EEL6728.001F22.89357 Intro to VHDL
VHDL PROJECT

--Bindu Sagar Aluri – U96935703 (1)

Simulation results:

Wakerly:

It's a 32x32-bit multiplier producing a 64-bit output. To speed up simulation, variables were used instead of signals during the process. Both structural and behavioral models were developed: the behavioral model represents the design's functionality as a set of concurrent algorithms, while the structural model implements the functionality using basic components like AND gates, XOR gates, and majority functions. Arrays of components were constructed using the generate command.

Behavioral implementation(32-BIT):

Without regard for hardware implementation specifics, an entity can be implemented in terms of the intended design algorithm. At this level, design is fairly like the C programming language. Internal signals in Wakerly are represented by a new data type (array32*32), which is a two-dimensional array with the id std logic variable. PC stores the product component bits, whereas PCS and PCC store the sum and carry of the ripple carry adder.

To retrieve product-component bits, the first for statement conducts AND operations. The following for statement starts the boundary condition. To make it quicker, we utilized variables instead of signals. Variables are quicker since the simulator only stores their values while the process is executing. Because variable values are allocated in a logical order. Signals, on the other hand, are always valuable. When the value of a signal in a process changes, the simulator adds a future event to its event list.

Structural implementation (32-BIT):

The entity is implemented at this level using logic gates (here we used and, OR, and majority function). This level of design is comparable to expressing a design in terms of a gate-level logic diagram. Using fundamental building blocks like and, XOR, and the majority function, our structural model functions were specified. The carry outputs are then computed using a three-input "majority function" called MAJ. Most of the inputs to an n-input majority function must be 1, or two out of three in the case of a 3-input majority function, for an output of 1. (If n is even, then n/2+1 inputs must all equal 1).

Truly behavioral implementation(32-BIT):

We can simply multiply two unassigned values with a single line of a truly behavioral statement. Because the hardware reality is unimportant in test bench design, behavioral models are extensively employed.

Word Serial Multiplier (WSM):

The whole word-serial multiplication processor may be constructed by connecting the data path and the control unit once they have been generated and tested. (The product of dd and qq is aq.) The multiplicand and the shifted sum are combined to generate a partial product in a word-serial method. Each step of the multiplication process involves one of three potential operations, either the subtraction of D, no operation, or the addition of D, i.e., either (P[i]D), P[i], or (P[i] +D). All registers are activated by the positive edge of the clock signal clk, and the control unit generates op-code signals to govern their operation.

Control unit and data path unit are the two components of the WSM unit. We use ‘dd’ and ‘qq’ as the two 8-bit inputs for the multiplicand and multiplier in this (7:0). The output ‘aq’ is 16 bits (15:0). Beginning with the start signal, the multiplication is performed (st). and the ready signal indicates completion. After doing the requisite number of multiplications, the step counter returns zI, which is interpreted in the control unit.

SIMULATION COMPARISION:

A 64-bit output p is produced by a 32x32 bit combinational multiplier that is created using a behavioral program. It employs 32 shifts and adds to multiply the x and y of the 32 bits. Except for the last row, this multiplier can handle all complete adders. To shorten the delay, the carries in the last row are linked together to form a ripple carry adder. Carry outputs are calculated via a three-input majority function (MAJ). The complete adders are utilized to generate the first 32 bits of the total. The following 31 bits of the total are produced by the ripple carry adders employed in the previous step, and the last bit is produced by the ripple carry adder.

The combinational multiplier, which is designed using a structural model, gives the designer complete control over the circuit structure that is created, ASIC.The output produced by the structural and behavioral models is same, but there are differences in how they are implemented and how the reasoning is created. A behavioral multiplier can be readily created. The 64 - bit result is provided at the output port in WSM after the shifted sum of the preceding partial product and a product of the multiplicand and the ith digit of the multiplier is done numerous times. When compared to a general procedure, the shifting, adding, and restoring of bits results in significant latency.

The Table below shows the simulation results:

Case	ddin_t (Hex)	qqin_t (Hex)	Expected Output (Decimal)	Expected Output (Hex)
Case 1	x"000000f8"	x"00000302"	248 × 770 = 190960	x"0002E890"
Case 2	x"00000509"	x"000000f8"	1289 × 248 = 319672	x"0004E0E8"
Case 3	x"00000301"	x"000000f1"	769 × 241 = 185329	x"0002D3F1"
Case 4	x"00002001"	x"00000211"	8193 × 529 = 4336897	x"00423F31"

Control Unit: ***Data inputs are synchronized by the control unit.***

The multiplicand and multiplier registers are loaded into the D and registers in the initial state, which is also known as the SI state, when st=0, and the ready signal is reset. The SM state is reached at st=1; during this period, the multiplication steps are repeated n times, resulting in the generation of signal zI. Finally, we shall receive AQ as output at the port. This asserts the ready signal.

The st must be reset to zero for additional multiplications. The reset signal in this case will reset the control unit to SI. The control unit is still in the state SI at st=0, and it transitions to the state SM when st=1. When zI is affirmed, the state SM is sent to the state SF.

Data Path Unit:

The combinational blocks of A, D, and Q registers, as well as the ALU and counters, compose the data path unit. These are denoted by the registers D-multiplicand register, Q-multiplicand register, A-most significant bit we obtain from this, and Q-least significant bit we obtain from this, and by concatenating we obtain the result P=AQ. SC - step counter, F – ALU

Q-REGISTER:

Here in this q register we have the Q will n+1 which was positive edge triggered which was having 9 bits. Here the opcode is (3 down to 2). Here the QOP 0-00, 1-01,2-02,3-11. The operation of Q-register was done by 3 operations. In this OP is 0 there no operation will not perform. when OP is 1 then it will be Q= ARTHshiftright i.e. (Qn_1<= f0 & qn_1) 1 to n. when 10,11 are performed Qn_1 will be QQ & q_1 and q0_1 will be Qn_1(1 down to 0) and load the value. The QQ & 0 will be concatenate and load the value.

D-REGISTER:

Here the multiplicand register is relatively basic and only conducts two operations. We have OP as the opcode and DD acts as multiplicand and the D will acts inputs. Here the operation DOP which consists of 0 and 1 such that when the DOP was 0 the operation will be D remain D there will no change in the value. similarly, when the DOP = 1 the D=DD which load the value to the register. Here will consider the OP (6) to procedure carried out by register that was determined.

A-REGISTER:

Three operations are performed by the partial product register. in this register the AOP bits are which 00-0, 01-1, 10-2,11-1. Here it performs the arithmetic shift right operation such that F (3:0) into A which can be More understand as A <F (3), F (3 to 1). Working of the A register was when the AOP was 0 the A will be A and there is no change in the output. When AOP= 01 it will arithmetic shift right by 2 digits and for AOP= 10,11 A will be Zero and it will ger reset.

ALU:

Here the ALU performs the arithmetic and logical operations such here it uses the Opcodes as FOP Q (0: -1) such that it uses FOP as 0-00, 1-01, 2-10, 3-11. It performs the operations here was addition, subtraction, Pass. The current value of the least significant pair of multiplier digits is used as the basis for the ALU's opcodes, Fop. The working of the ALU was when the FOP was 00,11 it returns the value of F to A and condition was pass. when the value F=01, it makes A+B, when it was F= 10, it subtracts A-B.

STEP COUNTER:

Here in the step counter, we have SOP 0,1,2,3 which will be 00,01,10,11 in this when OP = 0 there will be no operation performed. When we have op=01 then it starts initializing and we will get zi =1 After 7 cycles. here the value of zi will be initialized as zi = (Sc= n-1) in which when 01 comes at first it starts with zero and goes up to n=8 which 8-1= 7 CYCLES. At sop = 10,11 it will reset to zero and again after 01 it will zi=1 after 7 cycles.

ANALYSIS & CONCLUSIONS:

From the results, we observed that WSM relies on partial products, causing the output to be generated over multiple cycles. The behavioral description has significantly lower complexity and simulation time compared to a structural logic gate-based implementation.