Final Project

ALGORITHMIC PROBLEM SOLVING USING A SYSTEM CONSISTING OF AN ALU, A REGISTER BLOCK AND A STATE MACHINE

PROJECT DEFINITION

Students will develop an algorithmic solution to one of the problems below, using a simple processor system consisting an Arithmetic Logic Unit (ALU), a register block and a state machine. Problems will be randomly assigned to each student. **Corrections/announcements will be made if a mistake is detected in the project files on Ninova. Please check Ninova announcements regularly.**

List of the problems:

1) Using the method described at [1], multiply two 4-bit positive integers and produce 8-bit positive integer as $C = A \times B$.
2) Using the method described at [1], multiply two 4-bit signed integers represented with two's complement and produce 8-bit signed integer as $C = A \times B$.
3) Using the method described at [2], calculate $C = A^B$, where A and B are 2-bit positive integers and C is a 6-bit positive integer.
4) Using one of the methods described at [3], calculate $C = \frac{A}{B}$ where A is a 8-bit positive integer and B is a 4-bit positive integer. There will be two 8-bit positive integer outputs as quotient, Q, and remainder, R.
5) Using one of the methods described at [4], calculate $C = \sqrt{A}$ where A is a 8-bit positive integer and C is a 12-bit positive fixed point number with with 4-digits decimal and 8-digits fraction.
6) Using one of the methods described at [5], calculate $C = A \; mod \; B$ where A, B and C are 8-bit positive integers.
7) Using [6], calculate geometric mean of 8-bit positive integers, A and B as $C = \lfloor M(A,B) \rfloor$ by taking the number of iterations, n as 5. C is a 8-bit positive integer.
8) Using [7], calculate $C = A!$ where A is a 3-bit positive integer and C is a 13-bit positive integer.
9) Using [10], compare $2A$ and B where A and B are 8-bit positive integers. The output should give 0 when $2A < B$, 1 when $2A = B$ and, 2 when $2A > B$.
10) Using [10], compare A and $\frac{B}{2}$ where A and B are 8-bit positive integers. The output should give 0 when $A < \frac{B}{2}$, 1 when $A = \frac{B}{2}$ and 2 when $A > \frac{B}{2}$.
11) Using the method described at [11], find the greatest common divisor of two 8-bit positive integers as $C = \gcd(A, B)$. C is a 8-bit positive integer.
12) Using [12], calculate $C = abs(A - B)$ where A and B are 8-bit signed integers represented with two's complement. C is a 8-bit positive integer.
13) Using the method given at [13], calculate $C = \left\lfloor 2A + \frac{B}{3} \right\rfloor$ where A and B are 7-bit positive integers and C is a 8-bit positive integer.
14) Calculate $C = \left\lfloor \left(\frac{A+B}{2}\right)^2 \right\rfloor$, where A is a 3-bit positive integer and B is a 2-bit positive integer. C is a 8-bit positive integer.

15) Calculate arithmetic mean of three 8-bit positive integers, A, B and C as $D = \left\lfloor \frac{A+B+C}{3} \right\rfloor$. D is a 8-bit positive integer.
16) Calculate arithmetic mean of three 8-bit signed integers, A, B and C as $D = \left\lfloor \frac{A+B+C}{3} \right\rfloor$. D is a 8-bit signed integer.
17) Calculate $C = A - 3B$, where A and B are 6-bit positive integers and C is a 8-bit sign integer.
18) Using the trial division algorithm described in [15], find the multiples of one 8-bit positive integer.
19) Using the wheel factorization algorithm described in [16], find the multiples of one 8-bit positive integer.
20) Using the method described at [17], calculate $C = A \bmod B$ where A, B and C are 8-bit positive integers.
21) Using the method described at [18], modular multiply two 7-bit positive integers and produce 8-bit positive integer as $C = A \times B \bmod 129$.
22) Using the method described at [18], calculate $C = A \times 73 \bmod B$ where A is a 7-bit and B is a 8-bit positive integer.

PROJECT STEPS

1) Design your ALU as specified in design requirements. Place it in your top module.
2) Design an algorithm that can solve your given problem. The algorithm can only use the operations provided by your ALU.
3) Express your algorithm in the form of Algorithmic State Machine (ASM); your design should be Moore type [15].
4) Design a Control Unit (CU) that realizes this ASM, via behavioral coding. Be sure to properly assign values to all of your registers.
5) Prepare a testbench and show that your design works properly.
6) You should include both your testbench code and your output waveform which shows that your algorithm works as desired. You are encouraged to use extra texts and figures on waveform to explain simulation steps. Your simulation is expected to cover all possible cases related to your algorithm.
7) Include design RTL schematics in your report.

DESIGN REQUIREMENTS

1) Your TOP module should look exactly like Figure-1. Don't use any other elements or submodules except these three given submodules.
2) An example Register Block (RB) is given in Figure-2. Just like in this schematic, you should design your ALU operations as submodules.

3) CU must only contain output registers and state registers, there should be no other registers within this design.
4) Reset and clock inputs are common for CU and RB. Reset should be designed as asynchronous active-1.
5) You may use "dont_touch" constraint with your output registers.
6) CU must be a Moore type machine.
7) CU should begin working when "start" signal is high. Then, it will keep "busy" signal high until the result is calculated.
8) State transition branches in your CU should only use the ALU signals Carry-Out (CO) and Zero (Z) as branching conditions. ex: if(CO), if(!Z).
9) When designing the ALU, you must use submodules just as given in Figure-3. If you need to use combinational assignments, use them within AND, XOR, ADD or Shift blocks.
10) RTL's must be consistent with the figures given, thus you must use the exact signal names given in figures.
11) Your ALU must perform:
    - AND, when InsSel == '00',
    - XOR, when InsSel == '01',
    - ADD, when InsSel == '10',
    - Shift, when InsSel == '11'.

    For ADD, carry-out bit should be assigned to CO output. Shift should perform a circular left-shift (MSB will move to LSB), and CO will hold the old MSB. CO should be 0 for AND and XOR operations. Z signal should be 1 only when the ALU output is zero, will be 0 otherwise.

REFERENCES

[1] https://en.wikipedia.org/wiki/Multiplication_algorithm
[2] http://en.wikipedia.org/wiki/Exponentiation_by_squaring
[3] http://en.wikipedia.org/wiki/Division_algorithm
[4] http://en.wikipedia.org/wiki/Square_root_algorithm
[5] http://stackoverflow.com/questions/2773628/better-ways-to-implement-a-modulo-operation-algorithm-question
[6] http://en.wikipedia.org/wiki/Arithmetic–geometric_mean
[7] http://en.wikipedia.org/wiki/Factorial
[9] http://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html
[10] http://en.wikipedia.org/wiki/Digital_comparator
[11] http://en.wikipedia.org/wiki/Euclidean_algorithm
[12] https://en.wikipedia.org/wiki/Absolute_value
[13] http://courses.cs.vt.edu/~cs1104/BuildingBlocks/divide.030.html
[14] Brown&Vransic, Fundamentals of Digital Logic with Verilog Design, McGrawHill, 2002, Section 8.10.

[15] https://en.wikipedia.org/wiki/Trial_division
[16] https://en.wikipedia.org/wiki/Wheel_factorization
[17] https://brilliant.org/wiki/modular-arithmetic/
[18] https://en.wikipedia.org/wiki/Modular_arithmetic

Ögrenci No	Ad Soyad	Question
40160054	Yavuz Sultan Çakırca	5
40160243	Büşra Yassıbaş	17
40160256	Salih Daysal	10
40160262	Furkan Akçil	20
40170011	Abdullah Sak	2
40170038	Enes Mutta	16
40170040	Elif Dinç	9
40170051	Gülce Baysal	18
40170077	Mustafa Ensar Işkın	6
40170089	Furkan Emre Işık	11
40170099	Can Bozacı	15
40170213	Behiç Erdem	1
40170217	Seçkin Gezer	19
40170255	Melih Furkan Şen	4
40170742	Cihad Sefa Dağdeviren	12
40170931	Erdem Ibraimi	8
40180021	Muhammet Yusuf Alkuş	14
40180122	Muhammed Furkan Ertural	3
40180710	Enes Dilbilmez	13
40180730	Asya Turhal	21
40200789	Canberk Köse	7