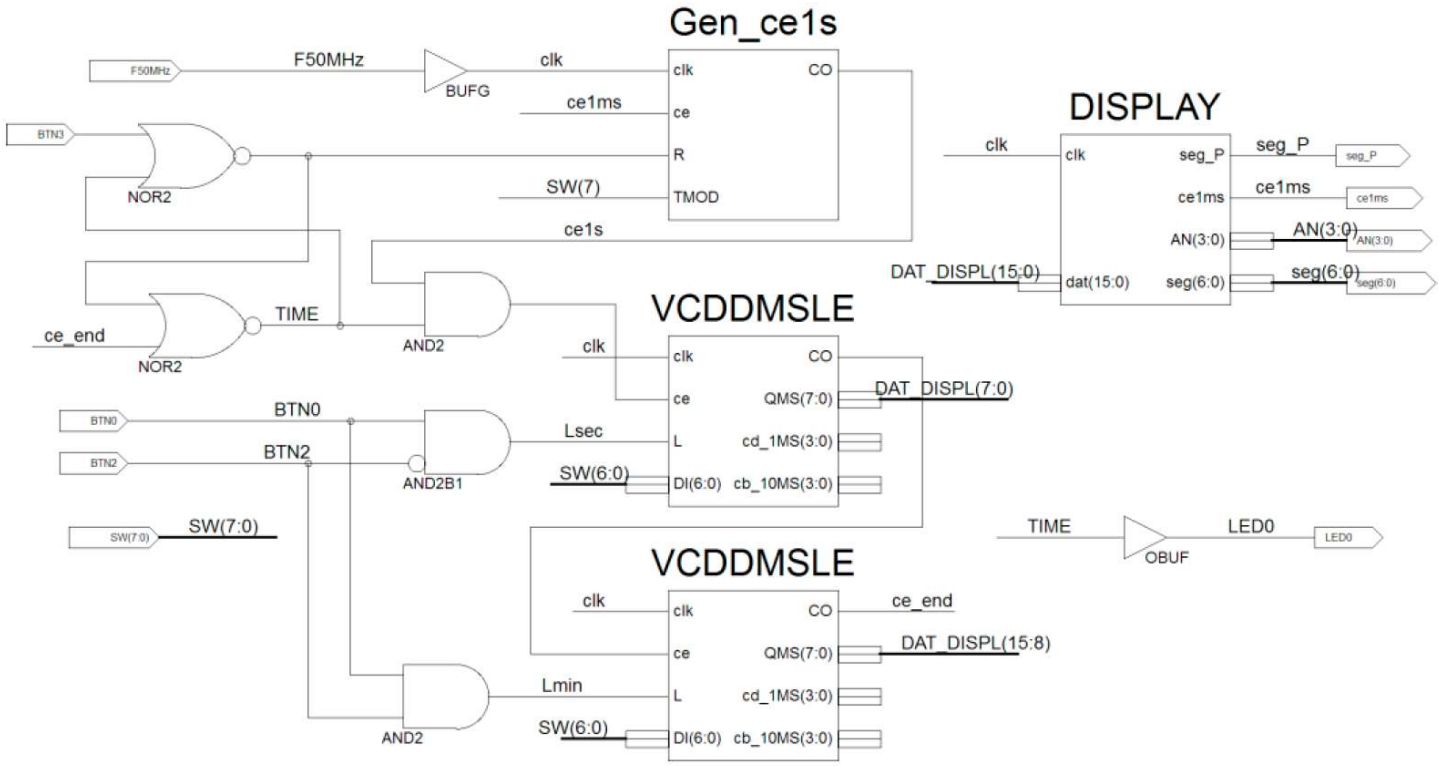
Разбор примеров типичных задач разработки цифровой электроники секундомер



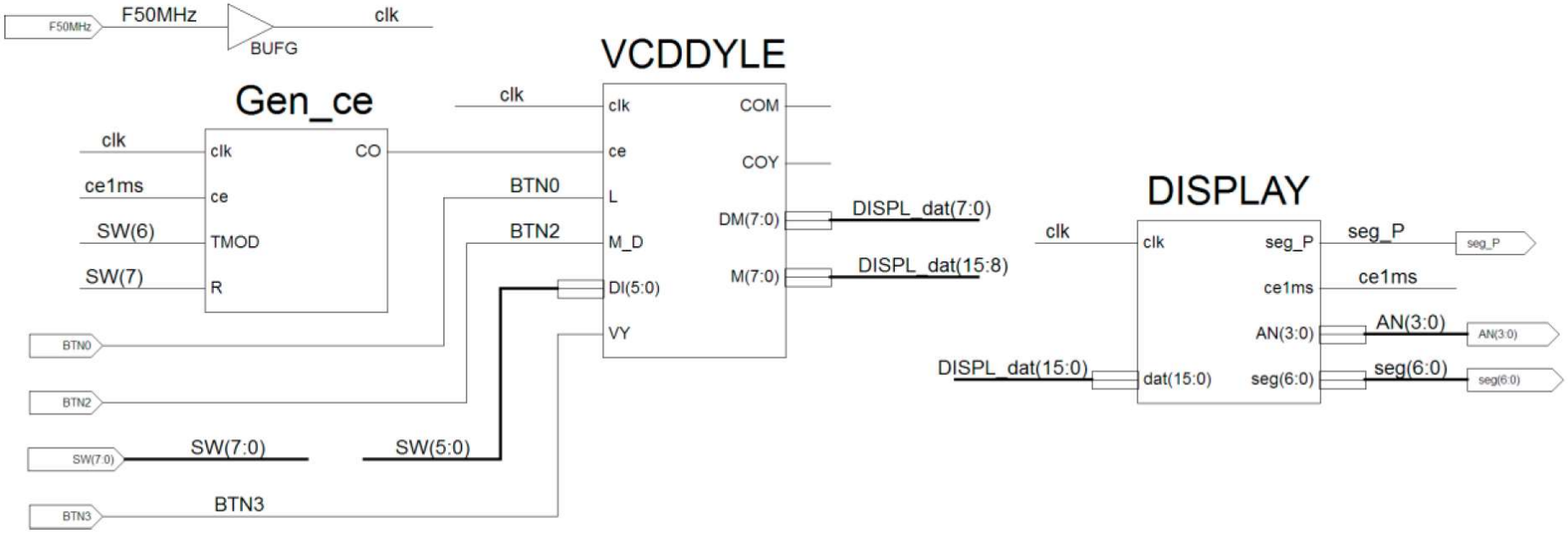
Разбор примеров типичных задач разработки цифровой электроники секундомер



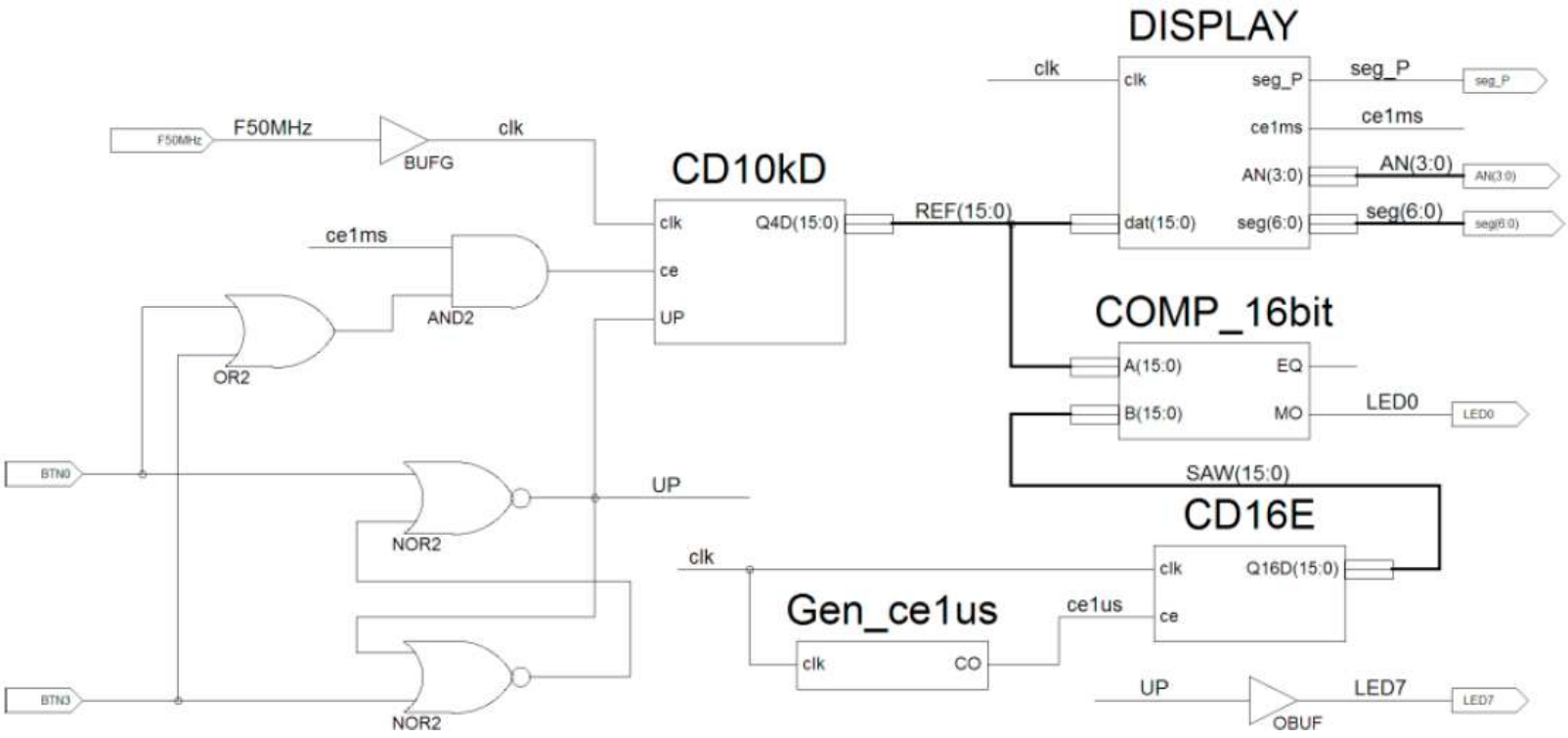
Разбор примеров типичных задач разработки цифровой электроники таймер



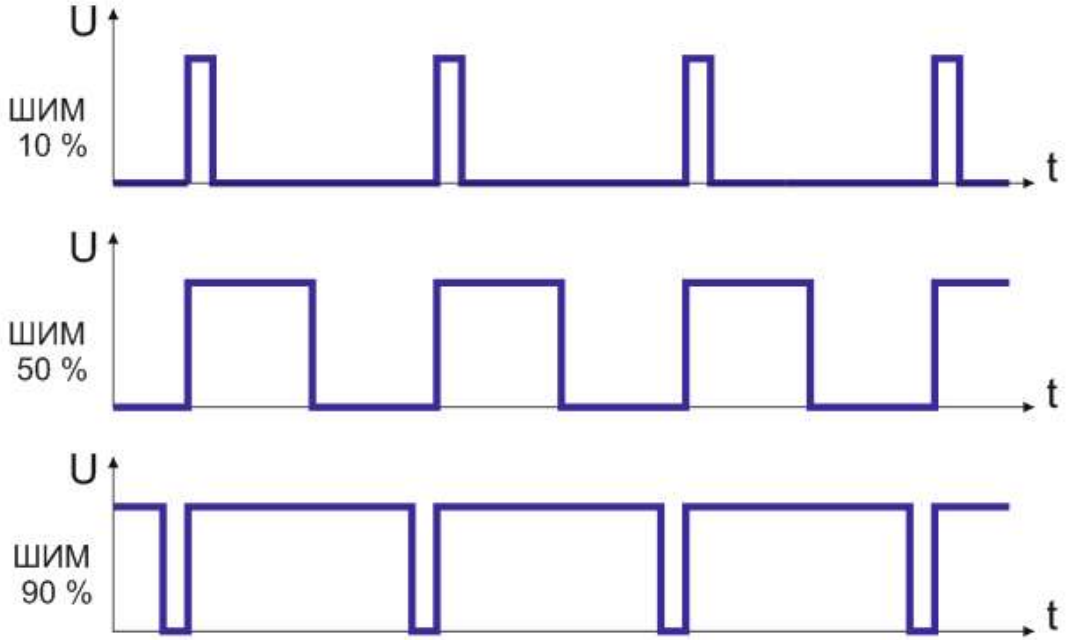
Разбор примеров типичных задач разработки цифровой электроники календарь



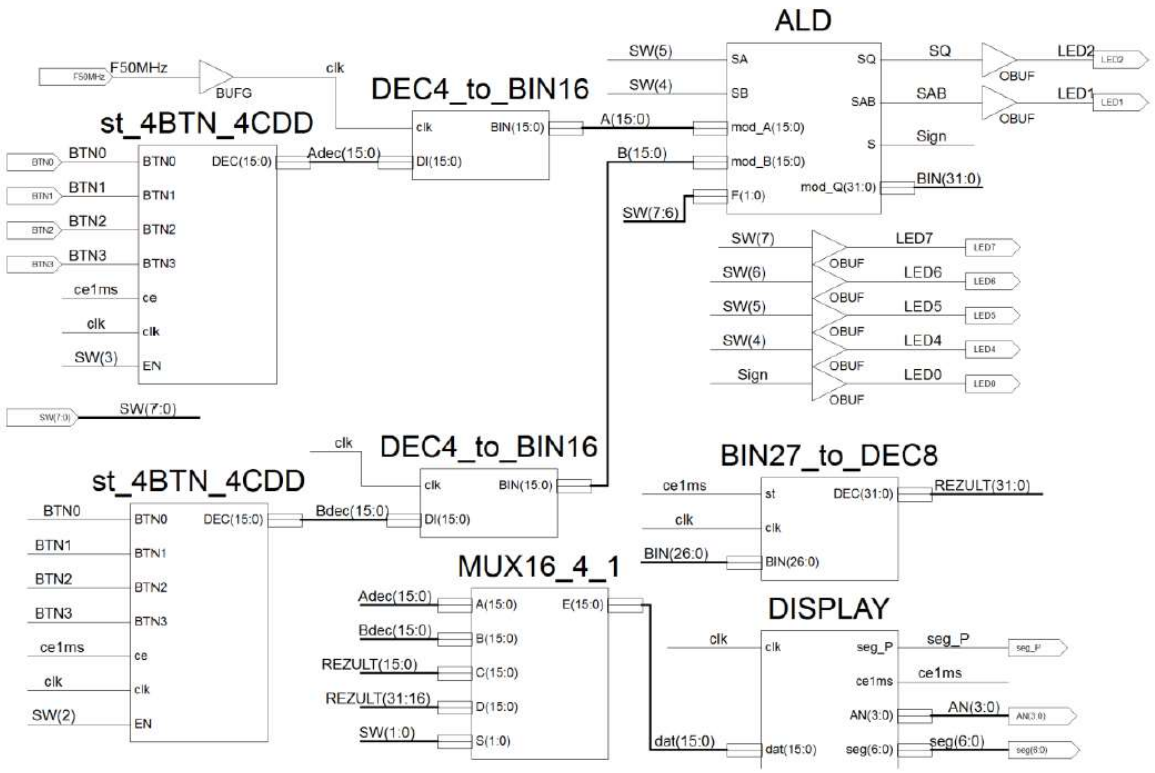
Плавная регулировка яркости диода (ШИМ)



Плавная регулировка яркости диода (ШИМ)



калькулятор



SW[7:6] – код операции
SW[5:4] – знаки операндов
SW[3:2] – выбор управляемого операнда
SW[1:0] – управление мультиплексора
BTN[3:0] – задание операндов

UART калькулятор

Команды:
00 – запись
80 – чтение
Второй байт обеих команд – количество байт данных в команде,
3-4 байты – адрес начала операции в памяти
последующие байты - данные

40 – команда сложения
41 – команда вычитания
42 – команда умножения
43 – команда деления
и т.п.
Это команды с 2 операндами.
Первые 2 байта после команды – адрес в памяти, в который будет записан результат исполнения
Далее 2 блока по 2 байта – адреса операндов в памяти

Любая команда возвращает на UART код успеха 01 или код ошибки 02,
затем данные результата выполнения команды

Целочисленное умножение двоичных чисел с накоплением

```
`define m 24 //24 разряда необходимо для перемножения мантисс
`define n 24  //24 разряда необходимо для перемножения мантисс
module FIX_MULT(input [`m-1:0] A,       output reg[`m+`n-1:0] M=0, //Регистр множимого
                input [`n-1:0] B,       output wire end_mult,  //Конец умножения
                input clk,              output reg en_mult=0,  //Интервал умножения
                input st); //Импульс запуска умножителя
//M=(A*B)
reg [6:0]cb_tact=0 ;            //Счетчик тактов
reg [`m+`n-1:0] bf_A=0 ;        //Регистр сдвига сомножителя A
reg [`n-1:0] bf_B=0 ;           // Регистр сдвига сомножителя B
assign end_mult = (cb_tact==1) ; //Конец умножения

always @ (posedge clk) begin
en_mult <= st? 1 : end_mult? 0 : en_mult ; //Интервал умножения
cb_tact <= st? `n : en_mult? cb_tact-1 : cb_tact ; //Счет тактов
bf_A <= st? A : en_mult? bf_A<<1 : bf_A ; //Сдвиг A влево (умножение A на 2)
bf_B <= st? B : en_mult? bf_B>>1 : bf_B ; //Сдвиг B вправо для выделения младшего бита B
M <= st? 0 : (en_mult & (bf_B[0]==1))? M+bf_A : M ; //Накопление произведения
end
endmodule
```

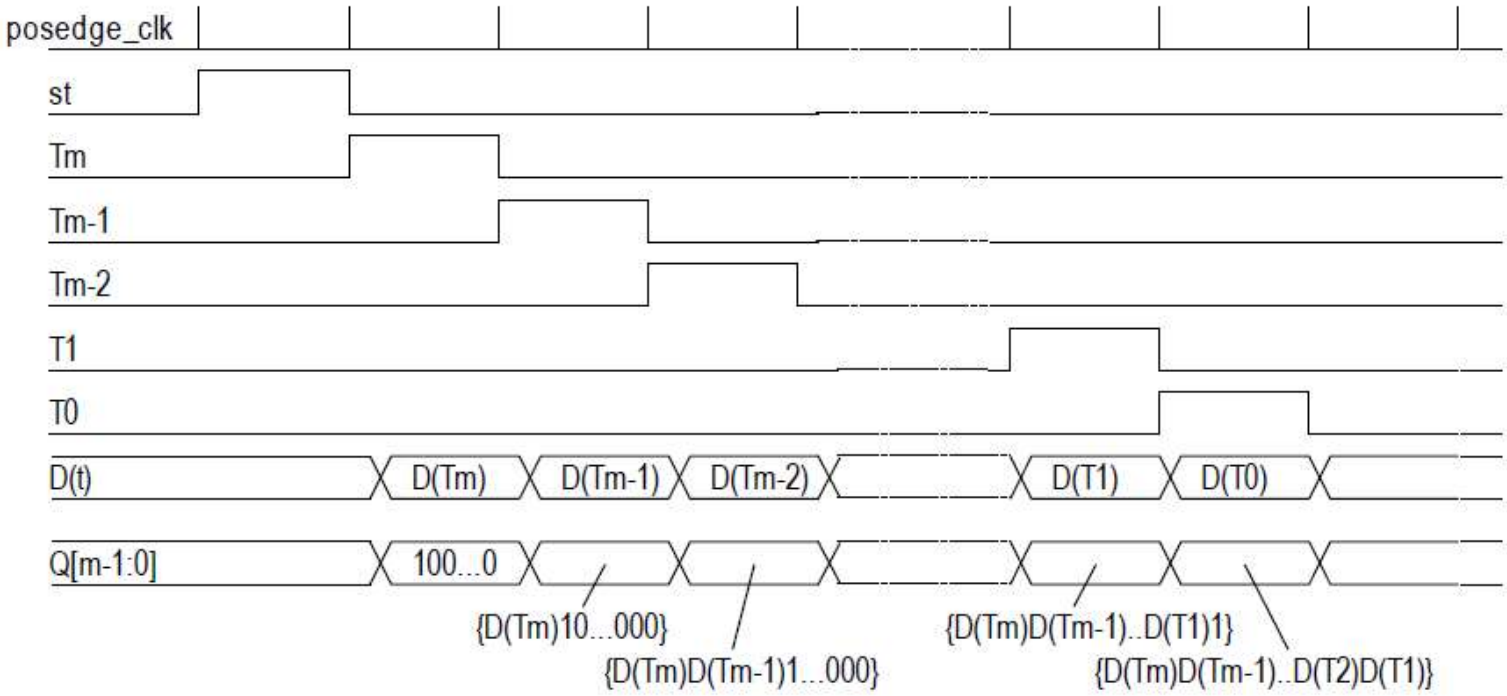
Умножение со знаком

```
`define m 18
module SMULT(
                input [`m-1:0] A,       output reg [2*`m-1:0] M=0, //Произведение
                input [`m-1:0] B,
                input clk,
                input st);

wire s_A = A[`m-1];             //Знак числа A
wire s_B = B[`m-1];             //Знак числа B
wire s_M = s_A ^ s_B ;          //Знак произведения M
wire [`m-2:0]mod_A= s_A? -A[`m-2:0] : A[`m-2:0]; //Модуль числа A
wire [`m-2:0]mod_B= s_B? -B[`m-2:0] : B[`m-2:0]; //Модуль числа B
wire [2*`m-3:0]mod_M =mod_A * mod_B; // Произведение модулей

always @ (posedge clk) begin
M <= st? 0 : (s_M & ce)? -mod_M : ce? mod_M : M; //Произведение
end
endmodule
```

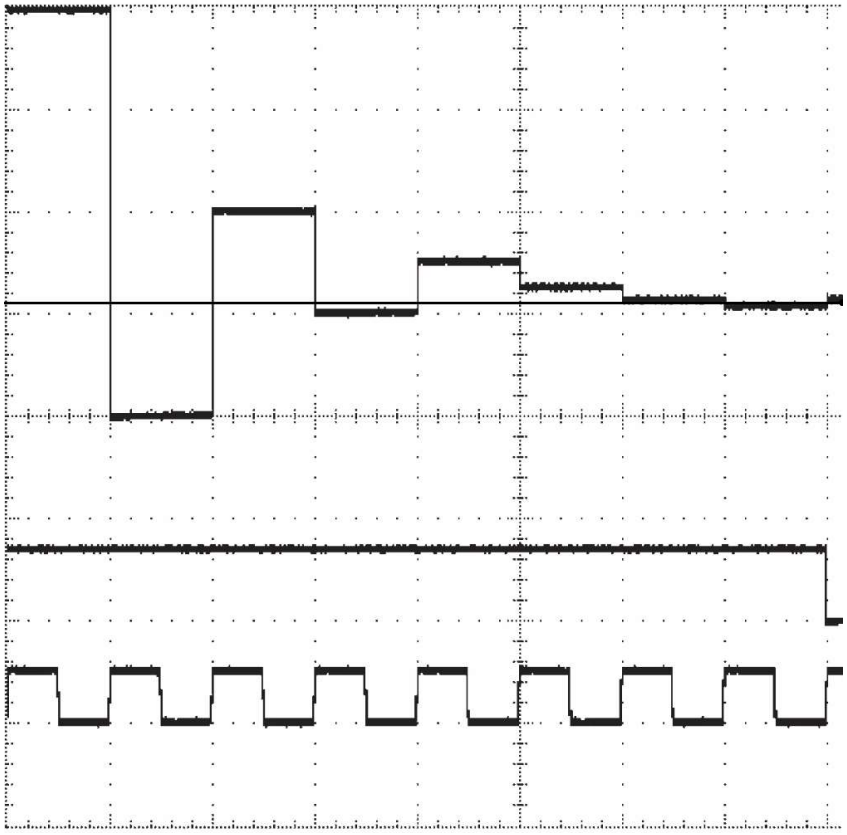
Последовательное приближение

```
`define m 16
module SAR_SR (    input st,        output reg en=0, //Интервал приближения
                   input D,         output reg [`m-1:0] Q=0, // Выходной регистр
                   input clk,       output wire ok); //Конец приближения

reg[`m:0] T=0 ;          //Регистр сдвига
assign ok = T[0] ;       //Конец последовательного приближения
integer i ;              //Индекс цикла for
always @ (posedge clk)  begin
T <= st? 1<<`m : en? T>>1 : T ; //Сдвиг импульсов T вправо
for (i=`m-1 ; i >= 0; i=i-1) // Цикл for
Q[i] <= T[i+1]? 1 : T[i]? D : Q[i] ; //Загрузка очередного бита выходного регистра Q
en <= st? 1 : (T[0] & en)? 0 : en ;     //Интервал  последовательного приближения
end
endmodule
```

Последовательное приближение



Последовательное приближение



АЦП методом последовательного приближения

```
module ADC( input [15:0] X,          output wire[15:0]Q, // Выход результата
            input st,                output wire en, //Интервал приближения
            input clk,               output wire ok, //Конец приближения
                                     output wire DI); // Входной бит регистра
assign DI =(Q<=X); //Входной бит регистра последовательного приближения
SAR_SR  DD1 (      .st(st),          .Q(Q), // Выход регистра
                   .D(DI),           .en(en), //Интервал приближения
                   .clk(clk),        .ok(ok)); //Конец приближения
endmodule
```

Вычисление квадратного корня методом последовательного приближения

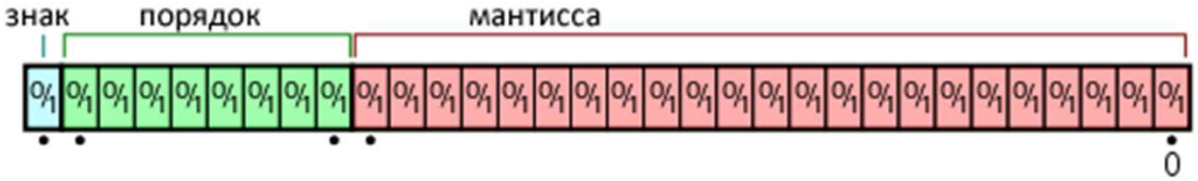
```
module SQRT_BL (  input [31:0] X,        output wire[15:0]Q,
                  input st,              output wire en,
                  input clk,             output wire ok,
                                         output wire DI);

wire [31:0]M=Q*Q ; // Возведение Q в квадрат
assign DI =(M<=X); //Входной бит регистра последовательного приближения
SAR_SR  DD1 (     .st(st),        .Q(Q), // Выход регистра
                  .D(DI),         .en(en), //Интервал приближения
                  .clk(clk),      .ok(ok)); //Конец приближения

endmodule
```

Вычисление кубического корня методом последовательного приближения

```
module CUBRT_BL (input [47:0] X,        output wire[15:0]Q, // Выход результата
                 input st,              output wire en, //Интервал приближения
                 input clk,             output wire ok, //Конец приближения
                                        output wire DI); // Входной бит регистра
wire [47:0]M=Q*Q*Q ; // Возведение Q в куб
assign DI =(M<=X); //Входной бит регистра последовательного приближения
SAR_SR  DD1 (      .st(st),         .Q(Q), // Выход регистра
                   .D(DI),          .en(en), //Интервал приближения
                   .clk(clk),       .ok(ok)); //Конец приближения
endmodule
```

Числа с плавающей точкой

$(-1)^s \times M \times 2^E$

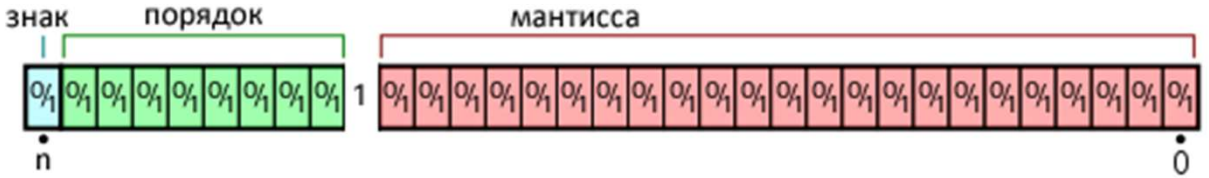


Нормализованный вид:

$(-1)^s \times 1.M \times 2^E$

+Денормализованный вид:

$(-1)^s \times 0.M \times 2^{Emin}$, если $E=E_{min}-1$



IEE754 (0: m=0, $e=e_{min}-1$; бесконечность: m=0, $e=e_{max}+1$; NaN: m – не 0, $e=e_{max}+1$)

s=0 (положительное)

$E=01111100_2-127_{10} = -3$

$M = 1.01_2$ (первая единица не явная)

$F = 1.01_2e-3 = 2^{-3}+2^{-5} =$

0,125 + 0,03125 = 0,15625

Числа с плавающей точкой

Округление в IEEE754:
0.5 округляется до четного: 1.5=2, 2.5=2

Проблема ассоциативности для чисел с плавающей точкой:
$(10^{20}+1)-10^{20}=0 \neq (10^{20}-10^{20})+1=1$

Сравнение специальных чисел IEEE754:

x	y	x < y? x: y	x <= y? x: y	x > y? y: x	x >= y? y: x
+0	-0	-0	+0	+0	-0
NaN	1	1	1	NaN	NaN

Умножение чисел с плавающей точкой

```
module MULT_FLOAT (output wire[47:0]MAxMB, // Произведение мантисс
input [31:0] A, output wire[31:0]M, //Произведение
input [31:0] B, output reg ok=0, //Конец умножения
input st,
input clk );
parameter E0 = 127;//E0=8'b01111111=8'h7F(смещение экспоненты)
wire ZA = (A==0) ; //Сомножитель A==0 ;
wire ZB = (B==0) ; //Сомножитель B==0 ;
wire [7:0]EA=A[30:23]- E0; //Экспонента A
wire [23:0]MA={1'b1,A[22:0]}; //Мантисса A
wire [7:0]EB=B[30:23]- E0; // Экспонента B
wire [23:0]MB={1'b1,B[22:0]}; // Мантисса B
assign MAxMB = MA*MB ; //Перемножение мантисс
wire SA=A[31] ; wire SB=B[31] ; //Знаки сомножителей
wire [7:0]EM= MAxMB[47]? (EA+EB)+ E0+1 : EA+EB+E0; //Экспонента M
wire [22:0]MM = MAxMB[47]? MAxMB>>22 : MAxMB>>23; // Модификация мантиссы M
wire SM = SA^SB ; // Знак произведения
assign M = (ZA | ZB)? 0 : {SM,EM,MM} ; //Сборка произведения
always @ (posedge clk) begin
ok <= st ; // Конец преобразования
end
endmodule
```

Умножение чисел с плавающей точкой

```
module ADD_FLOAT(
input st, output wire [31:0] SUMM, //Сумма чисел A и B
input [31:0] A, output wire [7:0] Exp_A, // Экспонента числа A
input [31:0] B, output wire [7:0] Exp_B, // Экспонента числа B
input clk, output wire [23:0] M_A, //Мантисса числа A
output wire [23:0] M_B, // Мантисса числа B
output wire [7:0] d_Exp, //Разность экспонент
output wire d_S, //Разность знаков слагаемых
output reg en_sh_M=0, //Разрешение сдвига мантиссы меньшего числа
output reg [7:0]cb_sh_M=0, //Счетчик сдвигов меньшей мантиссы
output wire ok_sh_M, // Конец сдвига мантиссы меньшего числа
output reg [23:0]sr_MA=0, //Регистр сдвига мантиссы A
output reg [23:0]sr_MB=0, // Регистр сдвига мантиссы B
output wire [24:0]Temp_SUMM, //Временная сумма/разность
output reg en_sh_TS=0, //Разрешение сдвига разности
output reg [23:0]sr_TS=0, //Регистр сдвига временной суммы/разности
output reg [6:0]cb_sh_TS=0, //Счетчик сдвига суммы/разности
output wire ok_sh_TS, //Конец сдвига разности
output reg ok_SUMM=0 ); //Конец суммирования
assign Exp_A=A[30:23]; assign M_A={1'b1,A[22:0]}; //Экспонента и мантисса числа A
assign Exp_B=B[30:23]; assign M_B={1'b1,B[22:0]}; // Экспонента и мантисса числа B
wire SA=A[31] ; wire SB=B[31] ; //Знаки чисел A и B
wire A_big_B = (A[30:0]>B[30:0]) ; //Модуль A больше модуля B
wire A_equ_B = (A[30:0]==B[30:0]) ; // Модуль A равен модулю B
assign d_S = SA^SB ; //Разность знаков слагаемых
assign d_Exp=A_big_B? Exp_A-Exp_B : Exp_B-Exp_A; // Разность экспонент
wire start= st & !(d_Exp==0); //Нет старта сдвига, если разность экспонент равна нулю
assign Temp_SUMM = !d_S? (sr_MA+sr_MB): A_big_B? sr_MA-sr_MB : sr_MB-sr_MA;
wire CO=Temp_SUMM[24] ; //Переполнение суммы мантисс
wire [7:0]max_Exp = A_big_B? Exp_A : Exp_B ; //Выбор максимальной экспоненты
wire S_SUMM = A_big_B? SA : SB ; //Знак суммы равен знаку большего числа
wire [7:0]Exp_SUMM = CO? max_Exp+1 : d_S? max_Exp-cb_sh_TS : max_Exp ;
wire [22:0]M_SUMM = CO? Temp_SUMM[23:1] : sr_TS[22:0] ;
assign SUMM = (A_equ_B & d_S)? 0 : //S=0 при равных модулях и разных знаках
(A==0)? B : //Если A==0, то SUMM=B
(B==0)? A : //Если B==0, то SUMM=A
{S_SUMM,Exp_SUMM,M_SUMM} ; //Сумма
reg ten_sh_M =0;
reg tst=0 ;
reg ten_sh_TS=0 ;
assign ok_sh_M = (d_Exp==0)? tst : !en_sh_M & ten_sh_M ; /* Конец сдвига мантиссы меньшего числа*/
assign ok_sh_TS= !en_sh_TS & ten_sh_TS ; // Конец сдвига временной суммы
always @ (posedge clk) begin
ten_sh_M <= en_sh_M ; tst <= st ; ten_sh_TS <= en_sh_TS ; //Задержка на такт
en_sh_M <= start? 1 : (cb_sh_M==1)? 0 : en_sh_M ; //Интервал сдвига мантисс
cb_sh_M <= start? d_Exp : en_sh_M? cb_sh_M-1 : cb_sh_M ; //Счет сдвигов мантиссы
sr_MA <= st? M_A : (en_sh_M & !A_big_B)? sr_MA>>1 : sr_MA ; //Сдвиг мантиссы A
sr_MB <= st? M_B : (en_sh_M & A_big_B)? sr_MB>>1 : sr_MB ; // Сдвиг мантиссы B
en_sh_TS <= (ok_sh_M & d_S & !(Temp_SUMM[23]))? 1 : (sr_TS[22])? 0 : en_sh_TS ; 6

cb_sh_TS <= ok_sh_M? 0 : en_sh_TS? cb_sh_TS+1 : cb_sh_TS ; // Счет сдвигов
sr_TS <= ok_sh_M? Temp_SUMM[23:0] : en_sh_TS? sr_TS<<1 : sr_TS ; //Сдвиг разности
ok_SUMM <= (!d_S | (d_S & (Temp_SUMM[23])))? ok_sh_M : ok_sh_TS ;
end
endmodule
```