DSD LAB 4

GROUP 07

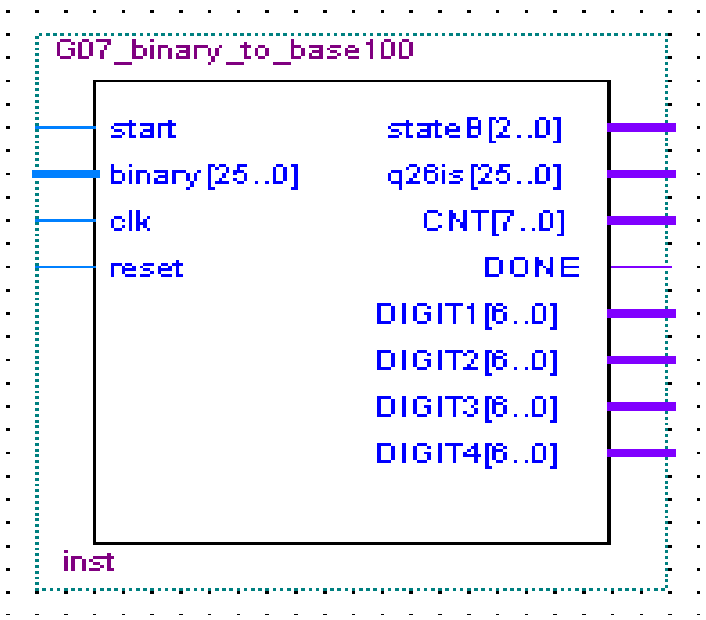
PAUL DOUMET 260 226 189

SIMON FOUCHER 260 223 197

PART 1: CONVERTER

Goal: Input a number in the form of 26 binary bits and display it on 4 x base 100 LED segments.

Primary unit:
G07_binary_to_base100 (all relevant files in /Part1/ G07_binary_to_base100/)



A) Inputs:

- start: Start button, used to initiate start sequence. The system will start when start goes to 0, then to 1
- binary[25..0]: binary number to be loaded into the system and displayed
- clk: System's clock at 50MHz
- reset: resets the system

B) Outputs:
(non vital; used for development)

- stateB[2..0]: 3 bit binary representation of the current state (to ensure that the state machine transitioned between states; we implemented 5 states)
- q26is[25..0]: content of the 26 bit register (to make sure that the binary number got loaded at the right time and that the shifting occurred properly)

- CNT[7..0]: outputs the count of the counter (since we used a full 7 bits, to make sure that it reset to 25 and stops counting at 0)

(functional outputs)

-DONE: will signal high when the system is at rest (done) and low when we are computing

-DIGIT: binary value of a number to be converted to its segment representation

C) Internal components of G07_binary_to_base100

I-adder14 (files in /Part1/adder14/)

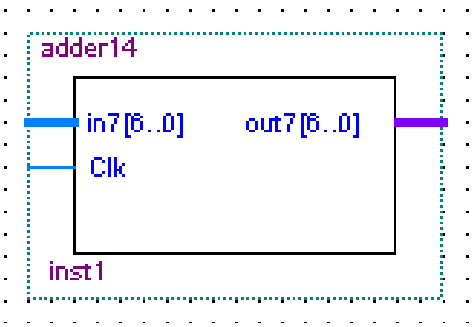
II- shift_register7 (files in /Part1/shift7/)

III- shift_register26 (files in /Part1/shift26/)

IV-counter (files in /Part1/counter/)

V- Finite State Machine (embedded in /Part1/G07_binary_to_base100/G07_binary_to_base100.vhd)

I-adder14

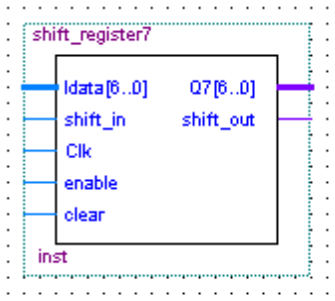


I/O ports:

- in7: 7 bit binary number input
- out7: 7 bit binary number output
- Clk: input for a clock. (in our implementation, internal connections are commented out on the VHDL code)

Function: This component asynchronously checks if in7 is greater than 49. If so, out7= in7+14. Otherwise, out7 = in7

II-shift_register7



I/O ports:

Inputs:

-ldata: a 7 bit vector which loads data presented by the adder

- enable: enable line

- shift_in: Bit to be shifted in (the rightmost register receives it from the 26 bit register; then every other register receives its shift_in from the register on its right.

- Clk: System clock at 60MHz

- clear : When high, forces every bit of the register to 0; regardless of clock events and inputs

Outputs:

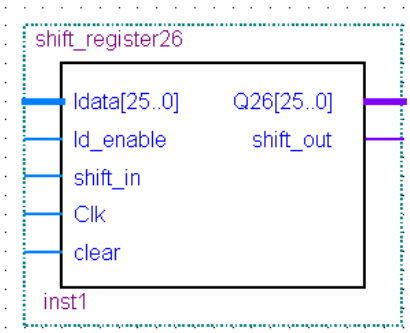
- Q7: 7 bit vector that outputs the content of the register

- shift_out: bit to be shifter out of the register

Function:

Since the adder constantly looks at the content of the register and add 14 if needed, the shift register is clocked to avoid conflicts with this function. Through the use of intermediate signals, the adder (on the rising edge of the clock) loads the data from the adder (which is effectively the same as adding 14 to it's content), then shifts the data.

III-shift_register26



I/O ports

Inputs:

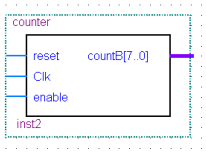
- ldata: a 26 bit wide bus that imports values for the register's bits
- ld_enable: When high, the register takes the values of ldata on the next rising clock edge
- shift_in: Bit to be shifted serially into the register (connected to GND)
- Clk: System's 50MHz clock
- clear: When high, all bits inside the register are asynchronously reset to 0

Outputs:

- Q26: The 26 bits contained in the register. Not used for anything else than troubleshooting.
- shift_out: Serial bits that gets shifted out of the register at every shifting clock pulse

Function: If clear = 1; all content of register is reset to 0 regardless of anything else. Otherwise, on the rising edge of the clock, the register either loads the data presented in ldata if ldata = 1; otherwise, shifts its content left.

IV-counter



I/O ports:

Inputs:

- reset : if =1; counter is reset to 25, regardless of any other inputs.
- Clk: 50Mhz system's clock.
- enable: When = 1; the counter counts downwards
- countB: 8 bit vector, contains count value in binary.

Function:

If reset = 1; the counter is reset to 25. Otherwise, on the rising edge of the clock, if enable =1, the counter counts down. When it reaches 0, it resets to 32 (but the FSM resets it to 25 before enabling it)

V- Finite State Machine:

Implemented as a controller that managed the control signals that are sent to the system during operations and makes decisions based on feedback received from the system.
Implemented using 5 states (we used a different implementation than the one proposed in the lab description). Each state has a string name (S1,S2,...,S5) and a name in binary (signal stateB) used so we were able to map the current stated of the system on a Waveform during simulation to make sure that state transitions went smoothly and that events happened in their respective states. In future developments, we could use a 'single-hot-state' implementation and recycle the state value signal as a control signal to be sent to the system.

When reset is high, the state is set to S0 and DONE is set to 1. Otherwise, state transition happens on the rising edge of the clock based on the feedback received from the system.

S0:

- Wait in S0 until start goes low;
- Then transition to to S1.

S1:

- Wait in S1 until start goes high;
- Then set reseting_reg = TRUE. This signal is connected to the reset line of every registers as well as the counter's reset line. When high, every registers' every bits are asynchronously reset to 0, and the counter's value is reset to 25 (Since the counter has an 8 bit signal bus, it normally loops back to 31 when it reaches 0).
- DONE is reset to 0 to indicate that a new task has begun. (because of our extra signal lines like state value and counter value, we never used 'DONE' since we could observe in detail the system;s operations, but we implemented anyways at the very end because it was required by the lab's description. Because of the speed at which the system operates (it takes roughly 30 cycles of the 50MHz clock to complete a display operation, even though the 'DONE' got wired to a LED on the board, it looks like it is always on).
- Then Transition to the next state.

S2:

- Wait in S2 until all registers are reset to 0. To implement this, we OR every bit of every register into a signal called 'done_reseting'. The only time done_reseting = 0 is when all the registers have been reset to 0. This signal also implies that the counter's value has been reset to 25, but we consider the content of the register to be suffiscien information to indicate that the task is complete (Waveform study and successful implementation proved this right)
- Stop reseting by making reseting_reg = 0
- Load the binary number to be displayed into the 26 bit register by setting load_binary = 1. (this signal is connected to the load line of the 26 bit register)
- Then transition to next state.

S4:

- We don't have to wait until the binary is done loaded before proceeding with this state. Since the binary number gets loaded asynchronously as soon as we leave S3 and that this task takes at most 12nS, and that it takes at least 1 clock cycle to transition to S4, we know that as soon as we have reached S4, the previous task has been completed.

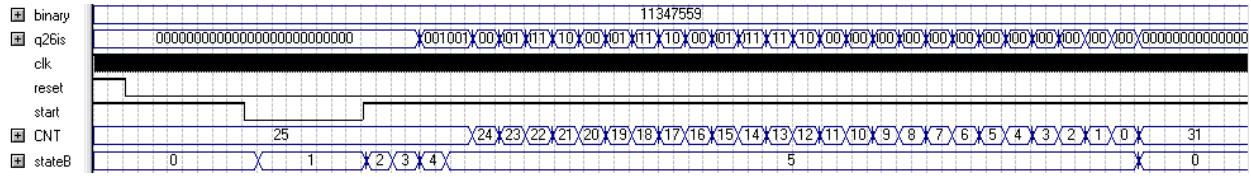
- Stop loading binary by reseting load_binary = 0
- Start the counter by setting counting26 = 1 (connected to the counter's enable line)
- Start shifting the binary data (shift enable is also connected to count26)
- Then transition to the next state.

S5

- Wait in S5 until done counting. The counter will count at every clock pulse, so when it reached 0, we shifted the whole 26 bit number through the system and the task is done. To detect this condition, all the bits of the counter's count are Ored into a signal line called done_counting. When done_counting = '0', the counter has reached 0.
- When this happens, stop counting by reseting counting26 = 0
- Set DONE = 1 to signal that the task is done
- Go back to S0 and wait for the next task

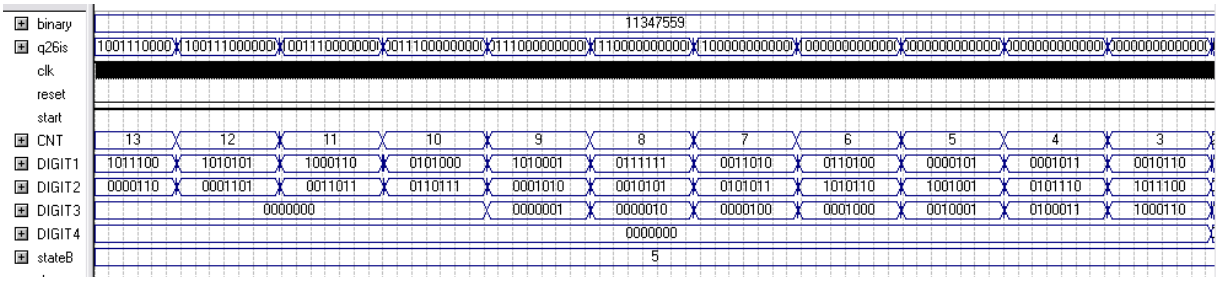
TROUBLESHOOTING

State transition and counter: Using wave files, we were able to observe adequate results:



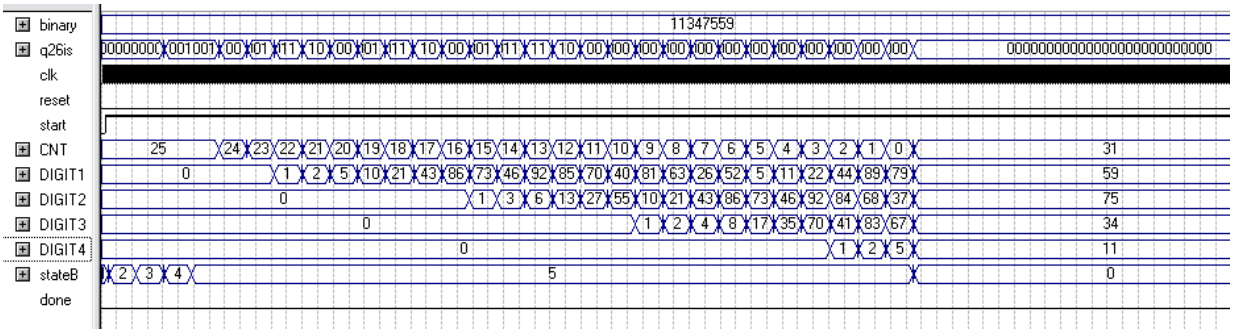
CNT: Counts from 25 to 0 (resets to 31, but we initialize it to 25 before starting to count)
stateB: Binary number representing current state. We can observe state transitions from 0 to 5, then back to 0, and follow the work in every state.

Shifting and adding



q26is: Displays the content of the 26 bit register. We can follow bits from q26 to all the digits as they are shifted left.

Final product:



We can observe the transition in count from 25 to 0 in state 5. Once the counter reaches 0 (signaling that the shifting operations are done), the state goes to 0 and the output on the DIGITs is stabilise to the adequate display. (here, binary:11347559 is displayed as D4: 11 D3: 34 D2: 75 D1: 59

Timing:

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To
1	Worst-case tsu	N/A	None	2.429 ns	reset	slow_clock:G10|lpm_counter:inst|cn
2	Worst-case tco	N/A	None	12.911 ns	shift_register7:G6|Q[0]	DIGIT4[0]
3	Worst-case th	N/A	None	2.694 ns	binary[1]	shift_register26:G8|Q[1]
4	Clock Setup: 'clk'	N/A	None	161.47 MHz (period = 6.193 ns)	slow_clock:G10|lpm_counter:inst|cntr_ptk:auto_generated|safe_q[18]	slow_clock:G10|g07_TFF:G0|lpm_ff:
5	Total number of failed paths					

According to this analysis, we can run the circuit on a 161MHz clock. Since we will be implementing this using a 50MHz clock, the signals have a comfortably long period to stabilize (more than 3 times the theoretical value)

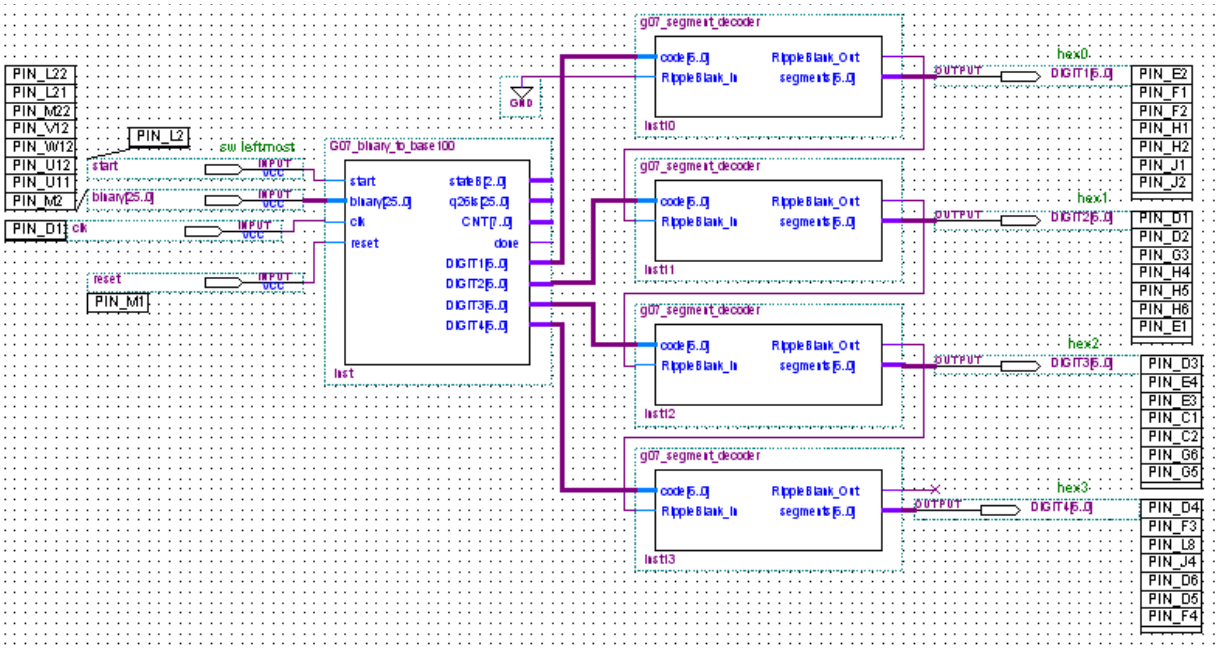
Hardware utilization:

Flow Status	Successful - Fri Nov 07 13:49:49 2008
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	g07_LAB4
Top-level Entity Name	G07_binary_to_base100
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	141 / 18,752 (< 1 %)
Total combinational functions	141 / 18,752 (< 1 %)
Dedicated logic registers	97 / 18,752 (< 1 %)
Total registers	97
Total pins	95 / 315 (30 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

We are using less than 1% of the board with this circuit, leaving plenty of room for expansion.

TESTBED:

[relevant files on /Part1/G07_binary_to_base100_TESTBED/]

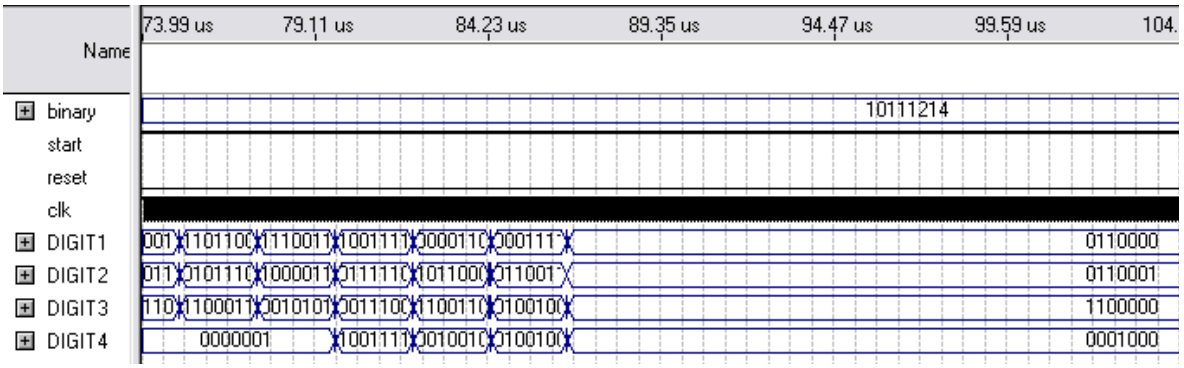


(we started a testbed with a circuit diagram –as shown above- but ended up implementing the circuit in the board using the vhdl code on 'TESTBED.vhd']

First, we simulated the TESTBED to display binary 10111213, which should be: A, b, C, D.
From the segment decoder, these letters are mapped as:

A : "0001000"
B : "1100000"
C : "0110001"
D : "1000010"

The waveform gave this exact result (after a bit of debugging)



A timing analysis of the TESTBED yielded once again adequate time for the signals to settle:

ming Analyzer Summary

Type	Slack	Required Time	Actual Time	From	To
Worst-case tsu	N/A	None	5.797 ns	key[2]	number[21]
Worst-case tco	N/A	None	18.585 ns	G07_binary_to_base100:G0|shift_register7:G6|Q[3]	hex3[5]
Worst-case th	N/A	None	0.643 ns	sw[2]	number[2]
Clock Setup: 'CLOCK'	N/A	None	168.46 MHz (period = 5.936 ns)	G07_binary_to_base100:G0|shift_register7:G6|Q[3]	G07_binary_to_base100:G0|state.s0
Total number of failed paths					

From these results, it seems that the TESTBED can operate on a clock even faster than the main component (169MHz instead of 161MHz)

IMPLEMENTAITON

(relevant files in /Part1/G07_binary_to_base100_TESTBED/)

We implemented this on the board using the segment decoder developed in LAB2. In order to load all 26 bits of the number to be displayed, we used the 4 buttons as 'load' commands and the 10 binary switches as data lines.

Using a process,

- When key(0) is pressed, the value of the switches is loaded into the 10 LSB of the number.
- When key(1) is pressed, the 10 switches are loaded into bit 19 down to 10 of binary.
- When key(2) is pressed, the 6 rightmost switches are loaded into the 6 MSB of binary. Other switches are ignored.
- When key(3) is pressed, the leftmost switch is recorded at 'START', and the second leftmost switch is recorded as 'RESET'. Every other switch is ignored.

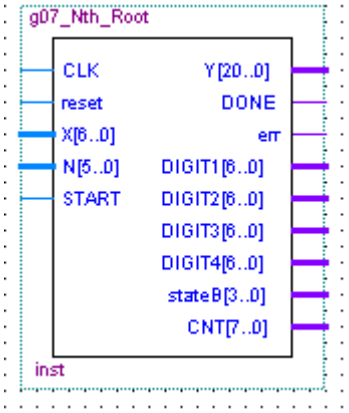
With adequate pin assignment, we implements the unit onto the Cyclone II board successfully and were successfully able to test many input values and observe the adequate output.

Flow Status	Successful - Tue Nov 18 14:46:00 2008
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	g07_LAB4
Top-level Entity Name	TESTBED
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	499 / 18,752 (3 %)
Total combinational functions	472 / 18,752 (3 %)
Dedicated logic registers	97 / 18,752 (< 1 %)
Total registers	97
Total pins	44 / 315 (14 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

From the flow summary of the simulation, we can see that 14% of the total pins are used and roughly 3% of the board's logic.

Part 2: Nth Root Circuit

Description of the Nth Root Circuit



This circuit calculates the Nth roots of a binary number X. There are some restrictions that will be applied: N has to be bigger or equal to one but smaller or equal to 63 and X is a 7 bit number between 0 and 99.

Inputs:

1) Clock : Pulses with a period of 88.32 MHz frequency. The VHDL code evaluates the different states of the FSM only at the rising edge of the clock.

2) reset: If reset = '1' we reset back to State 0, otherwise if reset = 0 it evaluates the other states.

3) X[6..0] : X is our 7 bit number that represents an integer between 0 and 99. The goal of our circuit here is to calculate the Nth root of this 7 bit number. The minimum value X can have is "0000000" and its maximum value is 1100011.

4) N : This input is used to let the user choose which type of root he intends to use (square root, third root etc…) N is a 6 bit number between 1 and 63, its minimum value is 00001 and its maximum value is 111111. The calculator can therefore compute up to the 63^{rd} root of our chosen number X.

Outputs:

Y [20..0] : Y is the Nth root of X. It has 21 bits: the first 7 bits represents the integer part of the Nth root of X and the 14 remaining bits represent the fractional part. Therefore we can represent up to 4 decimals with our 14 fractional bits. Ex: Square root of 2 = 1.4141 will give us 0000010.01010010101001;

Done : Done is equal to 1 when we're in reset state (S0) and Done is equal to 0 when we are going from state to state.

err: err is the output that flags for an error. Err is equal to one if N is not between 1 and 63 err or if X represents a number greater than 99. Otherwise it is 0.

Digit1[6..0]: This 7 bit output represents the Integer part of the Nth root of Y on the LED screen.

Digit2[6..0]: This 7 bit output represents the decimal point therefore the LED screen is always going to be blank. Digit2 is therefore always "0000000".

Digit3[6..0]: This 7 bit output represents the first and second decimal places of the Nth root of X.
Ex : square root of 3 is equal to 1.7320, in this case we will get Digit three to display 73(in base 100) on the LED screen which is 1001001 in binary.

Digit4[6..0]: This 7 bit output represent the third and fourth decimal places of the Nth root of X.
Ex : For the square root of 3 we will get Digit 4 to display 20(in base 100) on the LED screen which is 0010100 in binary.

stateB : State B is a 4 bit output to help us visualize our state transitions. Example stateB = "0000" for State 0 and sateB= "0001" for state 1.

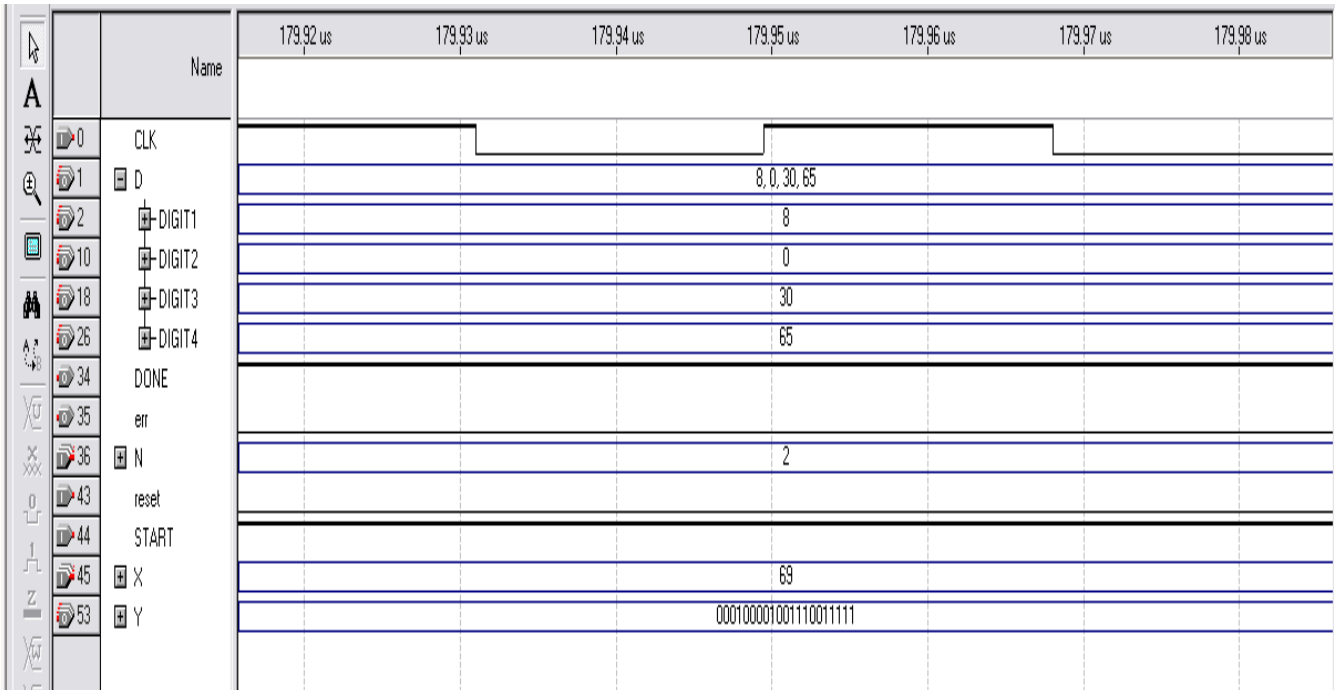
Timing Analysis of the Nth Root circuit

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To
1	Worst-case tsu	N/A	None	13.095 ns	X[0]	Y_TRANSIANT[10]~_Duplicate_1
2	Worst-case tco	N/A	None	18.336 ns	Y_TRANSIANT[2]~_Duplicate_2	DIGIT4[3]
3	Worst-case th	N/A	None	-1.119 ns	reset	Z_SHIFT[18]
4	Clock Setup: 'CLK'	N/A	None	88.32 MHz (period = 11.323 ns)	Z_SHIFT[18]	Z[31]
5	Total number of failed paths					

We can see from our timing analysis that our worst case Tsu is 13.095 nS, our worst case Tco is 18.336 nS and our worst case Th is 1.119 nS and our Clock period is 11.323 nS(frequency of 88.32MHz). Our clock runs at 50 MHz therefore our signals have sufficiently enough time to stabilize.

Waveform of the Nth Root Circuit



In this waveform we tried calculate the square root of 69. On a Calculator the answer would be 8.3066. Due to precision issue the answer on the waveform will be 8.3066*2^14 = 136095.3344, we take the

integer part which is 136095 and divide it by 2^14 we get 8.3065.

It's 8,3065 that will be shown on the waveform instead of 8,3066. As we can see in the simulation our input X is 69, our input N is 2. Therefore our output Y which represents the square root of 2 has a value of "00010000101001110011111". The first 7 bits of Y "0001000" represents the integer part of the square root of 68 which is 8.

The next "01001110011111" 14 bits represent decimal part of the square root of 69. To display the first two decimal places, we take the 14 digits and multiply it by 100 (which is 1100100 in binary) In the case of the square root of 69, this number is 30 .

To display the last two decimal places we take the previous result of multiplying the 14 bit fractional number by 100 and multiply it again by 100, which is equivalent to multiplying it by 10000. We decided to run two multiplications of 100 in parallel instead of one multiplication of 10000 such that both of them can be processed at the same time to adapt to faster clocks. In the case of the square root of 69, this digit represents 65.

To display the square root of 69, our first LED screen which is Digit 1 represents the integer part which is 8. Digit 2 represents the decimal point therefore it is always zero. The first two decimal places of the square root of 69 are represented in Digit 3 and the last two decimal places are represented by Digit 4

Flow Summary for Nth Root Circuit

Flow Status	Successful - Tue Nov 18 15:28:00 2008
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	g07_Nth_Root
Top-level Entity Name	g07_Nth_Root_TESTBED
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	563 / 18,752 (3 %)
Total combinational functions	552 / 18,752 (3 %)
Dedicated logic registers	123 / 18,752 (< 1 %)
Total registers	123
Total pins	45 / 315 (14 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	7 / 52 (13 %)
Total PLLs	0 / 4 (0 %)

We can see that our Nth Root circuit is using 14% of the total pins and less than 1% of the total logic elements.

Testbed

For the Testbed in Part 2, it is quite similar to the one in part 1 except that instead of loading a 26 bit binary number we are loading a 7 bit binary number (which represents X) and a 6 bit binary number which represents(N).

Implementation:

(relevant files in /DSD_LAB4/Part2/TESTBED/g07_Nth_Root_TESTBED.vhd)

We implemented this on the board using the segment decoder developed in LAB2. In order to load all our data, we used 3 of the buttons as 'load' commands and the 10 binary switches as data lines.

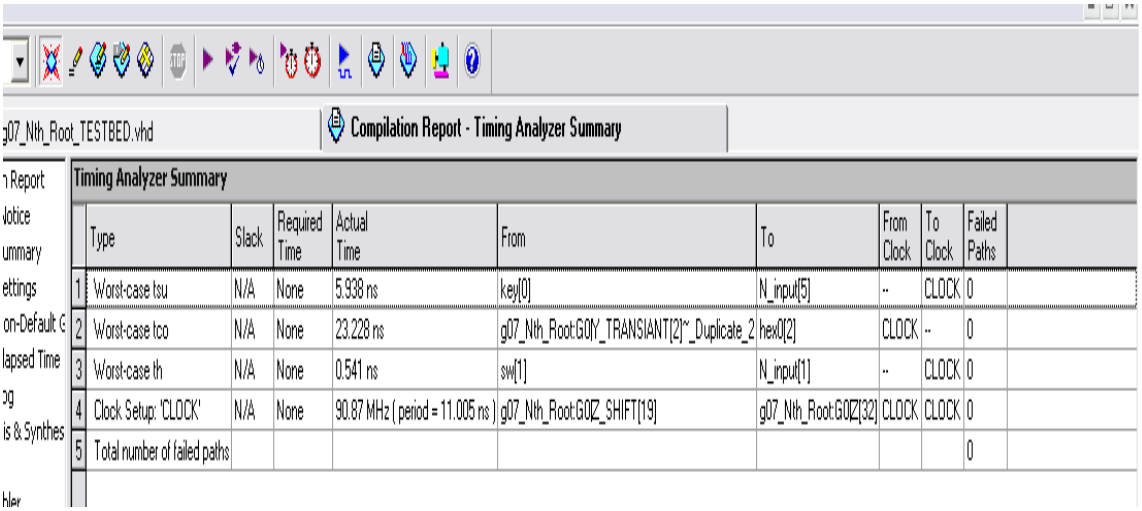
Using a process,

- When key(0) is pressed, the value of the 7 rightmost switches are loaded into X.
- When key(1) is pressed, the value of the 6 rightmost switches are loaded into N.
- When key(2) is pressed, nothing happens.

- When key(3) is pressed, the leftmost switch is recorded at 'START', and the second leftmost switch is recorded as 'RESET'. Every other switch is ignored.

With adequate pin assignment, we implements the unit onto the Cyclone II board successfully and were successfully able to test many input values and observe the adequate output.

Timing Analysis for our TestBed



g07_Nth_Root_TESTBED.vhd | Compilation Report - Timing Analyzer Summary

Timing Analyzer Summary

	Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
1	Worst-case tsu	N/A	None	5.938 ns	key[0]	N_input[5]	--	CLOCK	0
2	Worst-case tco	N/A	None	23.228 ns	g07_Nth_Root:G0|Y_TRANSIANT[2]~_Duplicate_2	hex0[2]	CLOCK	--	0
3	Worst-case th	N/A	None	0.541 ns	sw[1]	N_input[1]	--	CLOCK	0
4	Clock Setup: 'CLOCK'	N/A	None	90.87 MHz (period = 11.005 ns)	g07_Nth_Root:G0|Z_SHIFT[19]	g07_Nth_Root:G0|Z[32]	CLOCK	CLOCK	0
5	Total number of failed paths								0

Our worst case Tsu for our testbed circuit is 5.938 nS, our worst case Tco is 23.228 nS and finally our worst case Th is 0.541 nS. This circuit runs at a clock of a maximum frequency of 90.87 Mhz. Our current clock is at 50 MHz and therefore we are within the convenient. range

Flow Summary of Testbed

Flow Status	Successful - Tue Nov 18 14:21:33 2008
Quartus II Version	8.0 Build 215 05/29/2008 SJ Full Version
Revision Name	g07_Nth_Root
Top-level Entity Name	g07_Nth_Root_TESTBED
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	563 / 18,752 (3 %)
Total combinational functions	552 / 18,752 (3 %)
Dedicated logic registers	123 / 18,752 (< 1 %)
Total registers	123
Total pins	45 / 315 (14 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	7 / 52 (13 %)
Total PLLs	0 / 4 (0 %)

We can see from the Flow Summary that the total pins used is less than 14% than the total pins on the board and that the total logic used is less than 3% of the total logic elements on the borad.