EE 502

Computer Architecture

Lab Assignment #2

Zexi Liu

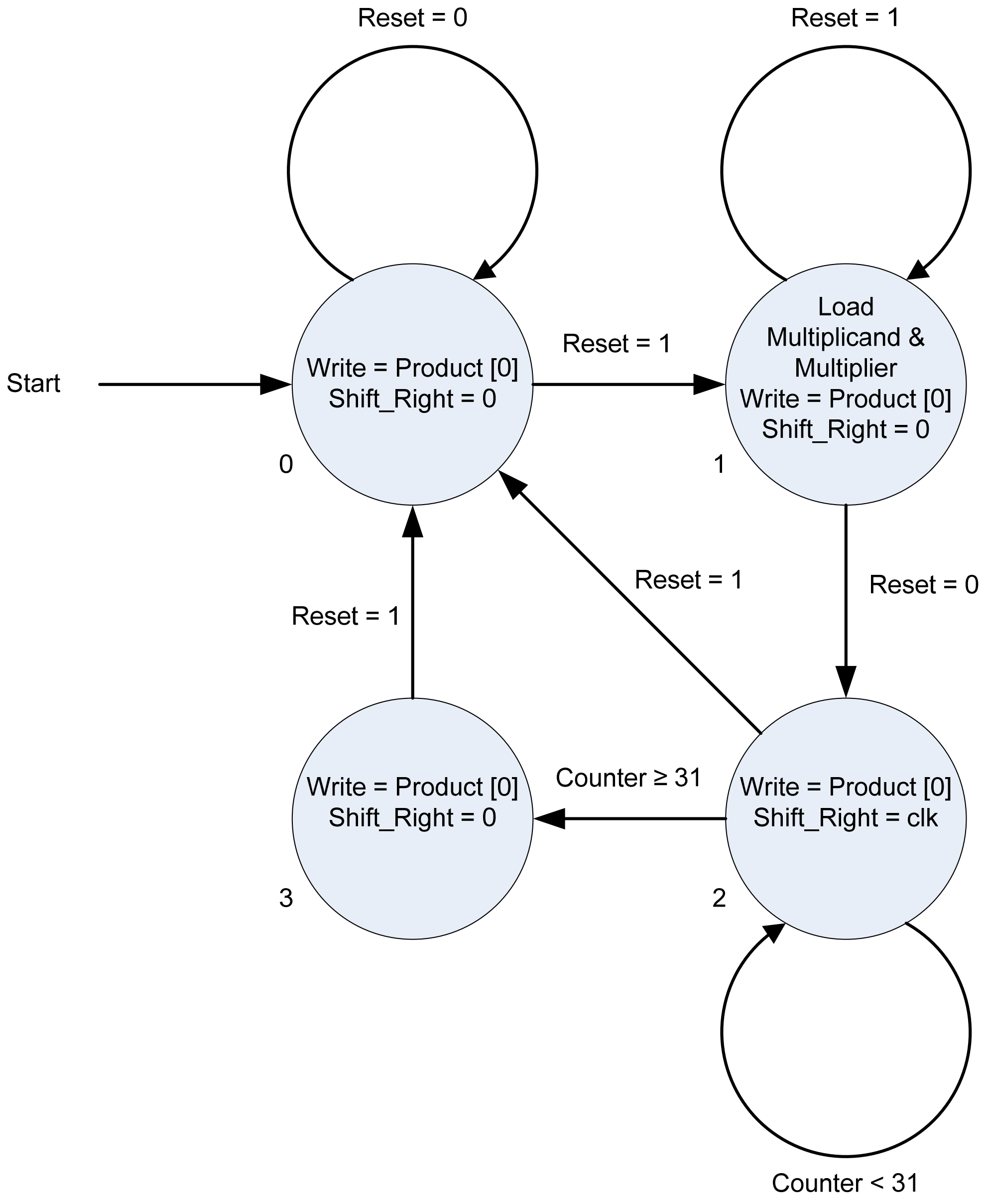
Mar 10th, 2007

Electrical and Computer Engineering

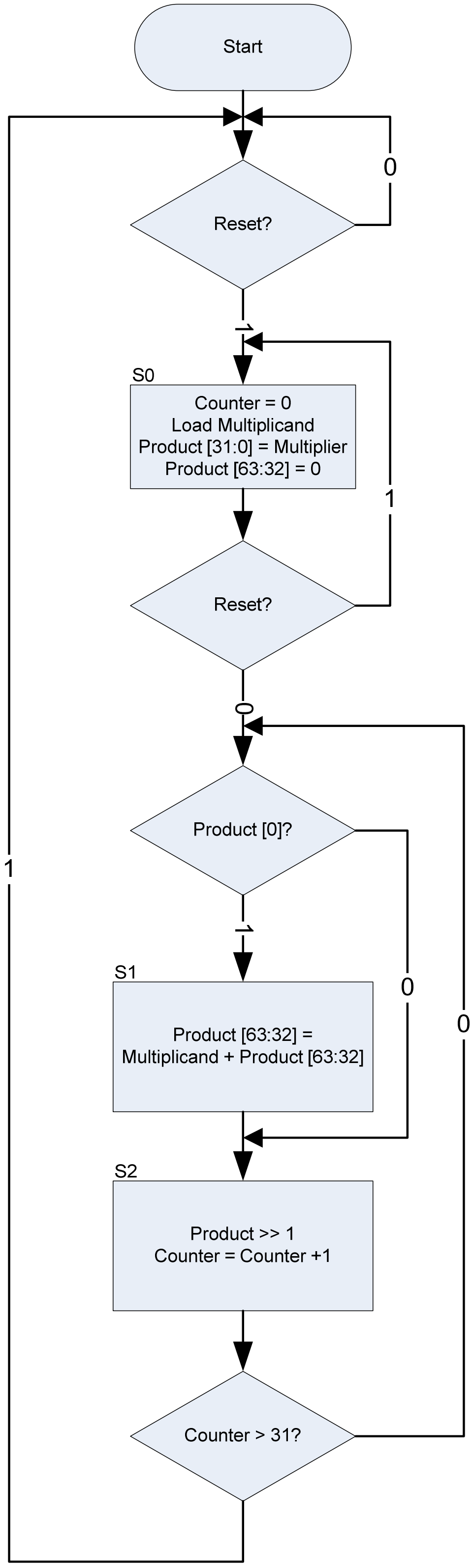
Temple University

1. **(10 points) Develop a multiplication algorithm for the refined multiplication hardware in Figure 3.7. It is similar to the multiplication algorithm shown in Figure 3.6. Please also derive a STG (state transition graph), ASM or ASMD (algorithmic state machine and datapath) for your algorithm. In your STG/ASM/ASMD, please clearly describe the states and data transfer movements.**

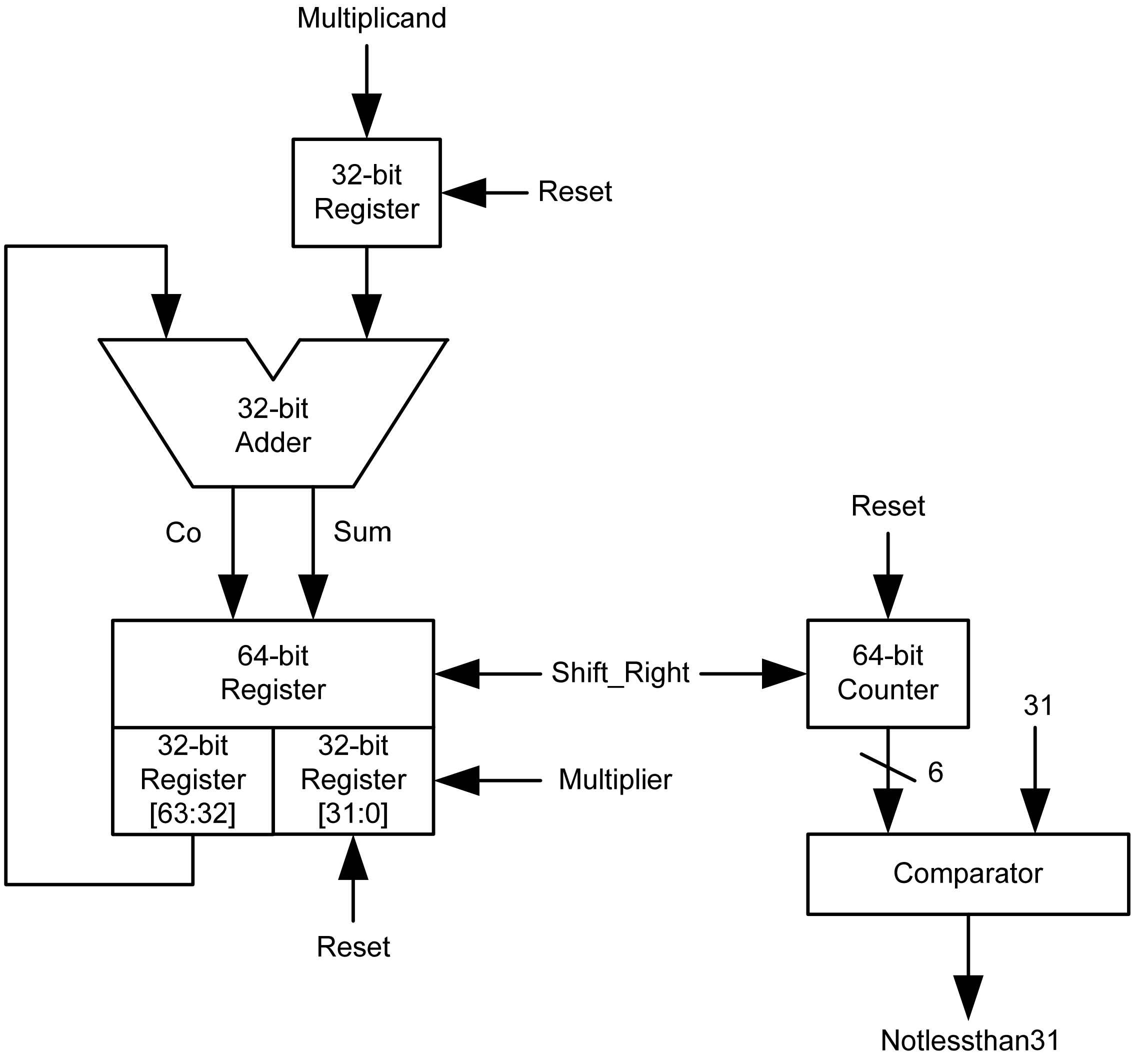
   STG (State Transition Graph)



ASM (Algorithmic State Machine)

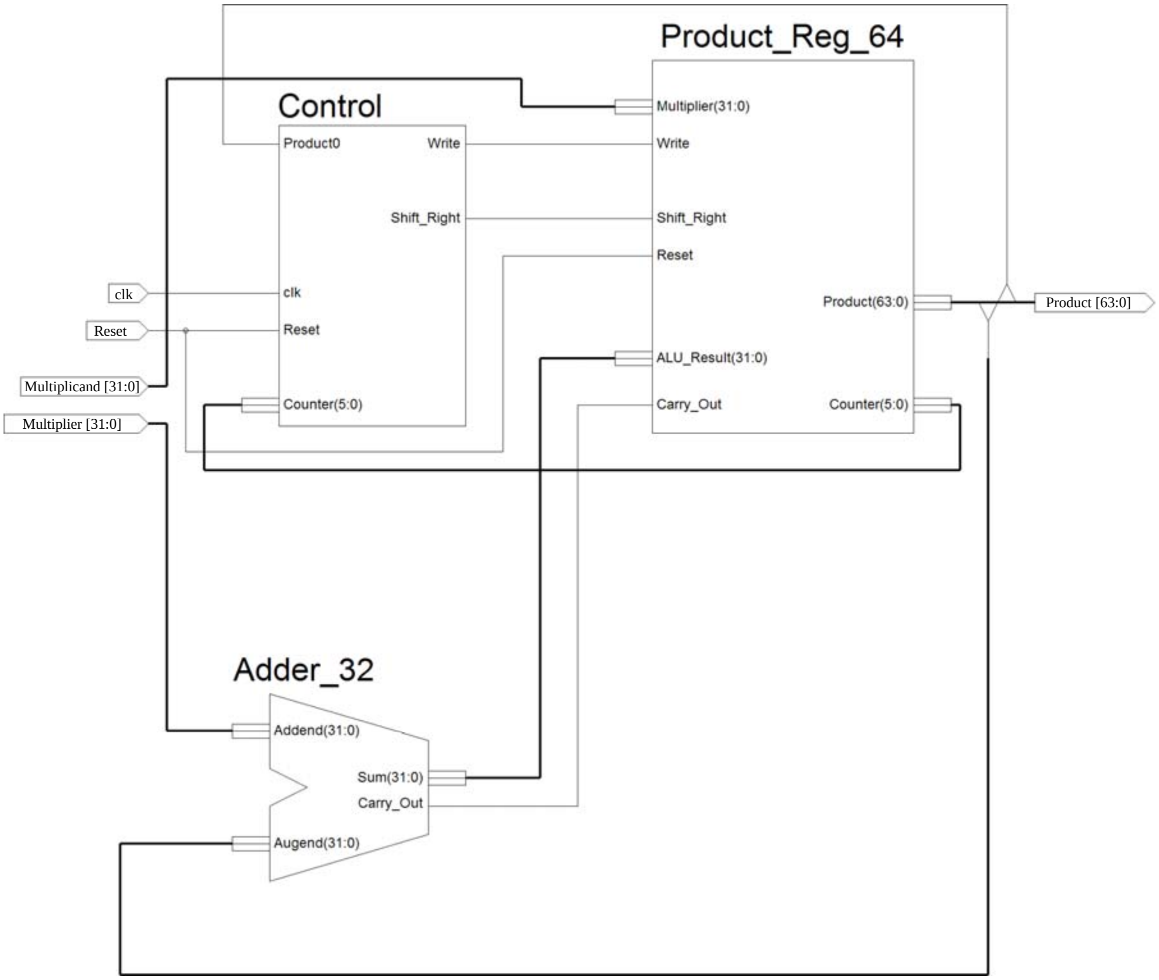


ASMD (Algorithmic State Machine and Datapath)



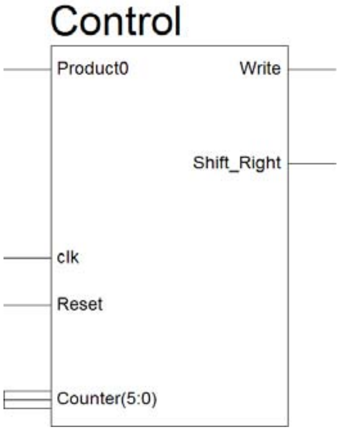
2. **(90 points) Implement the shift-add multiplication hardware using the multiplication algorithm developed in Part I. The datapath of the multiplication hardware is in Figure 3.7 of the textbook. Please design the multiplication hardware in either Verilog or VHDL.**

   Top-level block diagram



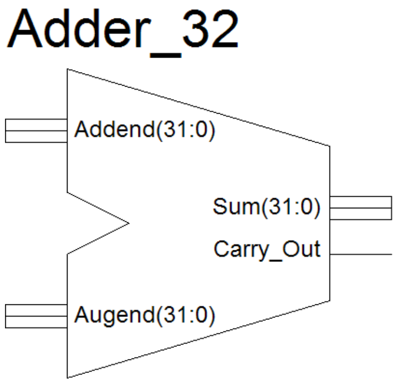
Design source codes in Verilog

Control Module



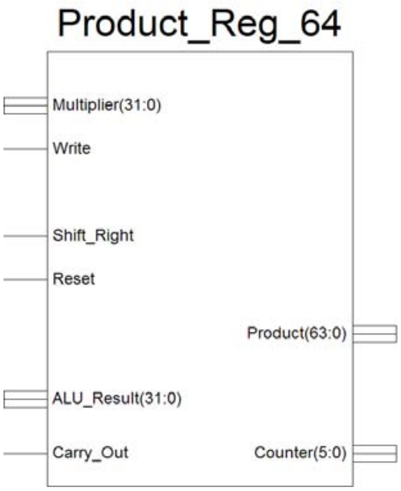
```
////////////////////////////////////////////////////////////////////////////////
// Company:             Temple University
// Student:             Zexi Liu
// Create Date:         16:51:48 03/05/2007
// Design Name:         Shift_Add_Multiplication
// Module Name:         Control
// Project Name:        Shift_Add_Multiplication
// Target Devices:      Spartan-3 XC3S400
// Tool versions:       ISE 8.2i
// Description:         This module generates all the control signal
////////////////////////////////////////////////////////////////////////////////
module Control(Write, clk, Counter, Reset, Shift_Right, Product0);
    input clk;                      // clock signal
    input Product0;                 // the LSB of the 64-bit Product Register
    input Reset;                    // connect to "Reset button"
    input [5:0] Counter;            // a 6-bit counter
    output Write;                   // determine whether add multiplicand to the left half of product or not
    output Shift_Right;             // clock signal for the 64-bit Product Register
    wire Shift_Right;               // the type must be "wire", since "assign" is used
    wire Write;                     // the type must be "wire", since "assign" is used
    reg Shift_Right_Enable;         // enable signal

    always @(negedge clk or posedge Reset)              // negative edge of clk
    begin
        if(Reset == 1)                                  // Reset
        begin
            Shift_Right_Enable <= 0;                    // disabled
        end
        else
        begin
            if(Counter < 31)
            begin
                Shift_Right_Enable <= 1;                // enable 32 clk circle
            end
            else
            begin
                Shift_Right_Enable <= 0;
            end
        end
    end

    assign Write = Product0;                            // controls the 64-bit Product Register
    assign Shift_Right = Shift_Right_Enable && clk;     // enable/disable the clk signal

endmodule
```

32-bit Adder



```
////////////////////////////////////////////////////////////////////////////////
// Company:              Temple University
// Student:              Zexi Liu
// Create Date:          11:25:48 03/05/2007
// Design Name:          Shift_Add_Multiplication
// Module Name:          ALU_32
// Project Name:         Shift_Add_Multiplication
// Target Devices:       Spartan-3 XC3S400
// Tool versions:        ISE 8.2i
// Description:          This is a 32-bit adder with carry out
////////////////////////////////////////////////////////////////////////////////
module Adder_32(Addend, Augend, Sum, Carry_Out);
    input[31:0]     Addend;        // 32-bit addend
    input [31:0]    Augend;        // 32-bit augend
    output [31:0]   Sum;           // 32-bit sum
    output          Carry_Out;     // 1-bit carry
    wire [31:0]     Sum;           // the type must be “wire”, since “assign” is used
    wire            Carry_Out;     // the type must be “wire”, since “assign” is used

    assign {Carry_Out, Sum} = Addend + Augend;      // operation

endmodule
```

64-bit Product Register



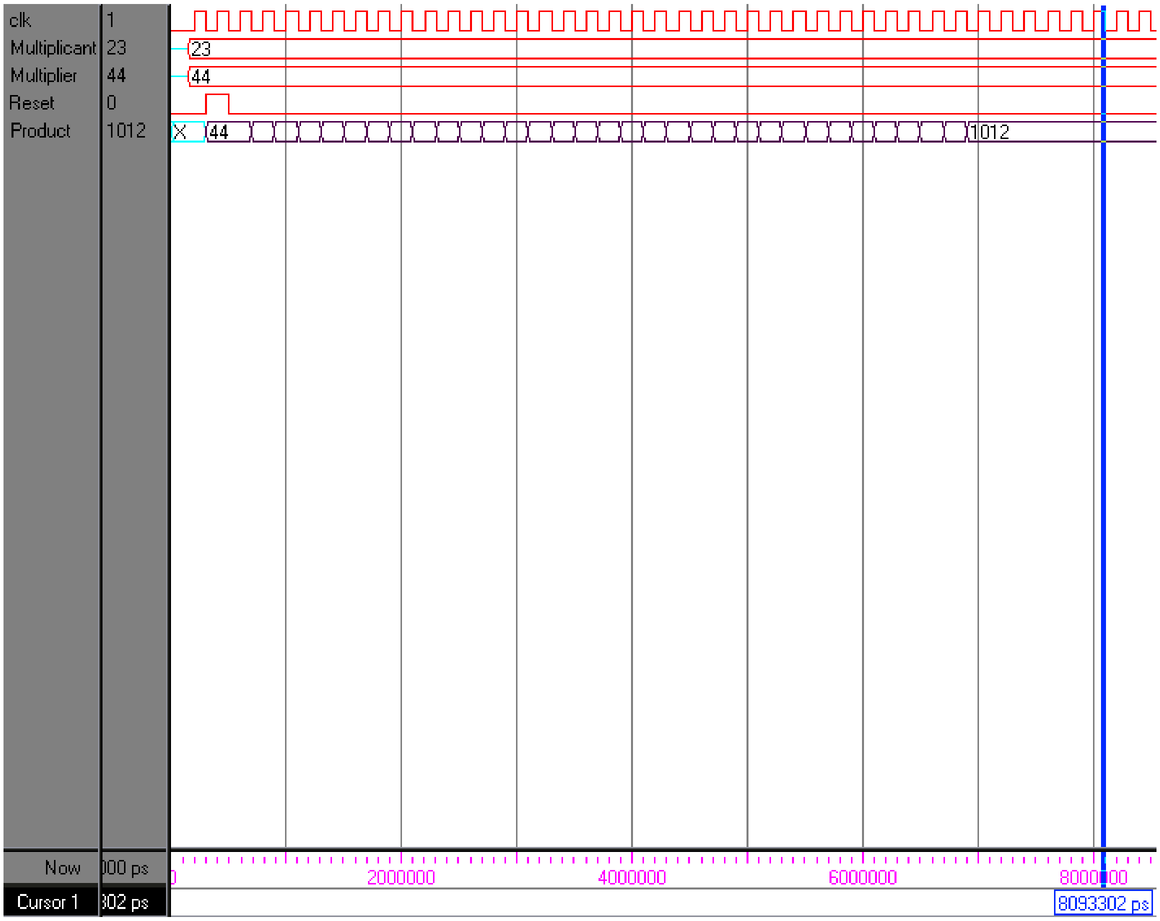
```
////////////////////////////////////////////////////////////////////////////////
// Company:                 Temple University
// Student:                 Zexi Liu
// Create Date:             14:36:02 03/05/2007
// Design Name:             Shift_Add_Multiplication
// Module Name:             Product_Reg_64
// Project Name:            Shift_Add_Multiplication
// Target Devices:          Spartan-3 XC3S400
// Tool versions:           ISE 8.2i
// Description:             This is a 64-bit Product Register
////////////////////////////////////////////////////////////////////////////////
module Product_Reg_64(ALU_Result, Multiplier, Reset, Write,
                      Carry_Out, Shift_Right, Product, Counter);
    input [31:0] ALU_Result;        // sum from the 32-bit adder
    input [31:0] Multiplier;        // multiplier as an input
    input Write;                    // from control module
    input Shift_Right;              // from control module, actually is the clk
    input Reset;                    // connect to “Reset bottom”
    input Carry_Out;                // carry from the 32-bit adder
    output [63:0] Product;          // the finial result, product
    output [5:0] Counter;           // 6-bit counter output
    reg [63:0] Product;             // the type must be “reg”, since register is involved
    reg [5:0] Counter;              // the type must be “reg”, since register is involved

    always @(negedge Shift_Right or posedge Reset)    // negative edge
    begin
        if(Reset == 1)                                // Reset
        begin
            Product <= {32'd0, Multiplier[31:0]};     // initialized
            Counter <= 0;                             // initialized
        end
        else
        begin
            if(Write == 1)                            // test product [0]
            begin
                Product <= {Carry_Out, ALU_Result[31:0], Product[31:1]};   // shift right 1 bit
            End                                                            // MSB is carry
            else
            begin
                Product <= {1'b0, Product[63:1]};                          // or shift right 1 bit
            end                                                            // MSB is 0
            Counter <= Counter + 1;
        end
    end
endmodule
```

Testbench for Top-level design

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// Company:             Temple University
// Student:             Zexi Liu
// Create Date:         Mon Mar 12 15:47:02 2007
// File Name:           Shift_Add_Multiplication_top_tbw_tb_0.v
// Project Name:        Shift_Add_Multiplication
// Target Devices:      Spartan-3 XC3S400
// Tool versions:       ISE 8.2i
// Description:         This is the Testbench for the top-level design
////////////////////////////////////////////////////////////////////////////////////////////////////
`timescale 1ns/1ps
`define NULL 0

module Shift_Add_Multiplication_top_tbw_tb_0;
    integer InputFile, OutputFile;          // file handles
    integer char, ret;                      // return value
    reg clk = 1'b0;                         // clock initial value is 0
    reg [31:0] Multiplicand;                // Multiplicand
    reg [31:0] Multiplier;                  // Multiplier
    reg Reset = 1'b0;                       // Reset initial value is 0
    wire signed [63:0] Product;             // the finial result

    parameter PERIOD = 200;                 // clock period is 200 ns
    parameter real DUTY_CYCLE = 0.5;        // 50%
    parameter OFFSET = 100;                 // offset 100 ns

    Shift_Add_Multiplication_top UUT (      // instantiate the Unit Under Test (UUT)
        .clk(clk),                          // I/O signals
        .Multiplicant(Multiplicant[31:0]),
        .Multiplier(Multiplier[31:0]),
        .Reset(Reset),
        .Product(Product));

    initial                                 // Clock process for clk
    begin
        #OFFSET;
        forever
        begin
            clk = 1'b0;
            #(PERIOD-(PERIOD*DUTY_CYCLE)) clk = 1'b1;
            #(PERIOD*DUTY_CYCLE);
        end
    end

    task Init_Reset;                        // generate a reset signal
    begin
        @(negedge clk);
            Reset = 1'b1;
        @(negedge clk);
            Reset = 1'b0;
    end
    endtask
```

```
        task Read_Multiplicand_and_Multiplier;  // read Multiplicand and Multiplier from a txt file
        begin
                ret = $fscanf(InputFile, "%d %d\n", Multiplicant, Multiplier);
        end                                     // “ret = ” is essential, without this, $fscanf can’t be recognized
        endtask

        initial
                begin : File_Block
                #150                                            // delay 150 ns
               $display("Testbench Started!");                 // system task
               InputFile = $fopen("ShAdMultIN.txt", "r");      // open the input file
               OutputFile = $fopen("ShAdMultOUT.txt", "w");    // open the output file
               if (InputFile == `NULL)                         // If error opening file
               begin
                       $display("Test file not exist!\n");     // display an error message
                       disable File_Block;                     // Just quit
               end
                Read_Multiplicant_and_Multiplier;              // read from the input file
                #150                                           // delay 150 ns
                Init_Reset;                                    // generates a reset signal
                #9000                                          // delay 9000 ns
                $display("Product = %h", Product);             // display result
                $fdisplay(OutputFile, "%h", Product);          // save the result
                #400                                           // delay 400 ns
                $display("Testbench Done!");                   // display an message
                $fclose(InputFile);                            // close files
                $fclose(OutputFile);
        end

endmodule
```

Behavioral simulation waveforms of the top-level design



clk 1
Multiplicant 23
Multiplier 44
Reset 0
Product 1012
Now 000 ps
Cursor 1 302 ps

Synthesis report using Xilinx Spartan-3 XC3S400 as the target device.

```
Release    - xst I.31
Copyright (c) 1995-2006 Xilinx, Inc.    All rights reserved.
--> Parameter TMPDIR set to ./xst/projnav.tmp
CPU : 0.00 / 0.23 s | Elapsed : 0.00 / 0.00 s

--> Parameter xsthdpdir set to ./xst
CPU : 0.00 / 0.23 s | Elapsed : 0.00 / 0.00 s

--> Reading design: Shift_Add_Multiplication_top.prj

TABLE OF CONTENTS
  1) Synthesis Options Summary
  2) HDL Compilation
  3) Design Hierarchy Analysis
  4) HDL Analysis
  5) HDL Synthesis
     5.1) HDL Synthesis Report
  6) Advanced HDL Synthesis
     6.1) Advanced HDL Synthesis Report
  7) Low Level Synthesis
  8) Partition Report
  9) Final Report
     9.1) Device utilization summary
     9.2) TIMING REPORT
```

```
=========================================================================
*                      Synthesis Options Summary                      *
=========================================================================
---- Source Parameters
Input File Name                    : "Shift_Add_Multiplication_top.prj"
Input Format                       : mixed
Ignore Synthesis Constraint File   : NO

---- Target Parameters
Output File Name                   : "Shift_Add_Multiplication_top"
Output Format                      : NGC
Target Device                      : xc3s400-5-ft256

---- Source Options
Top Module Name                    : Shift_Add_Multiplication_top
Automatic FSM Extraction           : YES
FSM Encoding Algorithm             : Auto
FSM Style                          : lut
RAM Extraction                     : Yes
RAM Style                          : Auto
ROM Extraction                     : Yes
Mux Style                          : Auto
Decoder Extraction                 : YES
Priority Encoder Extraction        : YES
Shift Register Extraction          : YES
Logical Shifter Extraction         : YES
XOR Collapsing                     : YES
ROM Style                          : Auto
Mux Extraction                     : YES
Resource Sharing                   : YES
Multiplier Style                   : auto
Automatic Register Balancing       : No

---- Target Options
Add IO Buffers                     : YES
Global Maximum Fanout              : 500
Add Generic Clock Buffer(BUFG)     : 8
Register Duplication               : YES
Slice Packing                      : YES
Pack IO Registers into IOBs        : auto
Equivalent register Removal        : YES

---- General Options
Optimization Goal                  : Speed
Optimization Effort                : 1
Keep Hierarchy                     : NO
RTL Output                         : Yes
Global Optimization                : AllClockNets
Write Timing Constraints           : NO
Hierarchy Separator                : /
Bus Delimiter                      : <>
Case Specifier                     : maintain
Slice Utilization Ratio            : 100
Slice Utilization Ratio Delta      : 5

---- Other Options
lso                                : Shift_Add_Multiplication_top.lso
Read Cores                         : YES
cross_clock_analysis               : NO
verilog2001                        : YES
safe_implementation                : No
Optimize Instantiated Primitives   : NO
use_clock_enable                   : Yes
use_sync_set                       : Yes
use_sync_reset                     : Yes
=========================================================================

=========================================================================
*                          HDL Compilation                               *
=========================================================================
Compiling verilog file "Product_Reg_64.v" in library work
Compiling verilog file "Control.v" in library work
Module <Product_Reg_64> compiled
Compiling verilog file "Adder_32.v" in library work
Module <Control> compiled
Compiling verilog file "Shift_Add_Multiplication_top.vf" in library work
Module <Adder_32> compiled
Module <Shift_Add_Multiplication_top> compiled
No errors in compilation
Analysis of file <"Shift_Add_Multiplication_top.prj"> succeeded.
=========================================================================
*                     Design Hierarchy Analysis                          *
=========================================================================
Analyzing hierarchy for module <Shift_Add_Multiplication_top> in library <work>.
Analyzing hierarchy for module <Adder_32> in library <work>.
Analyzing hierarchy for module <Control> in library <work>.
Analyzing hierarchy for module <Product_Reg_64> in library <work>.
Building hierarchy successfully finished.
=========================================================================
*                            HDL Analysis                                *
=========================================================================
Analyzing top module <Shift_Add_Multiplication_top>.
Module <Shift_Add_Multiplication_top> is correct for synthesis.

Analyzing module <Adder_32> in library <work>.
Module <Adder_32> is correct for synthesis.

Analyzing module <Control> in library <work>.
Module <Control> is correct for synthesis.

Analyzing module <Product_Reg_64> in library <work>.
Module <Product_Reg_64> is correct for synthesis.
=========================================================================
*                           HDL Synthesis                                *
=========================================================================

Performing bidirectional port resolution...

Synthesizing Unit <Adder_32>.
    Related source file is "Adder_32.v".
    Found 32-bit adder carry out for signal <$addsub0000>.
    Summary:
 inferred   1 Adder/Subtractor(s).
Unit <Adder_32> synthesized.

Synthesizing Unit <Control>.
    Related source file is "Control.v".
    Found 6-bit comparator less for signal <$cmp_lt0000> created at line 39.
    Found 1-bit register for signal <Shift_Right_Enable>.
    Summary:
 inferred   1 D-type flip-flop(s).
 inferred   1 Comparator(s).
Unit <Control> synthesized.

Synthesizing Unit <Product_Reg_64>.
    Related source file is "Product_Reg_64.v".
    Found 6-bit up counter for signal <Counter>.
    Found 64-bit register for signal <Product>.
    Found 64-bit 4-to-1 multiplexer for signal <$mux0000>.
    Summary:
 inferred   1 Counter(s).
 inferred  64 Multiplexer(s).
Unit <Product_Reg_64> synthesized.

Synthesizing Unit <Shift_Add_Multiplication_top>.
    Related source file is "Shift_Add_Multiplication_top.vf".
```

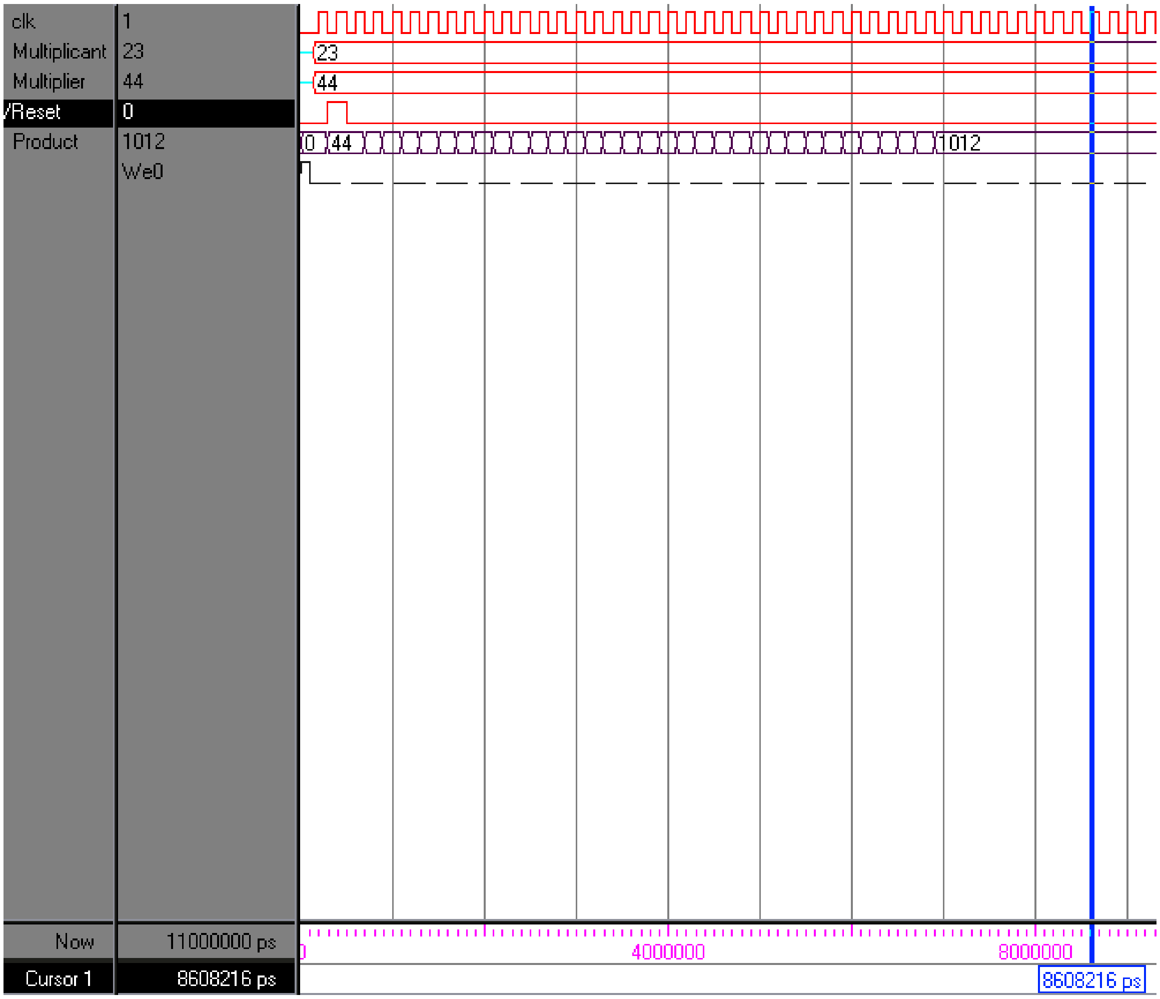
```
Unit <Shift_Add_Multiplication_top> synthesized.
=========================================================================
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors                                   : 1
  32-bit adder carry out                               : 1
# Counters                                             : 1
  6-bit up counter                                     : 1
# Registers                                            : 2
  1-bit register                                       : 1
  64-bit register                                      : 1
# Comparators                                          : 1
  6-bit comparator less                                : 1
# Multiplexers                                         : 1
  64-bit 4-to-1 multiplexer                            : 1
=========================================================================

=========================================================================
*                       Advanced HDL Synthesis                          *
=========================================================================
Loading device for application Rf_Device from file '3s400.nph' in environment F:\Xilinx.
=========================================================================
Advanced HDL Synthesis Report
Macro Statistics
# Adders/Subtractors                                   : 1
  32-bit adder carry out                               : 1
# Counters                                             : 1
  6-bit up counter                                     : 1
# Registers                                            : 2
  Flip-Flops                                           : 2
# Comparators                                          : 1
  6-bit comparator less                                : 1
# Multiplexers                                         : 1
  64-bit 4-to-1 multiplexer                            : 1
=========================================================================

=========================================================================
*                         Low Level Synthesis                           *
=========================================================================
Optimizing unit <Shift_Add_Multiplication_top> ...

Mapping all equations...
Building and optimizing final netlist ...
Found area constraint ratio of 100 (+ 5) on block Shift_Add_Multiplication_top, actual ratio is 2.
Final Macro Processing ...
=========================================================================
Final Register Report

Macro Statistics
# Registers                                            : 71
  Flip-Flops                                           : 71
=========================================================================

=========================================================================
*                           Partition Report                            *
=========================================================================

Partition Implementation Status
-------------------------------
  No Partitions were found in this design.
-------------------------------
=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : Shift_Add_Multiplication_top.ngr
Top Level Output File Name         : Shift_Add_Multiplication_top
```

```
Output Format                              : NGC
Optimization Goal                          : Speed
Keep Hierarchy                             : NO
Design Statistics
# IOs                                      : 130
Cell Usage :
# BELS                                     : 236
#      GND                                 : 1
#      INV                                 : 1
#      LUT2                                : 100
#      LUT3                                : 36
#      LUT4                                : 34
#      MUXCY                               : 32
#      MUXF5                               : 1
#      XORCY                               : 31
# FlipFlops/Latches                        : 71
#      FDC_1                               : 7
#      FDCP                                : 64
# Clock Buffers                            : 2
#      BUFG                                : 1
#      BUFGP                               : 1
# IO Buffers                               : 129
#      IBUF                                : 65
#      OBUF                                : 64
=========================================================================
Device utilization summary:
---------------------------
Selected Device : 3s400ft256-5

 Number of Slices:                       96  out of   3584     2%
 Number of Slice Flip Flops:             71  out of   7168     0%
 Number of 4 input LUTs:                171  out of   7168     2%
 Number of IOs:                         130
 Number of bonded IOBs:                 130  out of    173    75%
 Number of GCLKs:                         2  out of      8    25%
=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.
Clock Information:
------------------
----------------------------------------------------------------------+---------------------------------+-------+
Clock Signal                                                          | Clock buffer(FF name)           | Load  |
----------------------------------------------------------------------+---------------------------------+-------+
clk                                                                   | BUFGP                           | 1     |
XLXN_101(Ctrl/Shift_Right1:O)                                         | NONE(*)(ProductReg64/Counter_4) | 6     |
ProductReg64/Product_63__not00011(ProductReg64/Product_63__not00011:O)| BUFG(*)(ProductReg64/Product_36)| 64    |
----------------------------------------------------------------------+---------------------------------+-------+
(*) These 2 clock signal(s) are generated by combinatorial logic,
and XST is not able to identify which are the primary clock signals.
Please use the CLOCK_SIGNAL constraint to specify the clock signal(s) generated by combinatorial logic.
INFO:Xst:2169 - HDL ADVISOR - Some clock signals were not automatically buffered by XST with BUFG/BUFR resources. Please use the buffer_type constraint in order to insert these buffers to the clock signals to help prevent skew problems.
Asynchronous Control Signals Information:
----------------------------------------
---------------------------------------------------------------------+-----------------------------+-------+
Control Signal                                                       | Buffer(FF name)             | Load  |
---------------------------------------------------------------------+-----------------------------+-------+
Reset                                                                | IBUF                        | 39    |
ProductReg64/Product_63__and0001(XST_GND:G)                          | NONE(ProductReg64/Product_36)| 32    |
ProductReg64/Product_30__and0000(ProductReg64/Product_30__and00001:O)| NONE(ProductReg64/Product_30)| 1     |
ProductReg64/Product_30__and0001(ProductReg64/Product_30__and00011:O)| NONE(ProductReg64/Product_30)| 1     |
ProductReg64/Product_6__and0000(ProductReg64/Product_6__and00001:O)  | NONE(ProductReg64/Product_6) | 1     |
ProductReg64/Product_6__and0001(ProductReg64/Product_6__and00011:O)  | NONE(ProductReg64/Product_6) | 1     |
ProductReg64/Product_13__and0000(ProductReg64/Product_13__and00001:O)| NONE(ProductReg64/Product_13)| 1     |
ProductReg64/Product_13__and0001(ProductReg64/Product_13__and00011:O)| NONE(ProductReg64/Product_13)| 1     |
ProductReg64/Product_24__and0000(ProductReg64/Product_24__and00001:O)| NONE(ProductReg64/Product_24)| 1     |
ProductReg64/Product_24__and0001(ProductReg64/Product_24__and00011:O)| NONE(ProductReg64/Product_24)| 1     |
ProductReg64/Product_3__and0000(ProductReg64/Product_3__and00001:O)  | NONE(ProductReg64/Product_3) | 1     |
ProductReg64/Product_3__and0001(ProductReg64/Product_3__and00011:O)  | NONE(ProductReg64/Product_3) | 1     |
ProductReg64/Product_26__and0000(ProductReg64/Product_26__and00001:O)| NONE(ProductReg64/Product_26)| 1     |
ProductReg64/Product_26__and0001(ProductReg64/Product_26__and00011:O)| NONE(ProductReg64/Product_26)| 1     |
ProductReg64/Product_16__and0000(ProductReg64/Product_16__and00001:O)| NONE(ProductReg64/Product_16)| 1     |
ProductReg64/Product_16__and0001(ProductReg64/Product_16__and00011:O)| NONE(ProductReg64/Product_16)| 1     |
ProductReg64/Product_0__and0000(ProductReg64/Product_0__and00001:O)  | NONE(ProductReg64/Product_0) | 1     |
ProductReg64/Product_0__and0001(ProductReg64/Product_0__and00011:O)  | NONE(ProductReg64/Product_0) | 1     |
ProductReg64/Product_29__and0000(ProductReg64/Product_29__and00001:O)| NONE(ProductReg64/Product_29)| 1     |
ProductReg64/Product_29__and0001(ProductReg64/Product_29__and00011:O)| NONE(ProductReg64/Product_29)| 1     |
ProductReg64/Product_10__and0000(ProductReg64/Product_10__and00001:O)| NONE(ProductReg64/Product_10)| 1     |
ProductReg64/Product_10__and0001(ProductReg64/Product_10__and00011:O)| NONE(ProductReg64/Product_10)| 1     |
ProductReg64/Product_19__and0000(ProductReg64/Product_19__and00001:O)| NONE(ProductReg64/Product_19)| 1     |
ProductReg64/Product_19__and0001(ProductReg64/Product_19__and00011:O)| NONE(ProductReg64/Product_19)| 1     |
ProductReg64/Product_31__and0000(ProductReg64/Product_31__and00001:O)| NONE(ProductReg64/Product_31)| 1     |
ProductReg64/Product_31__and0001(ProductReg64/Product_31__and00011:O)| NONE(ProductReg64/Product_31)| 1     |
ProductReg64/Product_7__and0000(ProductReg64/Product_7__and00001:O)  | NONE(ProductReg64/Product_7) | 1     |
ProductReg64/Product_7__and0001(ProductReg64/Product_7__and00011:O)  | NONE(ProductReg64/Product_7) | 1     |
ProductReg64/Product_12__and0000(ProductReg64/Product_12__and00001:O)| NONE(ProductReg64/Product_12)| 1     |
ProductReg64/Product_12__and0001(ProductReg64/Product_12__and00011:O)| NONE(ProductReg64/Product_12)| 1     |
ProductReg64/Product_21__and0000(ProductReg64/Product_21__and00001:O)| NONE(ProductReg64/Product_21)| 1     |
ProductReg64/Product_21__and0001(ProductReg64/Product_21__and00011:O)| NONE(ProductReg64/Product_21)| 1     |
ProductReg64/Product_23__and0000(ProductReg64/Product_23__and00001:O)| NONE(ProductReg64/Product_23)| 1     |
ProductReg64/Product_23__and0001(ProductReg64/Product_23__and00011:O)| NONE(ProductReg64/Product_23)| 1     |
ProductReg64/Product_4__and0000(ProductReg64/Product_4__and00001:O)  | NONE(ProductReg64/Product_4) | 1     |
ProductReg64/Product_4__and0001(ProductReg64/Product_4__and00011:O)  | NONE(ProductReg64/Product_4) | 1     |
ProductReg64/Product_1__and0000(ProductReg64/Product_1__and00001:O)  | NONE(ProductReg64/Product_1) | 1     |
ProductReg64/Product_1__and0001(ProductReg64/Product_1__and00011:O)  | NONE(ProductReg64/Product_1) | 1     |
ProductReg64/Product_25__and0000(ProductReg64/Product_25__and00001:O)| NONE(ProductReg64/Product_25)| 1     |
ProductReg64/Product_25__and0001(ProductReg64/Product_25__and00011:O)| NONE(ProductReg64/Product_25)| 1     |
ProductReg64/Product_15__and0000(ProductReg64/Product_15__and00001:O)| NONE(ProductReg64/Product_15)| 1     |
ProductReg64/Product_15__and0001(ProductReg64/Product_15__and00011:O)| NONE(ProductReg64/Product_15)| 1     |
ProductReg64/Product_28__and0000(ProductReg64/Product_28__and00001:O)| NONE(ProductReg64/Product_28)| 1     |
ProductReg64/Product_28__and0001(ProductReg64/Product_28__and00011:O)| NONE(ProductReg64/Product_28)| 1     |
ProductReg64/Product_11__and0000(ProductReg64/Product_11__and00001:O)| NONE(ProductReg64/Product_11)| 1     |
ProductReg64/Product_11__and0001(ProductReg64/Product_11__and00011:O)| NONE(ProductReg64/Product_11)| 1     |
ProductReg64/Product_18__and0000(ProductReg64/Product_18__and00001:O)| NONE(ProductReg64/Product_18)| 1     |
ProductReg64/Product_18__and0001(ProductReg64/Product_18__and00011:O)| NONE(ProductReg64/Product_18)| 1     |
ProductReg64/Product_8__and0000(ProductReg64/Product_8__and00001:O)  | NONE(ProductReg64/Product_8) | 1     |
ProductReg64/Product_8__and0001(ProductReg64/Product_8__and00011:O)  | NONE(ProductReg64/Product_8) | 1     |
ProductReg64/Product_5__and0000(ProductReg64/Product_5__and00001:O)  | NONE(ProductReg64/Product_5) | 1     |
ProductReg64/Product_5__and0001(ProductReg64/Product_5__and00011:O)  | NONE(ProductReg64/Product_5) | 1     |
ProductReg64/Product_2__and0000(ProductReg64/Product_2__and00001:O)  | NONE(ProductReg64/Product_2) | 1     |
ProductReg64/Product_2__and0001(ProductReg64/Product_2__and00011:O)  | NONE(ProductReg64/Product_2) | 1     |
ProductReg64/Product_27__and0000(ProductReg64/Product_27__and00001:O)| NONE(ProductReg64/Product_27)| 1     |
ProductReg64/Product_27__and0001(ProductReg64/Product_27__and00011:O)| NONE(ProductReg64/Product_27)| 1     |
ProductReg64/Product_22__and0000(ProductReg64/Product_22__and00001:O)| NONE(ProductReg64/Product_22)| 1     |
ProductReg64/Product_22__and0001(ProductReg64/Product_22__and00011:O)| NONE(ProductReg64/Product_22)| 1     |
ProductReg64/Product_14__and0000(ProductReg64/Product_14__and00001:O)| NONE(ProductReg64/Product_14)| 1     |
ProductReg64/Product_14__and0001(ProductReg64/Product_14__and00011:O)| NONE(ProductReg64/Product_14)| 1     |
ProductReg64/Product_17__and0000(ProductReg64/Product_17__and00001:O)| NONE(ProductReg64/Product_17)| 1     |
ProductReg64/Product_17__and0001(ProductReg64/Product_17__and00011:O)| NONE(ProductReg64/Product_17)| 1     |
ProductReg64/Product_9__and0000(ProductReg64/Product_9__and00001:O)  | NONE(ProductReg64/Product_9) | 1     |
ProductReg64/Product_9__and0001(ProductReg64/Product_9__and00011:O)  | NONE(ProductReg64/Product_9) | 1     |
ProductReg64/Product_20__and0000(ProductReg64/Product_20__and00001:O)| NONE(ProductReg64/Product_20)| 1     |
ProductReg64/Product_20__and0001(ProductReg64/Product_20__and00011:O)| NONE(ProductReg64/Product_20)| 1     |
---------------------------------------------------------------------+-----------------------------+-------+
Timing Summary:
---------------
Speed Grade: -5
   Minimum period: 6.291ns (Maximum Frequency: 158.962MHz)
   Minimum input arrival time before clock: 6.479ns
   Maximum output required time after clock: 7.120ns
   Maximum combinational path delay: No path found
Timing Detail:
--------------
```

```
All values displayed in nanoseconds (ns)
=========================================================================
Timing constraint: Default period analysis for Clock 'XLXN_101'
  Clock period: 3.741ns (frequency: 267.280MHz)
  Total number of paths / destination ports: 21 / 6
-------------------------------------------------------------------------
Delay:               3.741ns (Levels of Logic = 2)
  Source:            ProductReg64/Counter_3 (FF)
  Destination:       ProductReg64/Counter_4 (FF)
  Source Clock:      XLXN_101 falling
  Destination Clock: XLXN_101 falling
  Data Path: ProductReg64/Counter_3 to ProductReg64/Counter_4
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDC_1:C->Q            3   0.626   1.066  ProductReg64/Counter_3 (ProductReg64/Counter_3)
     LUT4:I0->O            2   0.479   0.915  ProductReg64/Mcount_Counter_cy<3>11 (ProductReg64/Mcount_Counter_cy<3>)
     LUT2:I1->O            1   0.479   0.000  ProductReg64/Mcount_Counter_xor<4>11 (Result<4>)
     FDC_1:D                   0.176          ProductReg64/Counter_4
    ----------------------------------------
    Total                      3.741ns (1.760ns logic, 1.981ns route)
                                       (47.0% logic, 53.0% route)
=========================================================================
Timing constraint: Default period analysis for Clock 'ProductReg64/Product_63__not00011'
  Clock period: 6.291ns (frequency: 158.962MHz)
  Total number of paths / destination ports: 655 / 64
-------------------------------------------------------------------------
Delay:               6.291ns (Levels of Logic = 34)
  Source:            ProductReg64/Product_32 (FF)
  Destination:       ProductReg64/Product_62 (FF)
  Source Clock:      ProductReg64/Product_63__not00011 rising
  Destination Clock: ProductReg64/Product_63__not00011 rising

  Data Path: ProductReg64/Product_32 to ProductReg64/Product_62
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     FDCP:C->Q             3   0.626   0.941  ProductReg64/Product_32 (ProductReg64/Product_32)
     LUT2:I1->O            1   0.479   0.000  Add32/Madd__addsub0000_lut<0> (XLXN_108<0>)
     MUXCY:S->O            1   0.435   0.000  Add32/Madd__addsub0000_cy<0> (Add32/Madd__addsub0000_cy<0>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<1> (Add32/Madd__addsub0000_cy<1>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<2> (Add32/Madd__addsub0000_cy<2>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<3> (Add32/Madd__addsub0000_cy<3>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<4> (Add32/Madd__addsub0000_cy<4>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<5> (Add32/Madd__addsub0000_cy<5>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<6> (Add32/Madd__addsub0000_cy<6>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<7> (Add32/Madd__addsub0000_cy<7>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<8> (Add32/Madd__addsub0000_cy<8>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<9> (Add32/Madd__addsub0000_cy<9>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<10> (Add32/Madd__addsub0000_cy<10>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<11> (Add32/Madd__addsub0000_cy<11>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<12> (Add32/Madd__addsub0000_cy<12>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<13> (Add32/Madd__addsub0000_cy<13>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<14> (Add32/Madd__addsub0000_cy<14>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<15> (Add32/Madd__addsub0000_cy<15>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<16> (Add32/Madd__addsub0000_cy<16>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<17> (Add32/Madd__addsub0000_cy<17>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<18> (Add32/Madd__addsub0000_cy<18>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<19> (Add32/Madd__addsub0000_cy<19>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<20> (Add32/Madd__addsub0000_cy<20>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<21> (Add32/Madd__addsub0000_cy<21>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<22> (Add32/Madd__addsub0000_cy<22>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<23> (Add32/Madd__addsub0000_cy<23>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<24> (Add32/Madd__addsub0000_cy<24>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<25> (Add32/Madd__addsub0000_cy<25>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<26> (Add32/Madd__addsub0000_cy<26>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<27> (Add32/Madd__addsub0000_cy<27>)
     MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<28> (Add32/Madd__addsub0000_cy<28>)
    MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<29> (Add32/Madd__addsub0000_cy<29>)
    MUXCY:CI->O           1   0.056   0.000  Add32/Madd__addsub0000_cy<30> (Add32/Madd__addsub0000_cy<30>)
    XORCY:CI->O           1   0.786   0.704  Add32/Madd__addsub0000_xor<31> (XLXN_108<31>)
    LUT4:I3->O            1   0.479   0.000  ProductReg64/_mux0000<62>1 (ProductReg64/_mux0000<62>)
    FDCP:D                    0.176          ProductReg64/Product_62
    ----------------------------------------
    Total                     6.291ns (4.646ns logic, 1.645ns route)
                                      (73.9% logic, 26.1% route)
=========================================================================
Timing constraint: Default OFFSET IN BEFORE for Clock 'ProductReg64/Product_63__not00011'
  Total number of paths / destination ports: 1184 / 64
-------------------------------------------------------------------------
Offset:              6.479ns (Levels of Logic = 35)
  Source:            Multiplicant<0> (PAD)
  Destination:       ProductReg64/Product_62 (FF)
  Destination Clock: ProductReg64/Product_63__not00011 rising

  Data Path: Multiplicant<0> to ProductReg64/Product_62
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
     IBUF:I->O            2   0.715   1.040  Multiplicant_0_IBUF (Multiplicant_0_IBUF)
     LUT2:I0->O           1   0.479   0.000  Add32/Madd__addsub0000_lut<0> (XLXN_108<0>)
     MUXCY:S->O           1   0.435   0.000  Add32/Madd__addsub0000_cy<0> (Add32/Madd__addsub0000_cy<0>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<1> (Add32/Madd__addsub0000_cy<1>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<2> (Add32/Madd__addsub0000_cy<2>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<3> (Add32/Madd__addsub0000_cy<3>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<4> (Add32/Madd__addsub0000_cy<4>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<5> (Add32/Madd__addsub0000_cy<5>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<6> (Add32/Madd__addsub0000_cy<6>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<7> (Add32/Madd__addsub0000_cy<7>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<8> (Add32/Madd__addsub0000_cy<8>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<9> (Add32/Madd__addsub0000_cy<9>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<10> (Add32/Madd__addsub0000_cy<10>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<11> (Add32/Madd__addsub0000_cy<11>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<12> (Add32/Madd__addsub0000_cy<12>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<13> (Add32/Madd__addsub0000_cy<13>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<14> (Add32/Madd__addsub0000_cy<14>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<15> (Add32/Madd__addsub0000_cy<15>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<16> (Add32/Madd__addsub0000_cy<16>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<17> (Add32/Madd__addsub0000_cy<17>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<18> (Add32/Madd__addsub0000_cy<18>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<19> (Add32/Madd__addsub0000_cy<19>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<20> (Add32/Madd__addsub0000_cy<20>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<21> (Add32/Madd__addsub0000_cy<21>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<22> (Add32/Madd__addsub0000_cy<22>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<23> (Add32/Madd__addsub0000_cy<23>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<24> (Add32/Madd__addsub0000_cy<24>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<25> (Add32/Madd__addsub0000_cy<25>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<26> (Add32/Madd__addsub0000_cy<26>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<27> (Add32/Madd__addsub0000_cy<27>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<28> (Add32/Madd__addsub0000_cy<28>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<29> (Add32/Madd__addsub0000_cy<29>)
     MUXCY:CI->O          1   0.056   0.000  Add32/Madd__addsub0000_cy<30> (Add32/Madd__addsub0000_cy<30>)
     XORCY:CI->O          1   0.786   0.704  Add32/Madd__addsub0000_xor<31> (XLXN_108<31>)
     LUT4:I3->O           1   0.479   0.000  ProductReg64/_mux0000<62>1 (ProductReg64/_mux0000<62>)
     FDCP:D                   0.176          ProductReg64/Product_62
    ----------------------------------------
    Total                     6.479ns (4.735ns logic, 1.744ns route)
                                      (73.1% logic, 26.9% route)
=========================================================================
Timing constraint: Default OFFSET OUT AFTER for Clock 'ProductReg64/Product_63__not00011'
  Total number of paths / destination ports: 64 / 64
-------------------------------------------------------------------------
Offset:              7.120ns (Levels of Logic = 1)
  Source:            ProductReg64/Product_0 (FF)
  Destination:       Product<0> (PAD)
  Source Clock:      ProductReg64/Product_63__not00011 rising
```

```
   Data Path: ProductReg64/Product_0 to Product<0>
                                         Gate     Net
     Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
     ----------------------------------------  ------------
       FDCP:C->Q            34   0.626   1.585  ProductReg64/Product_0 (ProductReg64/Product_0)
       OBUF:I->O                 4.909          Product_0_OBUF (Product<0>)
     ----------------------------------------
     Total                      7.120ns (5.535ns logic, 1.585ns route)
                                        (77.7% logic, 22.3% route)
=========================================================================
CPU : 7.19 / 7.45 s | Elapsed : 7.00 / 7.00 s
 -->
Total memory usage is 135944 kilobytes

Number of errors   :    0 (   0 filtered)
Number of warnings :    0 (   0 filtered)
Number of infos    :    1 (   0 filtered)
```

Post-synthesis simulation waveforms of the top-level design



clk 1
Multiplicant 23
Multiplier 44
/Reset 0
Product 1012
We0
Now 11000000 ps
Cursor 1 8608216 ps

Identify the critical path the design using the static timing analysis (STA) tool of Xilinx ISE.   (Static Timing Report)

```
Maximum delay is      5.449ns.
--------------------------------------------------------------------------------
Delay:                          5.449ns (data path)
Source:                         Ctrl/Shift_Right_Enable (FF)
     Destination:               ProductReg64/Product_16 (FF)
     Data Path Delay:           5.449ns (Levels of Logic = 2)
     Source Clock:              clk_BUFGP falling

     Data Path: Ctrl/Shift_Right_Enable to ProductReg64/Product_16
     Delay type                 Delay(ns)   Logical Resource(s)
     ----------------------------  -------------------------------------
     Tcko                       0.626     Ctrl/Shift_Right_Enable
     net (fanout=2)             0.775     Ctrl/Shift_Right_Enable
     Tilo                       0.479     ProductReg64/Product_63__not00011
     net (fanout=1)             2.667     ProductReg64/Product_63__not00011
     Tgi0o                      0.001     ProductReg64/Product_63__not0001_BUFG
     net (fanout=48)            0.901     ProductReg64/Product_63__not0001
     ----------------------------  -------------------------------------
     Total                      5.449ns     (1.106ns logic, 4.343ns route)
                                            (20.3% logic, 79.7% route)
```

Calculate the highest clock rate of the multiplication hardware.

The following is from Synthesis Report

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period:                          6.291ns (Maximum Frequency: 158.962MHz)
   Minimum input arrival time before clock: 6.479ns
   Maximum output required time after clock: 7.120ns
   Maximum combinational path delay:        No path found
```

However, by setting up Timing constraint: TS_clk = PERIOD TIMEGRP "clk" 5.5 ns HIGH 50%, there are still 0.011 ns slack

```
Slack:                  0.011ns (requirement - (data path - clock path skew + uncertainty))
  Source:               ProductReg64/Product_35 (FF)
  Destination:          ProductReg64/Product_61 (FF)
  Requirement:          5.500ns
  Data Path Delay:      5.489ns (Levels of Logic = 16)
  Clock Path Skew:      0.000ns
  Source Clock:         ProductReg64/Product_63__not0001 rising at 0.000ns
  Destination Clock:    ProductReg64/Product_63__not0001 rising at 5.500ns
  Clock Uncertainty:    0.000ns
```

So, the highest could be 5.5 ns (181.82 MHz)