|  |  |
| --- | --- |
| **22bec121** |  |
| **Experiment – 10** | **Date:- 4/4/2024** |

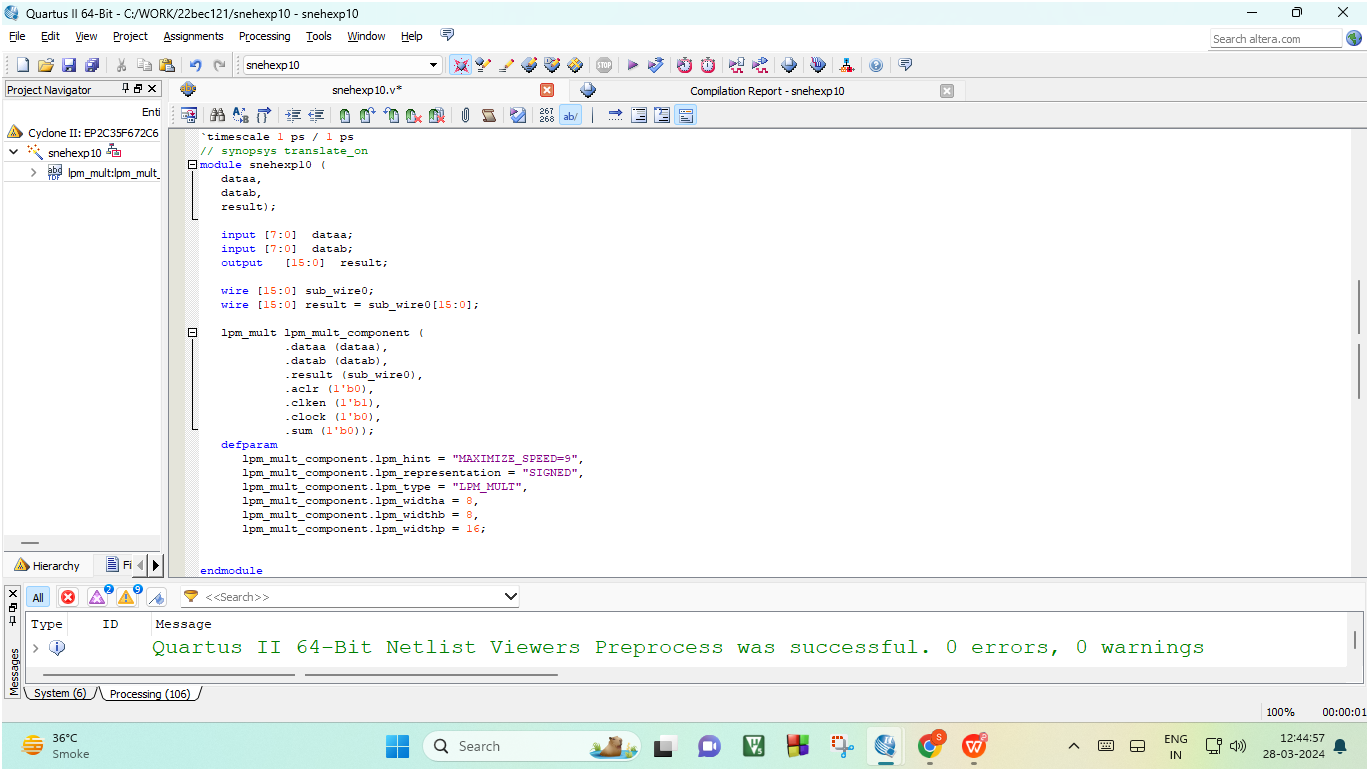
**Lab Work**

**Q1 .Design a 8-Bit Signed Multiplier using IPCores.**

# Code (IP Core 8-bit Signed Multiplier)

//IP Core Code

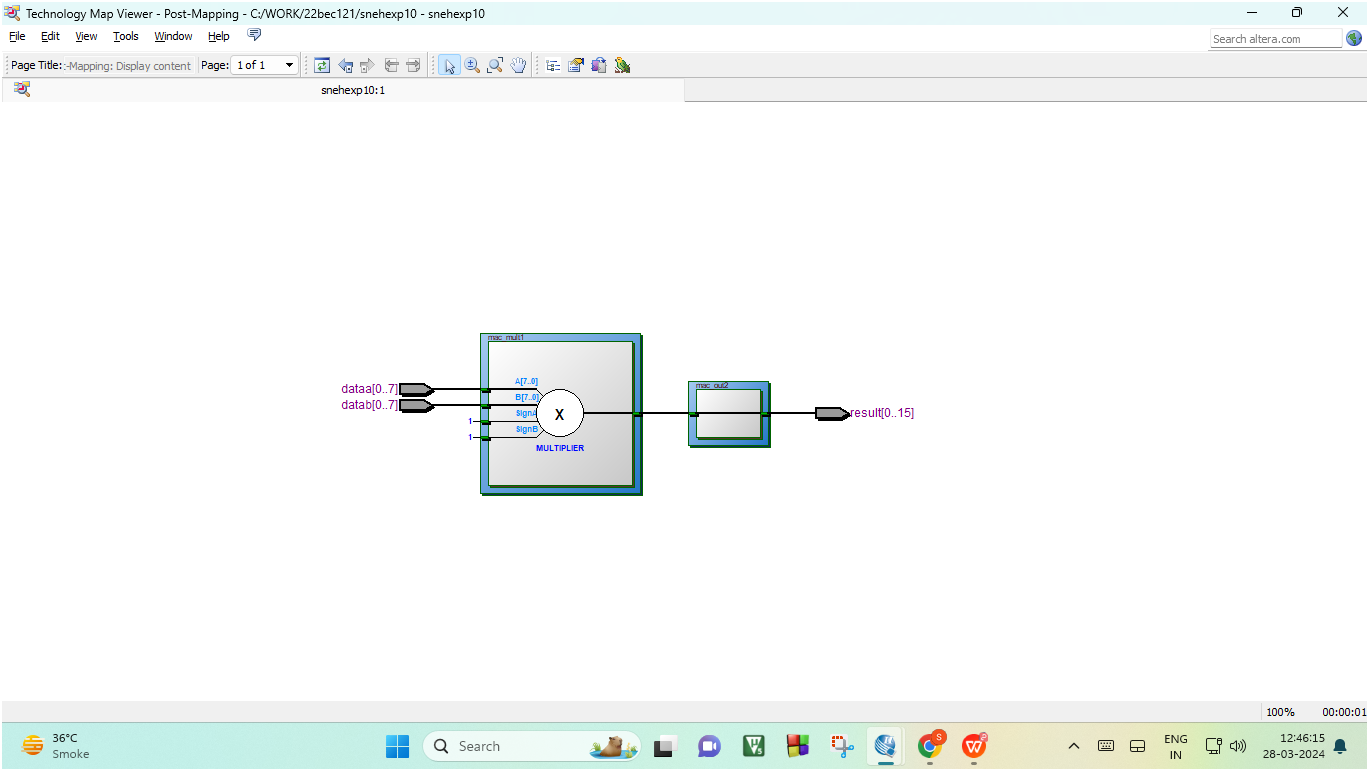
// megafunction wizard: %LPM\_MULT%  
// GENERATION: STANDARD  
// VERSION: WM1.0  
// MODULE: lpm\_mult  
  
// ============================================================  
// File Name: snehexp10.v  
// Megafunction Name(s):  
// lpm\_mult  
//  
// Simulation Library Files(s):  
// lpm  
// ============================================================  
// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
// THIS IS A WIZARD-GENERATED FILE. DO NOT EDIT THIS FILE!  
//  
// 13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition  
// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
  
//Copyright (C) 1991-2013 Altera Corporation  
//Your use of Altera Corporation's design tools, logic functions  
//and other software and tools, and its AMPP partner logic  
//functions, and any output files from any of the foregoing  
//(including device programming or simulation files), and any  
//associated documentation or information are expressly subject  
//to the terms and conditions of the Altera Program License  
//Subscription Agreement, Altera MegaCore Function License  
//Agreement, or other applicable license agreement, including,  
//without limitation, that your use is for the sole purpose of  
//programming logic devices manufactured by Altera and sold by  
//Altera or its authorized distributors.  Please refer to the  
//applicable agreement for further details.  
  
  
// synopsys translate\_off  
`timescale 1 ps / 1 ps  
// synopsys translate\_on  
module snehexp10 (  
dataa,  
datab,  
result);  
  
input [7:0]  dataa;  
input [7:0]  datab;  
output [15:0]  result;  
  
wire [15:0] sub\_wire0;  
wire [15:0] result = sub\_wire0[15:0];  
  
lpm\_mult lpm\_mult\_component (  
.dataa (dataa),  
.datab (datab),  
.result (sub\_wire0),  
.aclr (1'b0),  
.clken (1'b1),  
.clock (1'b0),  
.sum (1'b0));  
defparam  
lpm\_mult\_component.lpm\_hint = "MAXIMIZE\_SPEED=9",  
lpm\_mult\_component.lpm\_representation = "SIGNED",  
lpm\_mult\_component.lpm\_type = "LPM\_MULT",  
lpm\_mult\_component.lpm\_widtha = 8,  
lpm\_mult\_component.lpm\_widthb = 8,  
lpm\_mult\_component.lpm\_widthp = 16;  
  
  
endmodule  
  
// ============================================================  
// CNX file retrieval info  
// ============================================================  
// Retrieval info: PRIVATE: AutoSizeResult NUMERIC "1"  
// Retrieval info: PRIVATE: B\_isConstant NUMERIC "0"  
// Retrieval info: PRIVATE: ConstantB NUMERIC "0"  
// Retrieval info: PRIVATE: INTENDED\_DEVICE\_FAMILY STRING "Cyclone II"  
// Retrieval info: PRIVATE: LPM\_PIPELINE NUMERIC "0"  
// Retrieval info: PRIVATE: Latency NUMERIC "0"  
// Retrieval info: PRIVATE: SYNTH\_WRAPPER\_GEN\_POSTFIX STRING "1"  
// Retrieval info: PRIVATE: SignedMult NUMERIC "1"  
// Retrieval info: PRIVATE: USE\_MULT NUMERIC "1"  
// Retrieval info: PRIVATE: ValidConstant NUMERIC "0"  
// Retrieval info: PRIVATE: WidthA NUMERIC "8"  
// Retrieval info: PRIVATE: WidthB NUMERIC "8"  
// Retrieval info: PRIVATE: WidthP NUMERIC "16"  
// Retrieval info: PRIVATE: aclr NUMERIC "0"  
// Retrieval info: PRIVATE: clken NUMERIC "0"  
// Retrieval info: PRIVATE: new\_diagram STRING "1"  
// Retrieval info: PRIVATE: optimize NUMERIC "1"  
// Retrieval info: LIBRARY: lpm lpm.lpm\_components.all  
// Retrieval info: CONSTANT: LPM\_HINT STRING "MAXIMIZE\_SPEED=9"  
// Retrieval info: CONSTANT: LPM\_REPRESENTATION STRING "SIGNED"  
// Retrieval info: CONSTANT: LPM\_TYPE STRING "LPM\_MULT"  
// Retrieval info: CONSTANT: LPM\_WIDTHA NUMERIC "8"  
// Retrieval info: CONSTANT: LPM\_WIDTHB NUMERIC "8"  
// Retrieval info: CONSTANT: LPM\_WIDTHP NUMERIC "16"  
// Retrieval info: USED\_PORT: dataa 0 0 8 0 INPUT NODEFVAL "dataa[7..0]"  
// Retrieval info: USED\_PORT: datab 0 0 8 0 INPUT NODEFVAL "datab[7..0]"  
// Retrieval info: USED\_PORT: result 0 0 16 0 OUTPUT NODEFVAL "result[15..0]"  
// Retrieval info: CONNECT: @dataa 0 0 8 0 dataa 0 0 8 0  
// Retrieval info: CONNECT: @datab 0 0 8 0 datab 0 0 8 0  
// Retrieval info: CONNECT: result 0 0 16 0 @result 0 0 16 0  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10.v TRUE  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10.inc TRUE  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10.cmp TRUE  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10.bsf TRUE  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10\_inst.v TRUE  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10\_bb.v TRUE  
// Retrieval info: GEN\_FILE: TYPE\_NORMAL snehexp10\_syn.v TRUE  
// Retrieval info: LIB\_FILE: lpm



# Output 1)Cyclone II (RTL View)

# 

**2)Cyclone II (TTL View)**



**Conclusion :- In this experiment we learnt to use IP Cores made by other users and implement it on the FPGA Kit.**

**We also learnt how RTL and TTL forms for a 8bit Signed Multiplier . We simulated the IPCore on ModelSim software to view its functioning . We also implemented 8-Bit Signed Multiplier on the FPGA Kit .**