ECSE 323 – Final Project
Base-100 Financial Calculator

Names :	Christian Gallai (260218797)
	Stefanos Koskinas (260211145)
Group Number:	Group 30
Date :	Tuesday December 2nd, 2008
Lab Periods:	T: 1:00-3:00PM ; R: 4:00-6:00PM

SYSTEM FEATURES

This financial calculator has been designed to perform three main operations. All operations relate the following four quantities:

- PV: present value of investment
- FV: future value of investment
- i: interest rate
- N: years of investment

These quantities are interrelated by the following equality: $FV = PV\left(1 + i/100\right)^N$

The three operations performed by this calculator are the following.

1. $FV = PV\left(1 + i/100\right)^N$
   Determines the future value, given the present value, interest rate and years of investment.

2. $i = 100\left(\sqrt[N]{M} - 1\right)$
   Determines the interest rate, given the ratio of future value over present value $M = \dfrac{FV}{PV}$, and the years of investment.

3. $N = \log_{(1+i/100)} 2$
   Determines the years of investment, given the interest rate and a ratio of future value over present value $M = \dfrac{FV}{PV} = 2$.

SYSTEM BLOCK DIAGRAM

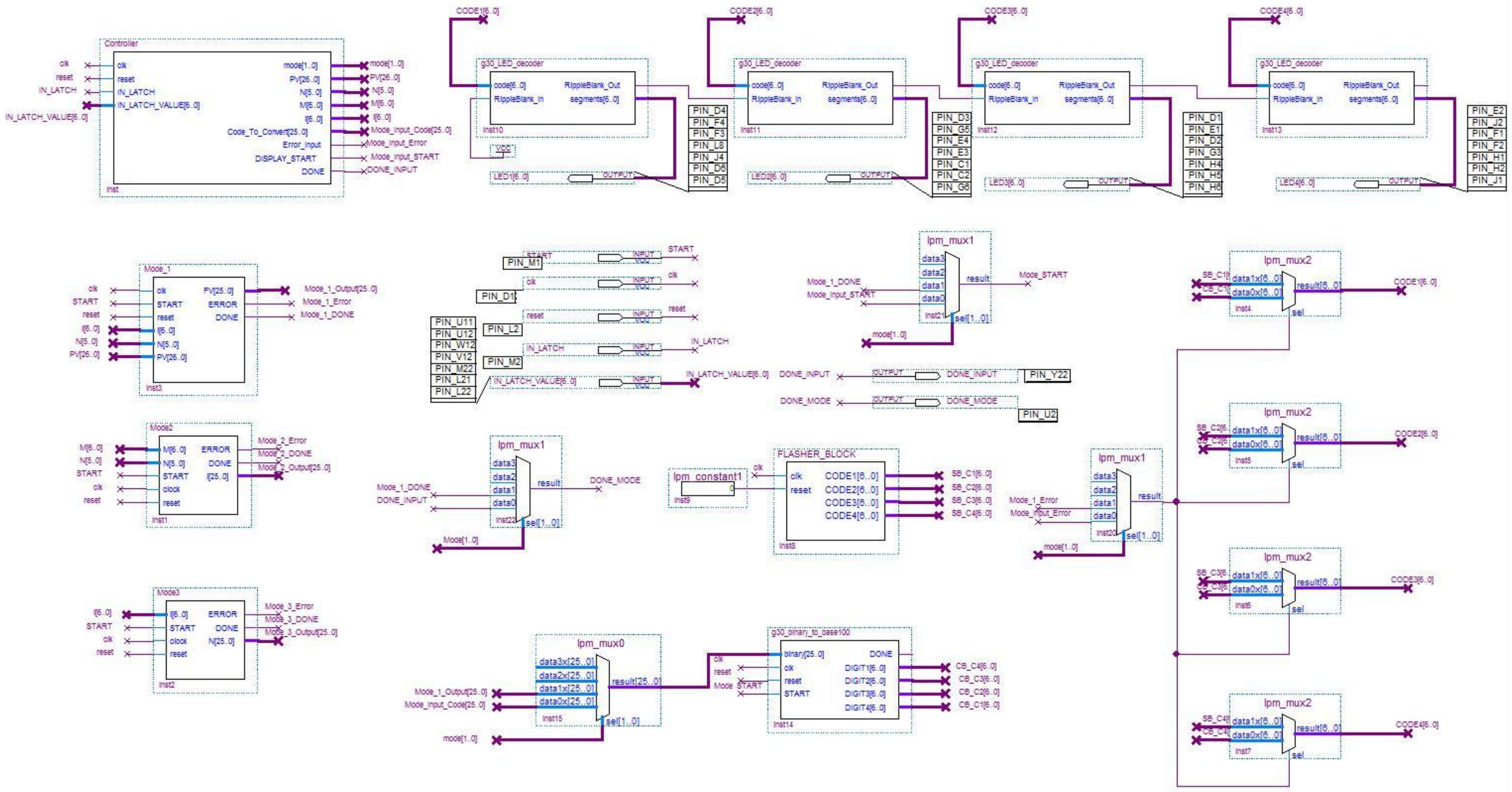


Figure 1: System Block Diagram

SYSTEM OPERATION

Controller System:

The controller system was synthesized in VHDL using a FSM approach. The controller takes as an input a 'clock', 'reset', 'In_Latch' signal to be mapped to a toggle switch for latching data, and an 'In_Latch_Value[6..0]' signal to store the latched values. The output of the controller consists of outputs for 'mode[1..0]', PV, M, N, and i, which are the values required to perform the various mode calculations of the calculator. The controller also outputs an error signal called 'Error_Input' which is asserted whenever an error has occurred in the inputting process. Also, a 'Code_To_Convert[25..0]' signal is outputted to be used to output the values being inputted into the system after they have been latched. The final output is the 'DONE' signal which is used to be outputted via an LED on the Altera board to indicate to the user that data entry has been successful and is complete. The controller block can be found in the top left corner of the system block diagram in **Figure 1**.

The Controller FSM contains 29 states which dictate it's control flow throughout the operation of the calculator. The general process for input begins with mode selection, followed by information input based on the mode selected. If all information inputted is valid and within acceptable range, the DONE state is eventually reached and all required input data is fully latched into the system to begin a mode. If an error is made, the Controller FSM is sent to an error state where is outputs a flashing error message. This state is left only when reset is asserted and the FSM restarts the inputting process. The Controller also outputs each of the input values after they have been fully entered so that they can be displayed to the user.

Output Converter System:

The output converter system consists of the multiplexor and 'g30_binary_to_base100' block located in the bottom center of the system block diagram. The multiplexor is controlled the 'mode[1..0]' signal outputted from the controller block. This allows the converter block to convert the output from the correct mode which has performed the desired calculation chosen by the user. The data to be converted is then passed through the converter to be later outputed to the user.

Error Detection System:

The Error detection system consists of the multiplexors in the bottom right-hand corner of the system block diagram. The first leftmost multiplexor brings together the error signals obtained from each of the mode calculation blocks as well as the controller system. The multiplexor is controlled once again by the 'mode[1..0]' signal. This allows the circuit to check if an error has occurred as the system moves from the input mode to a specific mode calculation. This error signal is then passed as a control signal to multiplexors controlling each of the four output codes to be later passed as output through the LED decoders. If an error has occurred, the code multiplexors will pass the signals generated from the 'Flasher_Block' located in the center of the system diagram which outputs a flashing error message. If no error occurs, then the output code obtained from the output converter system will be passed through to be later outputted.

Display Output System:

The display output system consists of the four 'g30_LED_Decoder' blocks located at the top of the system block diagram. These blocks take as input the codes outputted from the error detection system. The outputs of these blocks are then mapped to the four 7-Segment LEDs to output the result to the user.

Mode Blocks:

The three individual mode calculation blocks were each synthesized in VHDL. They perform the calculations set by the user, taking as input the data outputted from the controller system. The beginning of the calculation is controlled by the 'START' signal which is connected to a toggle switch on the Altera board. Once the user is ready to perform the desired calculation after all required data has been latched, he or she may assert the START signal to begin the mode calculation whose input is later taken to be displayed with the display output system.

USER INTERFACE

The Base-100 Calculator uses an array of switches and LEDs on the Alters DE1 board in order to input and display information. Let's begin with the input mechanisms. **Figure 2** below shows a snapshot of the Altera board, indicating the components used for input.

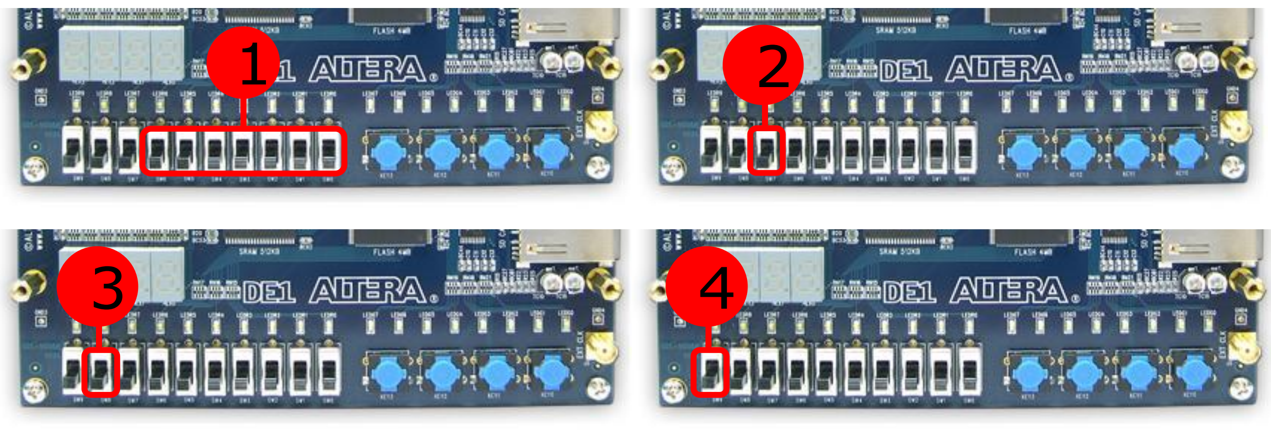


Figure 2: Input Components Used for Base-100 Financial Calculator

A description of these components is shown below in **Table 1**.

Table 1: Input Component Descriptions

ID Number	COMPONENT	DESCRIPTION
1	Input Switches	These switches are used to input the desired data seven bits at a time. The leftmost switch represent the MSB of the input and the input process latches inputs from Least to Most significant for inputs requiring more than 7 bits (i.e. input [7..0] followed by input[14..8], etc..).
2	Input Latch	The 'Input Latch' switch is used to store the values set on the Input Switches into the system. Whenever this switch is toggled (i.e. set to '1' then set to '0') the controller saves the value currently set on the Input Switches and the user may then proceed to arranging the next input value.
3	Start Switch	The Start switch is used to begin the desired mode calculation once the user has entered all inputs for the desired mode. Once the ' Input Latched ' LED is on (see output component descriptions in **Table 2**), the user may set this switch to '1' to begin calculation.
4	Reset Switch	The 'Reset' switch is used to reset the system (i.e. in the case of an error or after an undesired input has been latched)

Similarly, the calculator uses an array of LEDs to display the outputs and state of the calculator, These output components are indicated in **Figure 3** below and are then described in **Table 2**.

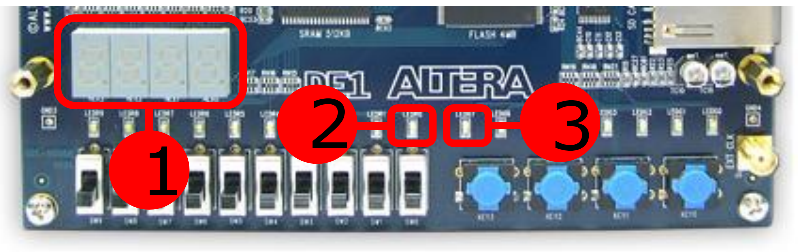


Figure 3: Output Components Used for Base-100 Financial Calculator

Table 2: Output Component Descriptions

ID NUMBER	COMPONENT	DESCRIPTION
1	LED Display	The calculation outputs are displayed on the four 7-segment LED displays shown. The outputs on these displays are the base-100 representation of the calculated values. These LEDs are also used to display the inputted values once latched into the system.
2	Input Latched	The 'Input Latched' LED turns on whenever all the inputs have been latched for the specified mode of operation.
3	Calculation Complete	The 'Calculation Complete' LED turns on whenever the desired mode calculation has completed and it's result has been displayed on the LED display

Described below are the inputting processes for calculations in each of the three modes of the Base-100 Financial Calculator:

Input Process For Mode 1:

1) Set input switches to "0000001" to input mode and latch
2) Input i on input switches and latch
3) Input N on input switches and latch
4) Input PV[6..0] on input switches and latch
5) Input PV[13..7] on input switches and latch
6) Input PV[20..14] on input switches and latch
7) Input PV[26..21] on input switches and latch
8) Set Start Switch to 1 to display FV

Input Process For Mode 2:

1) Set input switches to "0000010" to input mode and latch
2) Input N on input switches and latch
3) Input M on input switches and latch
4) Set Start Switch to 1 to display i

Input Process For Mode 3:

1) Set input switches to "0000011" to input mode and latch
2) Input i on input switches and latch
3) Set Start Switch to 1 to display N

TESTING

The process of testing follows the procedure:

- Choose selective inputs
- Calculate and predict expected output
- Perform simulation
- Compare simulation output to predicted output

We begin our testing for ***Mode1***. **Table 3** below shows 8 different groups of input values tested. Cases 1, 3, 6 and 7 are chosen as representative groups shown in Figures 4,5,6, and 7 respectively.

Table 3: Testing I/O for Mode 1

CASE	I	N	PV	FV	PREDICTED FV	ERROR %
1.	5	32	193939.39	795454.52	924109.84	14
2.	5	32	1000.00	4101.56	4764.94	14
3.	2	10	1000.00	1156.25	1218.99	5.1
4.	2	10	357.18	412.98	435.40	5.1
5.	4	15	125.00	212.89	225.11	5.4
6.	4	2	125.00	134.76	135.20	0.33
7.	15	4	125.00	214.84	218.63	1.7
8.	15	4	193939.39	333333.32	339201.21	1.7

We have used test sets for different situations, such as small i – large N, large i – small N, similar i and N, and a number of scales for PV. By analyzing the data from the table, we can deduce the following conclusions:

- Simulated outputs are close enough to the predicted outputs to deduce that the circuit is functioning properly. This conclusion is not necessarily true, but given a large sample of data, the probability of it being true is very large.
- There is always a relative output error, as the conversion between bases is not perfectly accurate.
- The simulated FV is always smaller than the predicted FV. This most likely occurs due to rounding down.
- As PV values get smaller, the relative output error reduces.
- When $i << N$, the error is large, while for $i >> N$, the error is small.

- N is the input variable that mostly influences the output error.

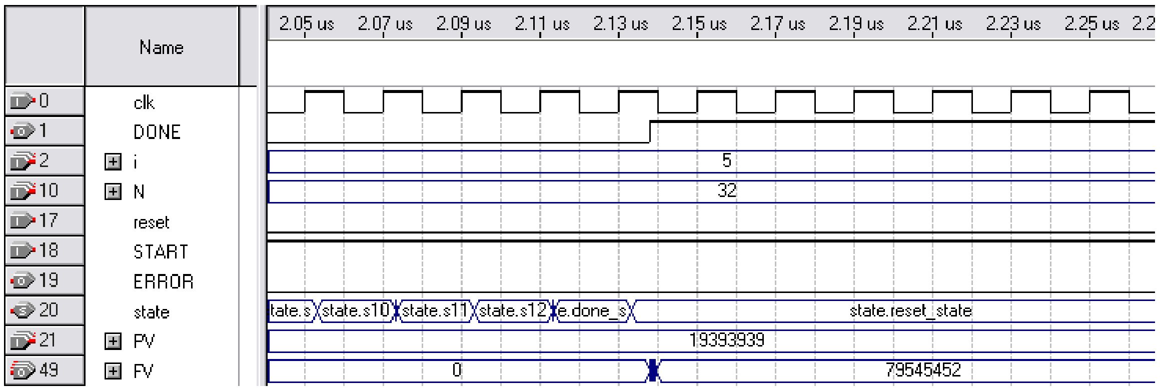


Figure 4: Mode1 simulation for $i = 5, N = 32, PV = 193939.39 \Rightarrow FV = 795454.52$.

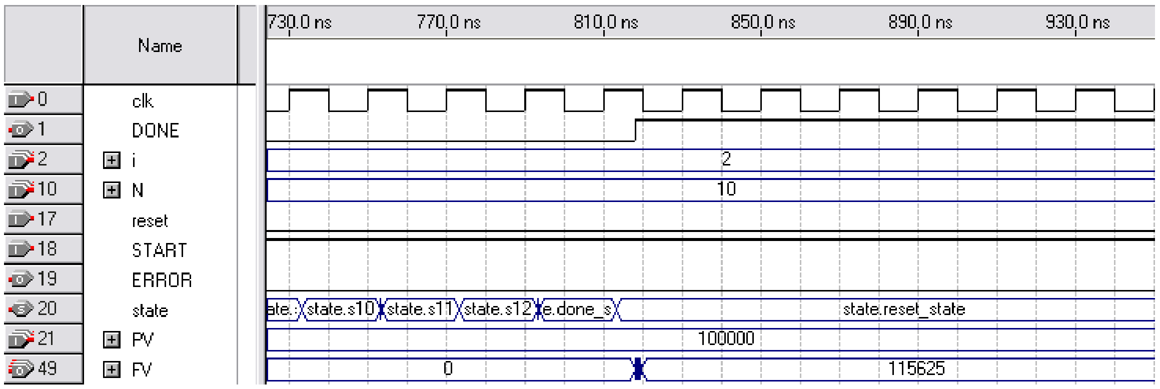


Figure 5: Mode1 simulation for $i = 2, N = 10, PV = 1000.00 \Rightarrow FV = 1156.25$.

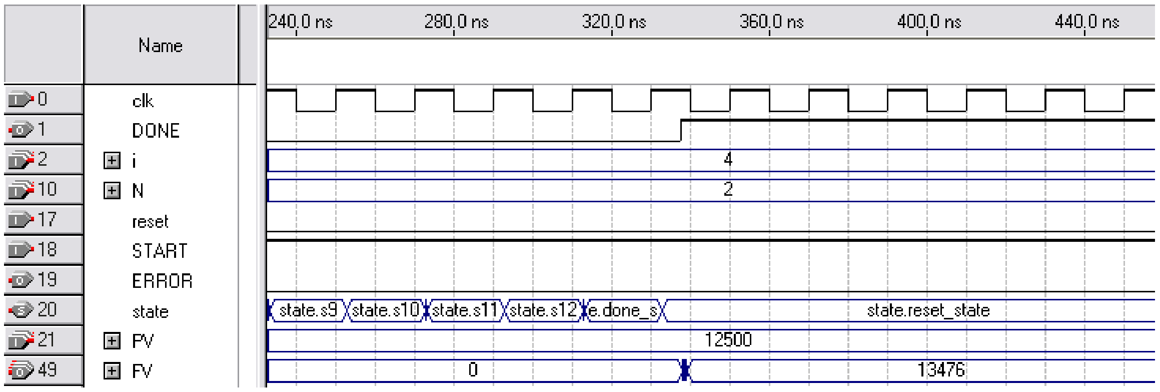


Figure 6: Mode1 simulation for $i = 4, N = 2, PV = 125.00 \Rightarrow FV = 134.76$.

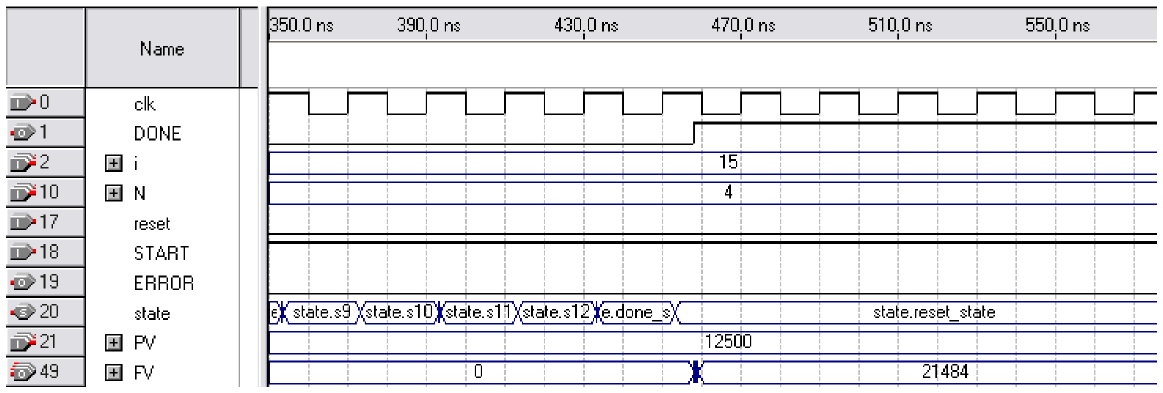


Figure 7: Mode1 simulation for $i = 15, N = 4, PV = 125.00 \Rightarrow FV = 214.84$.

We proceed to our testing for ***Mode2***. **Table 4** below shows 10 different groups of input values tested. Cases 2, 3, 5, 7 and 10 are chosen as representative groups shown in Figures 8,9,10,11 and 12 respectively.

Table 4: Testing I/O for Mode 2

CASE	M	N	I	PREDICTED I
1.	0	24	0 (Error)	Error
2.	1	24	0 (Error)	0 (Error)
3.	2	24	2	3
4.	2	63	0	1
5.	15	15	19	20
6.	15	4	96	97
7.	18	4	0 (Error)	> 99 (Error)
8.	2	2	41	41
9.	2	1	0	> 99 (Error)
10.	2	0	0 (Error)	Error

We have used test sets for different situations, varying both M and N. We have also included the range limits. By analyzing the data from the table, we can deduce the following conclusions:

- Simulated outputs are the same as the predicted outputs for most cases.

- For the cases that they are not the same, the simulated i is always lower by 1, and the difference occurs because the simulation rounds numbers down.
- As N increases or M decreases, i goes to zero.
- As M increases or N decreases, i exceeds the limit of 99 and gives an error.
- When M or N are zero, i is set to zero and an error is displayed.
- Just above the limits, the design recognizes the error, but only as far as the range of numbers that can be represented by the bit lengths. So, for example, $M = 18, N = 4 \Rightarrow i = 106$ will give an error, since i has 7 bits and thus a max value of 127. However, $M = 30, N = 4 \Rightarrow i = 134$ will not give an error, because 134 cannot be represented. It will be truncated, leaving an entirely different number. This is a limitation of this design, caused by the inputs and outputs bit lengths.

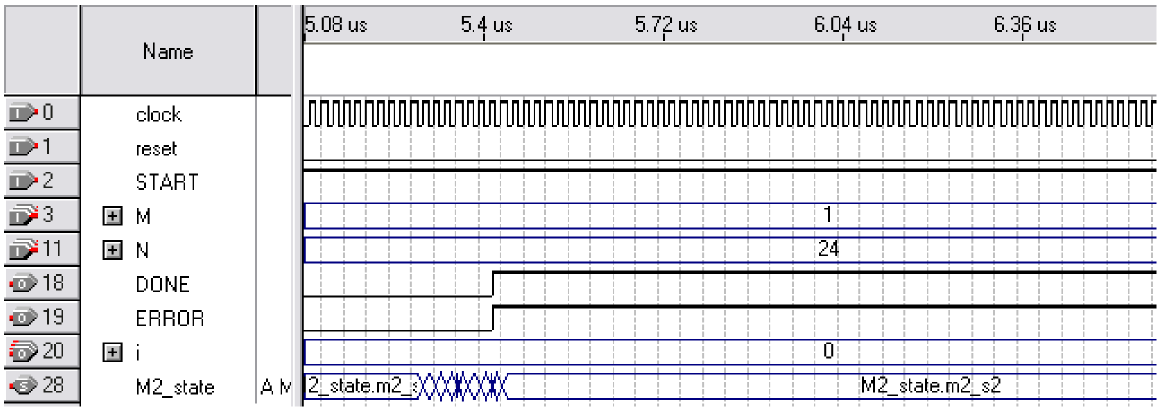


Figure 8: Mode2 simulation for $M = 1, N = 24 \Rightarrow i = 0(Error)$.

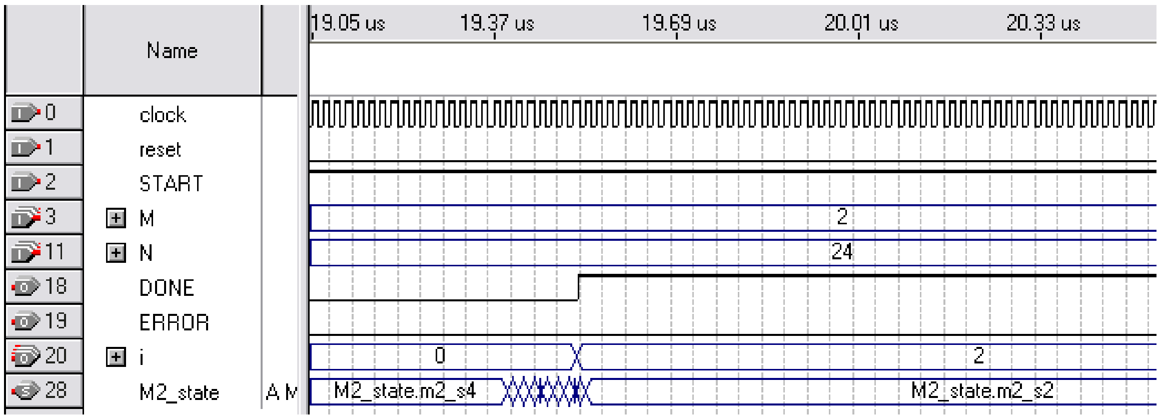


Figure 9: Mode2 simulation for $M = 2, N = 24 \Rightarrow i = 2$.

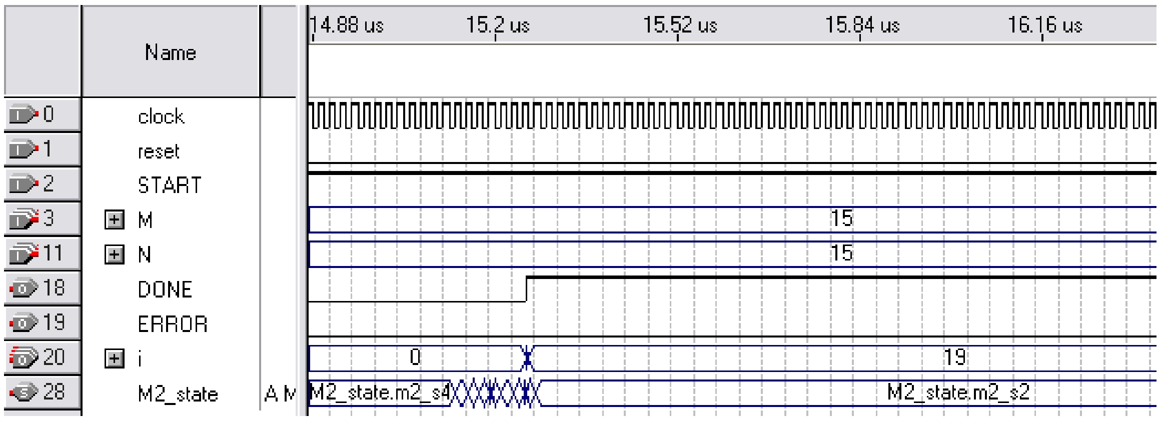


Figure 10: Mode2 simulation for $M = 15, N = 15 \Rightarrow i = 19$.

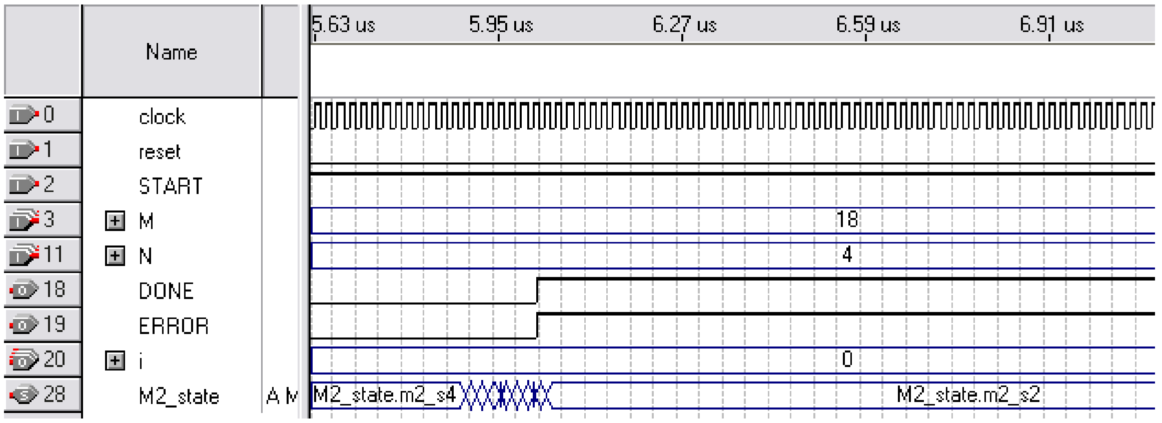


Figure 11: Mode2 simulation for $M = 18, N = 4 \Rightarrow i = 0(Error)$.

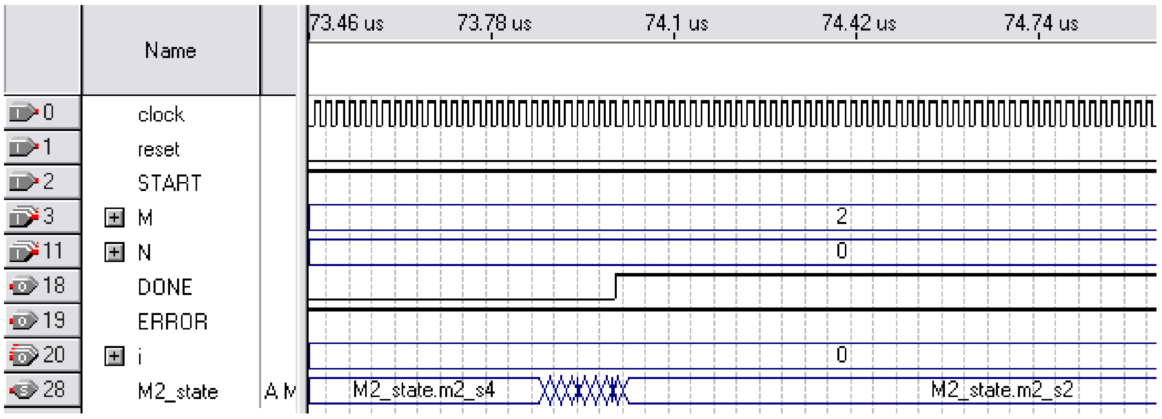


Figure 12: Mode2 simulation for $M = 2, N = 0 \Rightarrow i = 0(Error)$.

Finally, we test the functionality of ***Mode3***. **Table 5** below shows 8 different groups of input values tested. Cases 2, 4, 6 and 8 are chosen as representative groups shown in Figures 13,14,15, and 16 respectively.

Table 5: Testing I/O for Mode 3

CASE	I	N	PREDICTED N
1.	0	0 (Error)	Error
2.	1	69	70
3.	2	35	35
4.	5	14	14
5.	17	4	4
6.	30	2	3
7.	58	1	2
8.	99	1	1
9.	> 100	0 (Error)	Error

For this mode, the testing involved varying i. By analyzing the data from the table, we can deduce the following conclusions:

- Simulated outputs are the same as the predicted outputs for most cases.
- For the cases that they are not the same, N is always lower by 1, and the difference occurs because the simulation rounds numbers down.
- The simulation process involves using the log2 function, which in turn uses a lookup table (LUT). Therefore, the error is reduced by minimizing operations.

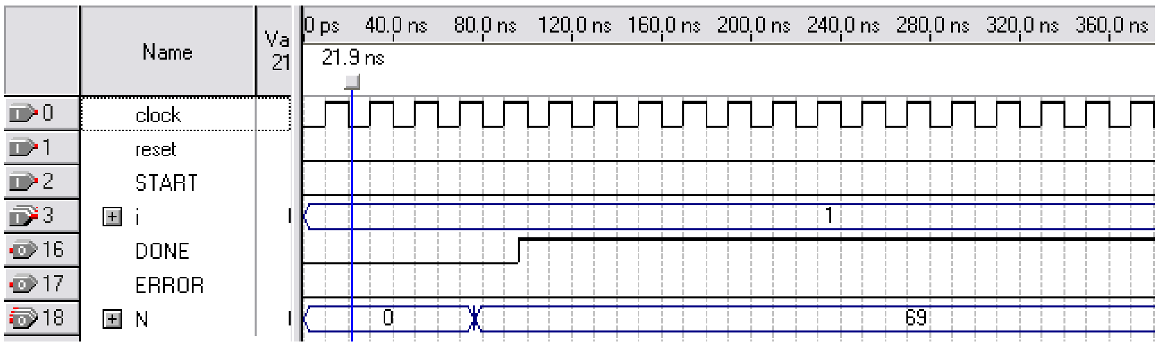


Figure 13: Mode3 simulation for $i = 1 \Rightarrow N = 69$.

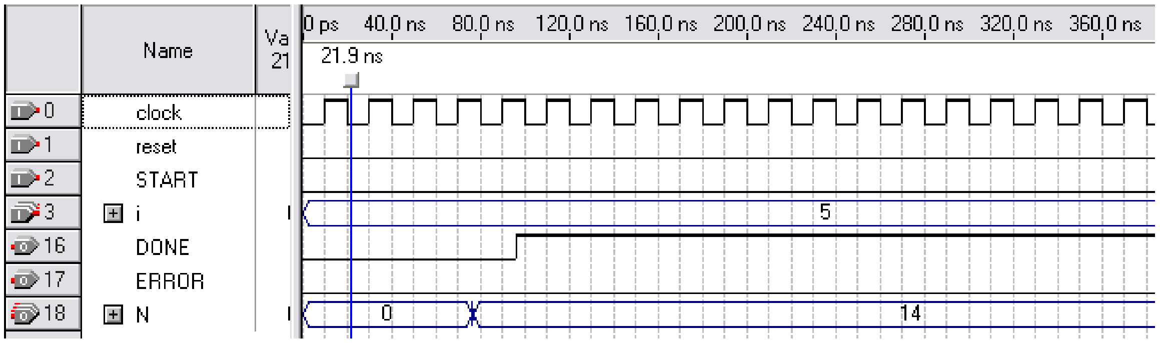


Figure 14: Mode3 simulation for $i = 5 \Rightarrow N = 14$.

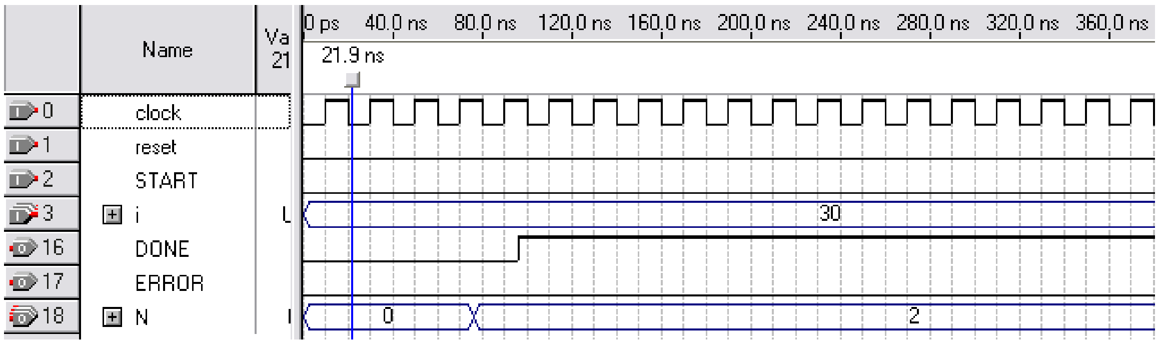


Figure 15: Mode3 simulation for $i = 30 \Rightarrow N = 2$.

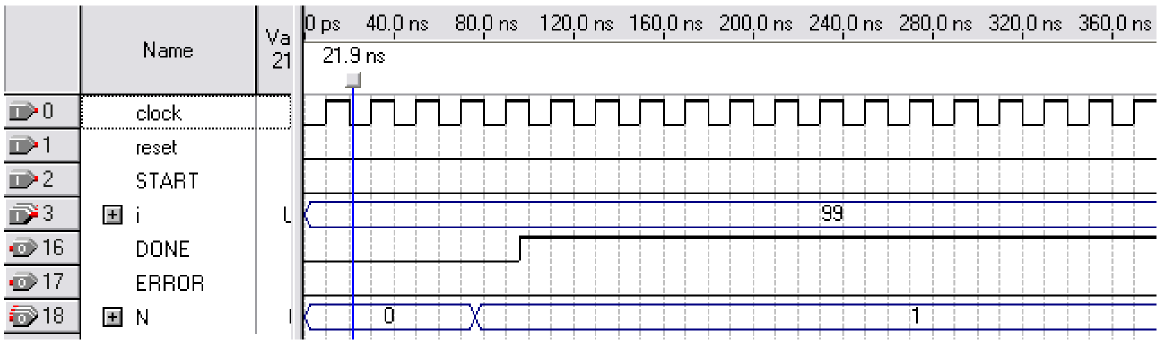
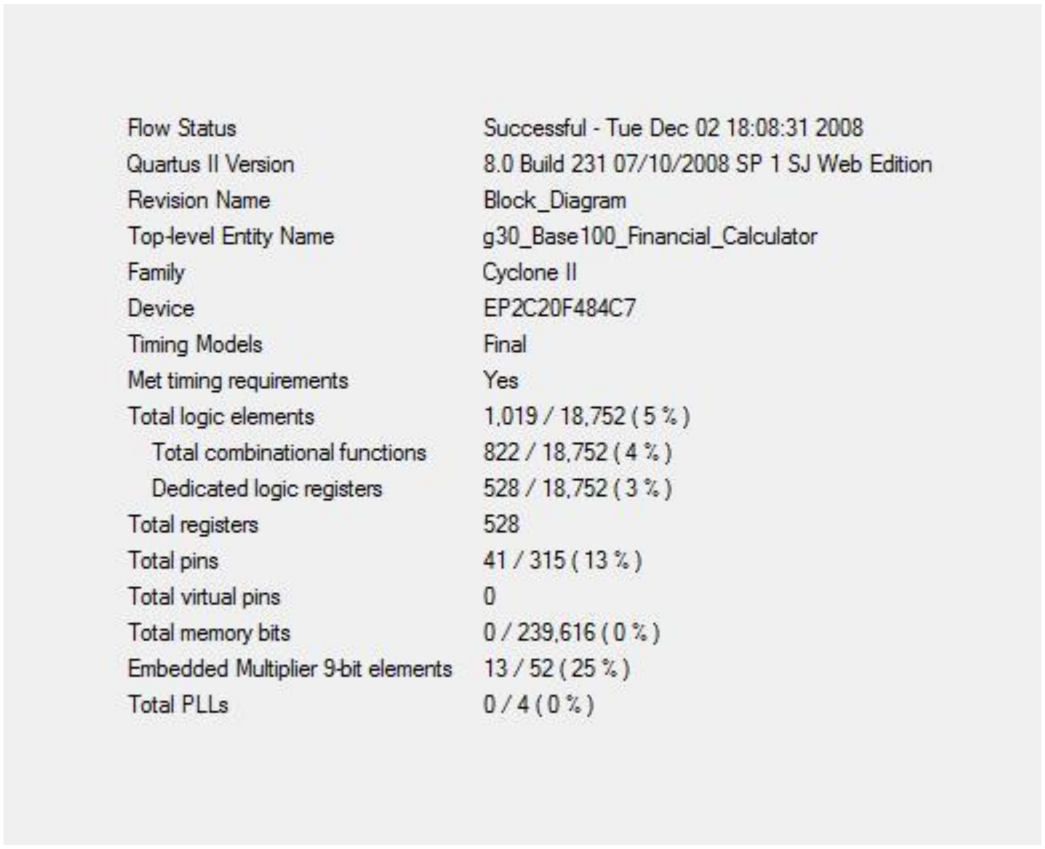


Figure 16: Mode3 simulation for $i = 99 \Rightarrow N = 1$.

FPGA RESOURCE UTILIZATION

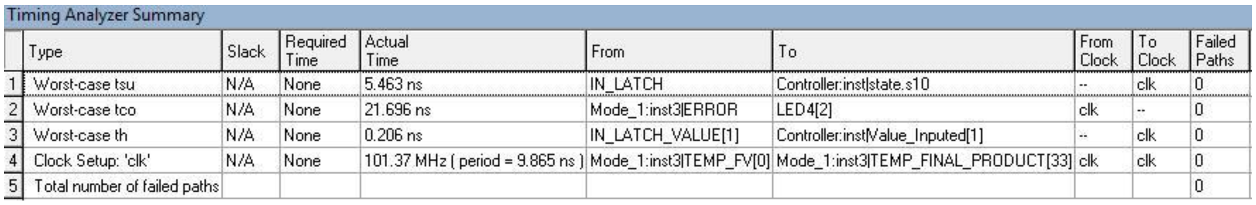
The Flow Summary of the compiled 'g30_Base100_Financial_Calculator' circuit is shown below in **Figure 17**.



Flow Status	Successful - Tue Dec 02 18:08:31 2008
Quartus II Version	8.0 Build 231 07/10/2008 SP 1 SJ Web Edition
Revision Name	Block_Diagram
Top-level Entity Name	g30_Base100_Financial_Calculator
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1,019 / 18,752 (5 %)
Total combinational functions	822 / 18,752 (4 %)
Dedicated logic registers	528 / 18,752 (3 %)
Total registers	528
Total pins	41 / 315 (13 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	13 / 52 (25 %)
Total PLLs	0 / 4 (0 %)

Figure 17: Flow Summary for 'g30_Base100_Financial_Calculator' circuit

Looking at this summary, a total of 1,019 logic elements were used in the synthesis of this circuit. Continuing on, we see the timing analysis for the circuit as shown below in **Figure 18**.



Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	5.463 ns	IN_LATCH	Controller:inst|state.s10	--	clk	0
2	Worst-case tco	N/A	None	21.696 ns	Mode_1:inst3|ERROR	LED4[2]	clk	--	0
3	Worst-case th	N/A	None	0.206 ns	IN_LATCH_VALUE[1]	Controller:inst|Value_Inputed[1]	--	clk	0
4	Clock Setup: 'clk'	N/A	None	101.37 MHz (period = 9.865 ns)	Mode_1:inst3|TEMP_FV[0]	Mode_1:inst3|TEMP_FINAL_PRODUCT[33]	clk	clk	0
5	Total number of failed paths								0

Figure 18: Timing Analysis for the 'g30_Base100_Financial_Calculator' circuit

Looking at the timing analysis for this circuit, the worst case tco is found to be 21.696 [ns].

CONCLUSION

The design process was long and complex. We had to pay a lot of attention to details, as one small change could alter everything. Additionally, syntax errors are more or less easy to fix because Quartus recognizes them. However, logic errors are much harder to identify.

Below are some significant issues that occurred during the process, as well as potential improvements to the design:

- A common and very important problem was the timing issue. While a circuit might work for a functional simulation, it could give different results for a timing simulation. The reason for this is because the functional simulation does not take time into account and performs operations in a logical order. In timing, however, each operation has to wait for the previous one to be registered. Most importantly, in transitions between states within a process, the output of an operation at one state will only be registered at the next state (i.e. clock cycle). To avoid timing issues, sufficient time should be allowed to pass at each stage of the process. Operations using the same or interdependent variables should never occur simultaneously; extra time should be allowed for certainty.

- In some cases, states within a process were synthesized away. We were never able to perfectly understand the reason behind this, but the solution was to specify the signal for each state directly, e.g. “0001” for state 1, instead of defining the signals of a state type. Also, all variables should be reset initially, so that their output as the end of the process is not synthesized away.

- A third problem was the accuracy of calculations. For example, in Mode1, the resulting FV was never perfectly accurate. The relative error incurred is unavoidable, because when converting numbers between bases, accurate representations are not always possible. In this case, the goal of the designer is to minimize the error to a point where it will not be of importance for that case. To reduce the error, the design could be modified to incorporate a greater number of bits to represent variables. The greater the number of bits, the smaller the error.

- The controller is also a very sensitive component. By implementing all modes together, the designer has to be careful so that each component does not interfere with the operation of others.

- A general realization we get from working on this design is that the design should always be as simple as possible. When there are two options to perform an operation, we should choose the one that is simpler, and thus, faster, provided that there is no other major disadvantage, such as error, limited range etc. For example, multiplication can be performed by the multiplication operator and by using appropriate signals and bit lengths, or by a library multiplier. The first one should be preferred because it is simpler and faster, and therefore, less likely to cause timing issues.