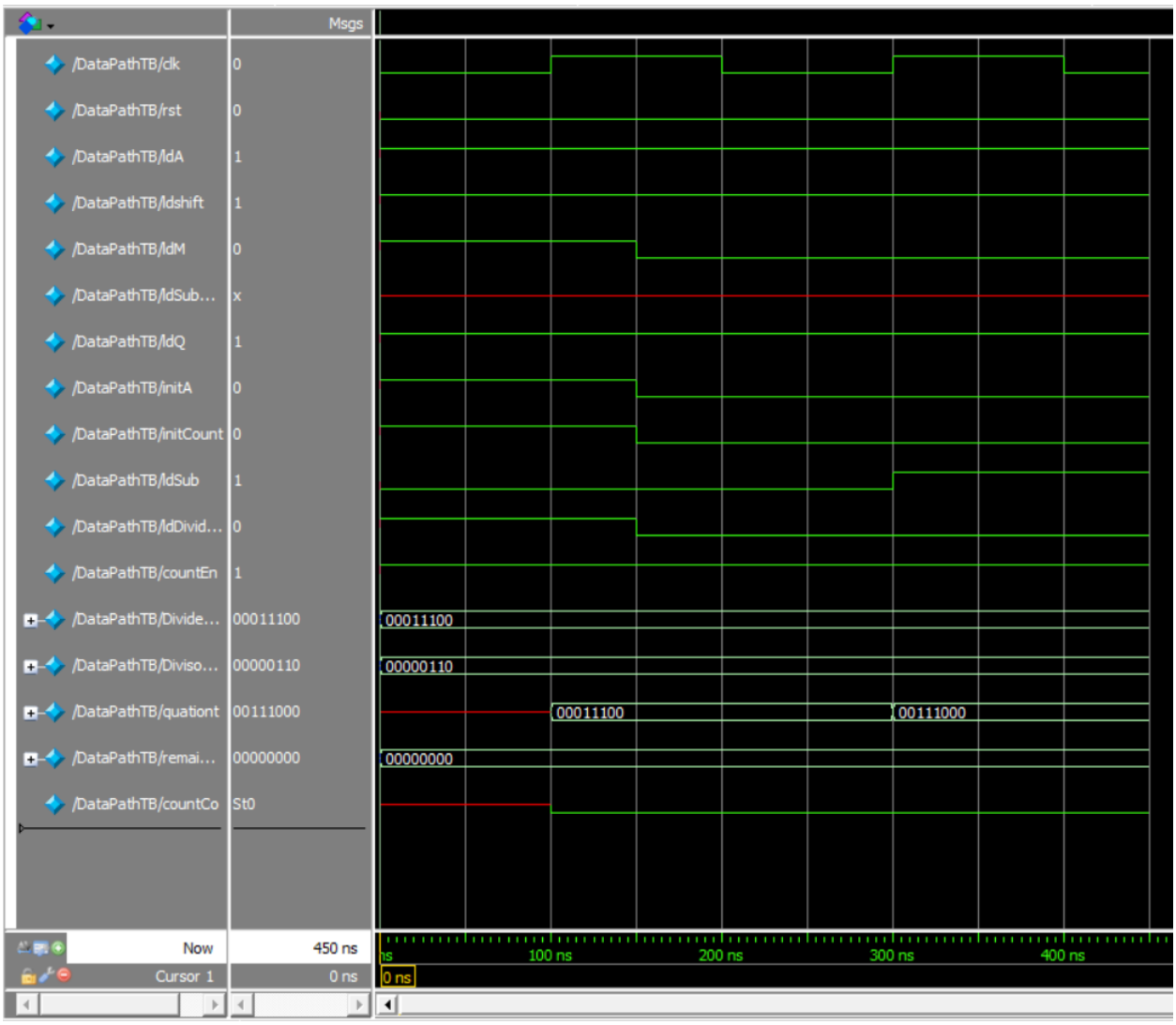
In this project we are going to design a **Divider** circuit.

```
Ln#
 1  `timescale 1ns/1ns
 2  module Q_reg(input clk, rst, cen ,incrmnt, input [7:0] pin, output reg [7:0] pout);
10
11  module M_reg(input clk, rst, cen, input [7:0] pin, output reg [7:0] pout);
18
19  module A_reg(input clk, rst, cen, input [7:0] pin, output reg [7:0] pout);
26
27  module Mux(input [7:0] a, b, input sla, slb, output [7:0] s);
31
32  module Subtractor(input [8:0] a, b ,  output logic [15:0] po);
37
38  module Shifter (input [15:0] a , output [15:0] out);
41
42  module counter(input cen, iz, clk, rst, output logic [2:0] po, output co);
53
54  module counterTB ();
67
68  module DataPath(input clk, rst, ldA, ldshift, ldM, ldQ, ldSub, ldDividend, initA, initCount, countEn,
69  input [7:0] Dividend_bus , Divisor_bus , output [7:0] quationt , remainder , output countCo );
70
71  wire [2:0] count ;
72  wire [7:0] Qreg , Mreg , Areg , QBus ,Abus ,sub ,Qshift ,Ashift ,mx ;
73  wire [15:0] shiftin ,shiftout ;
74  Q_reg Dividend (clk , rst ,ldQ,  ~sub[7], QBus, Qreg);
75  M_reg Divisor  (clk , rst ,ldM, Divisor_bus, Mreg);
76  A_reg Afill (clk , initA ,ldA, Abus , Areg);
77  Subtractor Subt ({1'b0,Areg} , {1'b0,Mreg} , sub);
78  Mux abus (mx , Ashift , ldSub , ldshift , Abus);
79  Mux SUB (sub , Areg , ~sub[7]  , sub[7] , mx);
80  Mux qbus (Dividend_bus , Qshift , ldDividend , ldshift , QBus);
81  counter cnt (countEn , initCount , clk , rst , count , countCo);
82  assign shiftin = {Areg , Qreg};
83  Shifter shft (shiftin , shiftout);
84  assign {Ashift[7:0] , Qshift[7:1] , Qshift[0] } = {shiftout[15:1] , ~sub[7]} ;
85
86  assign quationt = Qreg;
87  assign remainder = Areg;
88  endmodule
89
```

DataPath Codes. Assuming all modules and wiring them together in DataPath module.

```
90  module DataPathTB();
91  reg clk =0 ;
92  reg rst, ldA, ldshift, ldM, ldSubReg, ldQ, initA, initCount , ldSub, ldDividend ,countEn;
93  reg [7:0] Dividend_bus , Divisor_bus;
94  wire[7:0] quationt , remainder ;
95  DataPath UUT1 (clk, rst, ldA, ldshift, ldM, ldSubReg, ldQ, ldSub, ldDividend, initA, initCount, countEn, Dividend_bus , Divisor_bus, quationt , remainder ,countCo);
96  always #100 clk = ~clk;
97  initial begin
98  Dividend_bus = 8'b00011100;
99  Divisor_bus = 8'b00000110;
100 //LOAD
101 ldA =1 ; ldM =1 ; ldQ =1 ; ldshift = 1; initA =1;
102 rst =0 ; initCount =1 ; ldSub =0; ldDividend=1; countEn=1;
103 #150
104 //shift
105 ldA =1 ; ldM =0 ; ldQ =1 ; ldshift = 1; initA =0;
106 rst =0 ; initCount =0 ; ldSub =0; ldDividend=0; countEn=1;
107 #150
108 //Minuse
109 ldA =1 ; ldM =0 ; ldQ =1 ; ldshift = 1; initA =0;
110 rst =0 ; initCount =0 ; ldSub =1; ldDividend=0; countEn=1;
111
112 #150  $stop;
113 //
114 end
115 endmodule
```

TestBench for DataPath. Assuming All control Signals that should come from controller manualy just to test the Functionality of circuit.

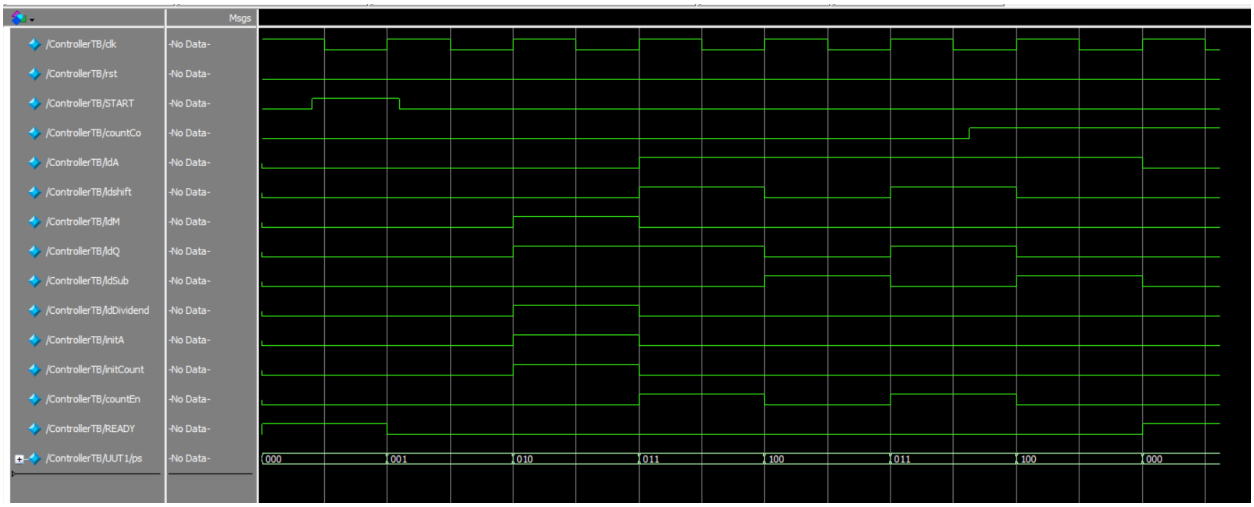


we have not connected the Controller Unit so we don’t expect Dividing from this circuit.

Hopefully DataPath is working Correcty.

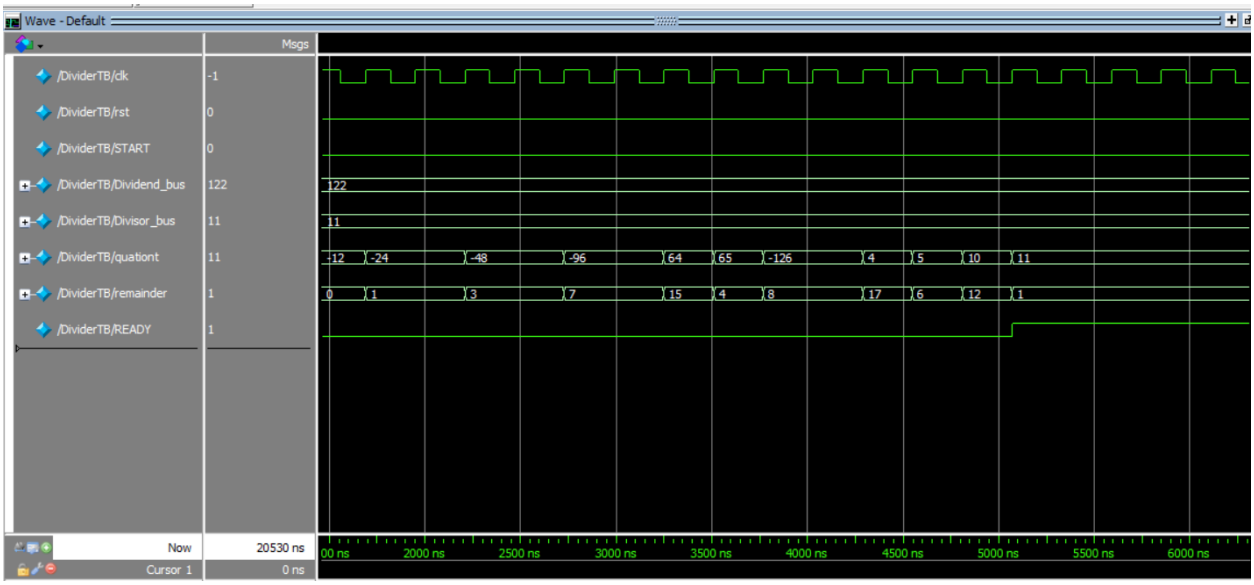
```
116
117 module Controler (input clk , rst, START ,countCo , output logic ldA, ldshift, ldM, ldQ, ldSub, ldDividend, initA, initCount, countEn ,READY);
118 parameter Idle = 3'd0 , Starting = 3'd1 , Load = 3'd2 , Shift = 3'd3 , Minuse =3'd4 ;
119 logic [2:0] ps , ns = 3'd0;
120 always @(ps, START, countCo)begin
121 {ldA, ldshift, ldM, ldQ, ldSub, ldDividend, initA, initCount, countEn ,READY} = 10'b0;
122 case (ps)
123 Idle : begin ns = (START) ? Starting : Idle ;
124         READY =1; end
125 Starting : begin ns = (START) ? Starting : Load;
126 end
127 Load : begin ns = Shift;
128         {initA ,ldQ ,ldM ,initCount ,ldDividend } =5'b11111; end
129 Shift : begin ns = Minuse;
130         {ldA ,ldQ , ldshift ,countEn } = 4'b1111; end
131
132 Minuse : begin ns = (countCo) ? Idle : Shift ;
133         {ldA , ldSub } =2'b11; end
134 default ns = 3'b0;
135 endcase
136 end
137 always @(posedge clk, posedge rst) begin
138                 if (rst) ps <= 3'b 000;
139                 else ps <= ns;
140         end
141 endmodule
142
143 module ControllerTB();
144 logic clk =1 , rst=0 , START =0 , countCo =0 ;
145 wire ldA, ldshift, ldM, ldQ, ldSub, ldDividend, initA, initCount, countEn ,READY;
146 Controler UUT1 (clk ,rst ,START ,countCo ,ldA, ldshift, ldM, ldQ, ldSub, ldDividend, initA, initCount, countEn ,READY);
147 always #100 clk = ~clk;
148         initial begin
149         #80 START = 1;
150         #140 START = 0;
151         #800;
152         #105 countCo = 1; START = 0;
153         #400 $stop;
154         end
155 endmodule
156
```

Assuming All Signals that should come from DataPath as inputs manualy just to test the Functionality of circuit.



Aparantly Controller is Working Correctly. In each state control signals that we want are assuming.

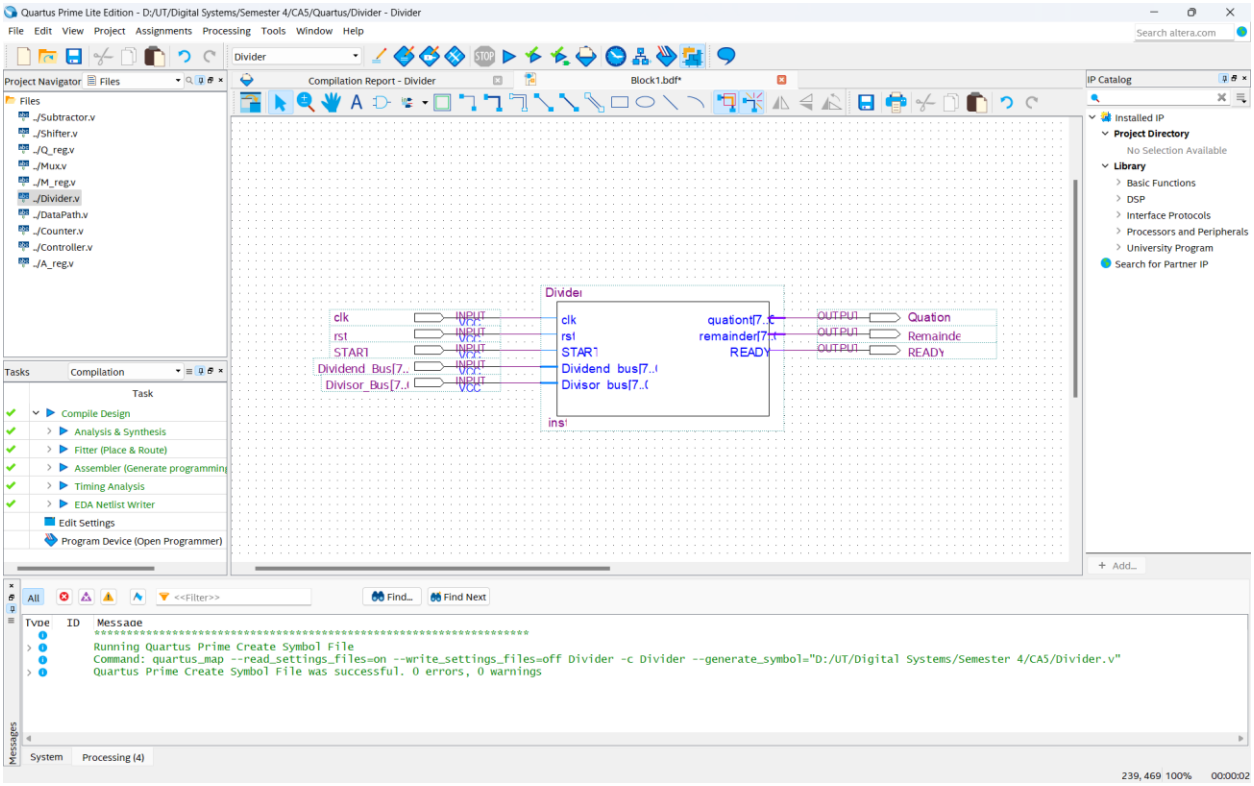
Now connecting controler to DataPath should work correctly. Hopefully!



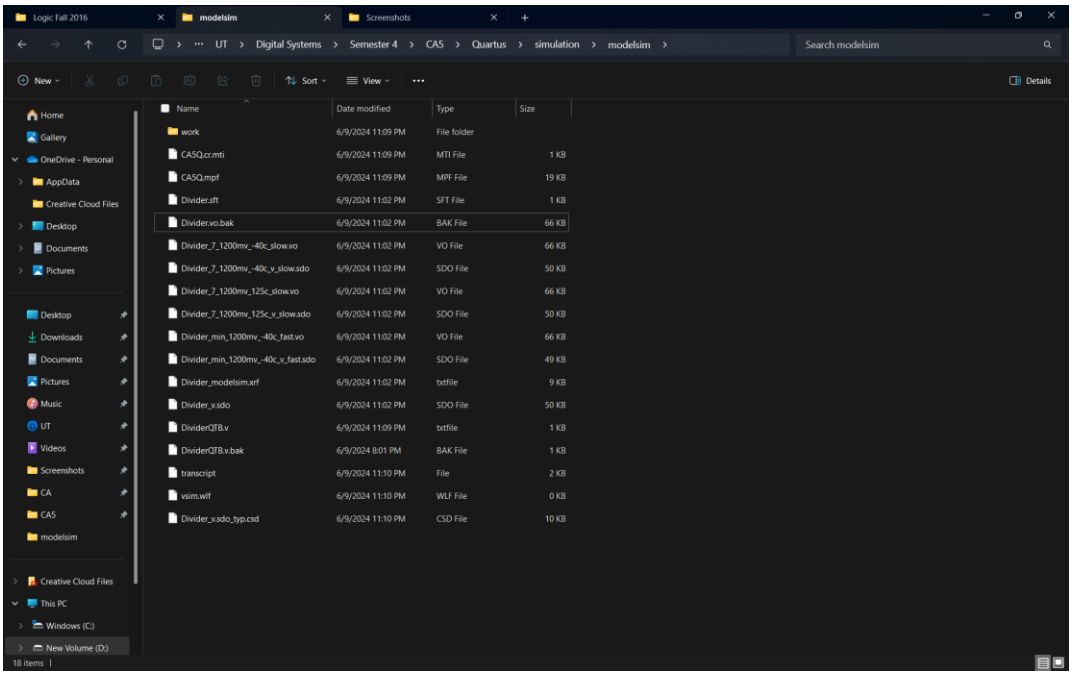
When the READY signal is turned on, we read the outputs and they are right according to inputs.

Beautifully Done!

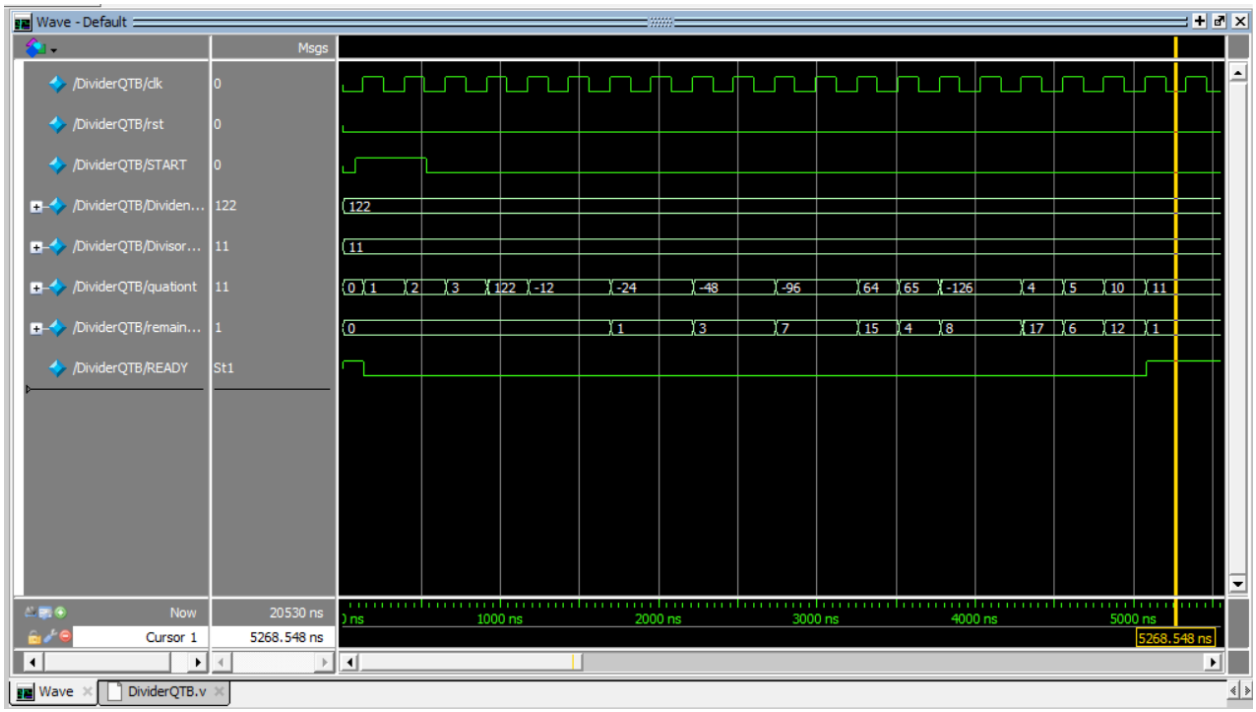
Quartus



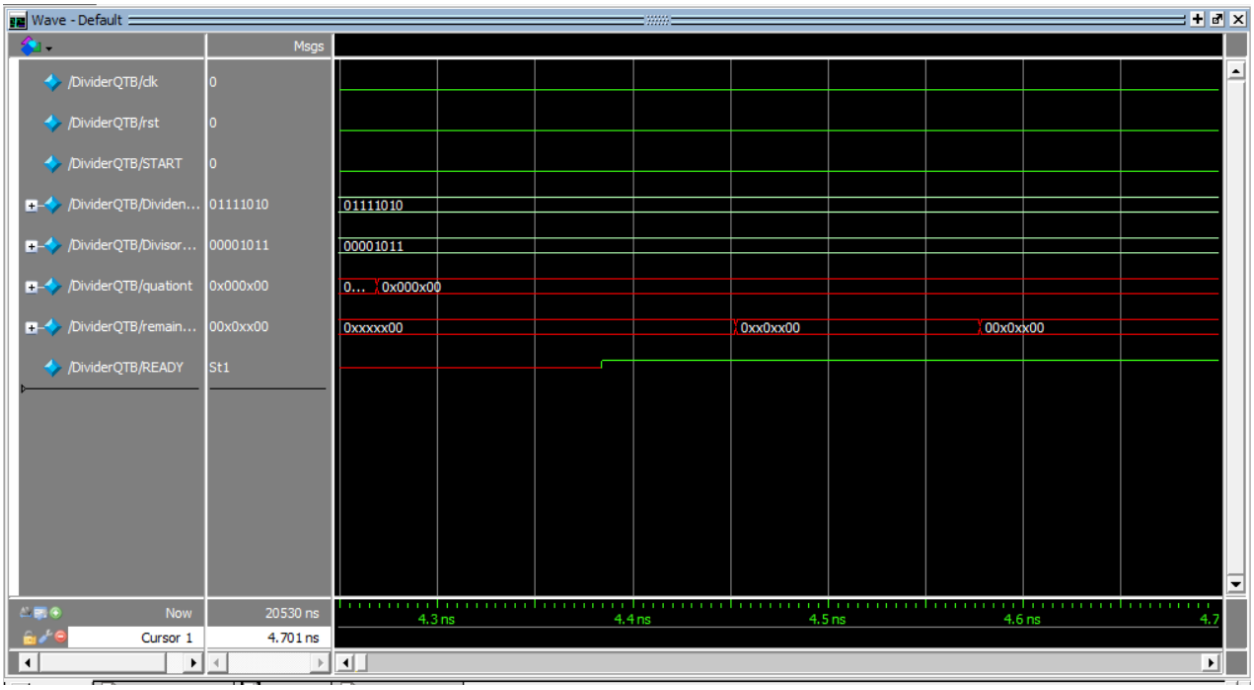
Symbol for Top level block diagram in quartus.



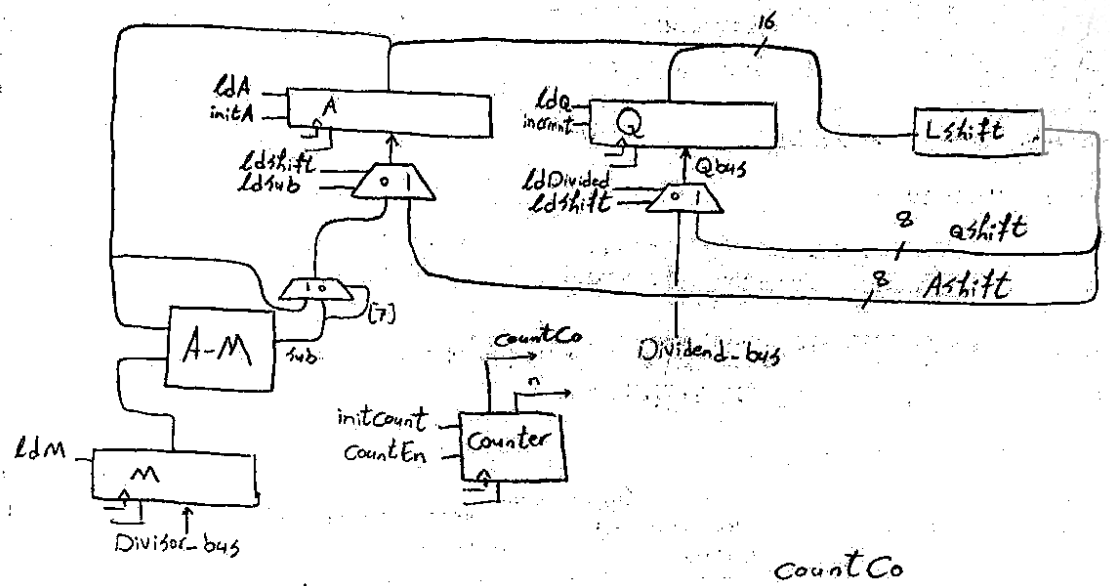
.sdo and .vo files are created.

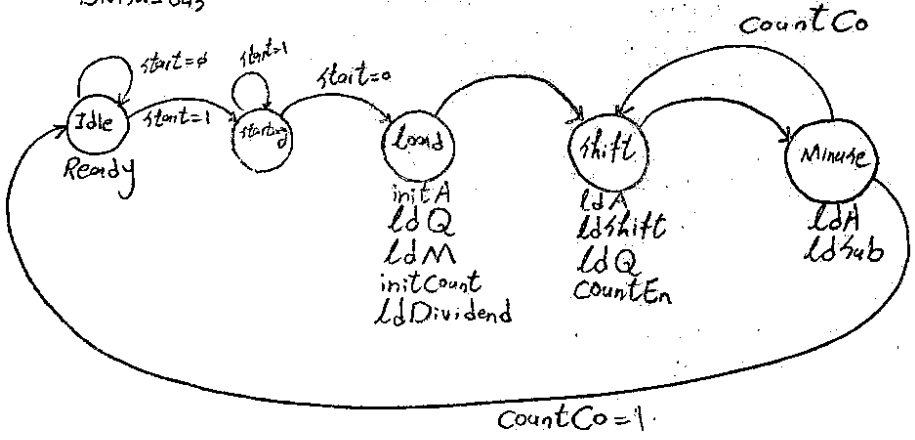


in this simulation we can see timing is included.



Timing is obvious in this Waveform.


16
ldA
initA
A
ldQ
initQ
Q
Lshift
ldAshift
ldSub
Qbus
ldDivided
ldQshift
8
Qshift
8
Ashift
A-M
sub
[7]
countCo
n
Dividend_bus
initCount
countEn
Counter
ldM
M
Divisor_bus


start=0
start=1
start=0
countCo
Idle
Ready
start=1
starting
Load
initA
ldQ
ldM
initCount
ldDividend
shift
ldA
ldAshift
ldQ
countEn
Minus
ldA
ldSub
countCo=1