Midterm Exam

ECE 448
Spring 2011

Wednesday Section

(15 points)

Instructions:

Please read this entire document carefully before beginning!

Zip all your deliverables into an archive <last_name>.zip and submit it through Blackboard no later than Wednesday, March 9, 10:15 PM EST.

Your task is to describe in VHDL, debug, and implement a simple hash function.

Introduction:

The **AV-4** hash function circuit is specified below using its:

a. Pseudocode,
b. Block Diagram,
c. Interface,
d. Table of input/output ports,
e. Timing requirements.

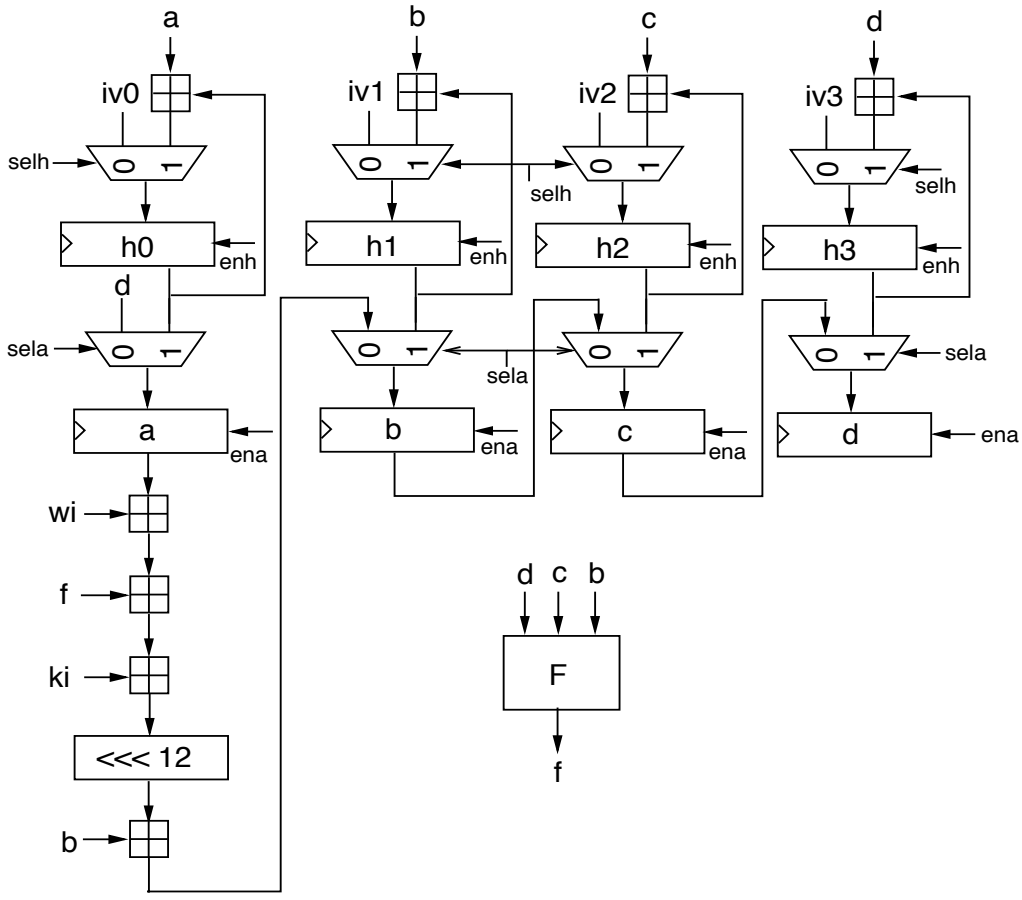
Pseudocode:

```
//Initialize hash result:
var int h0 := iv0
var int h1 := iv1
var int h2 := iv2
var int h3 := iv3

//Process the message in successive 64-bit blocks:
for each 64-bit block of the message
    break the block into four 16-bit words w[i], 0 ≤ i ≤ 3

    //Initialize the state for the block:
    var int a := h0
    var int b := h1
    var int c := h2
    var int d := h3

    //Main loop:
    for i from 0 to 3
            f := (b and c) or ((not b) and d)
            wi := w[i]
        temp := d
        d := c
        c := b
        b := b + leftrotate((a + f + ki + wi), 12)
        a := temp
    end for

    //Add the state to the hash result so far:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
end for

digest := h0 || h1 || h2 || h3
```

Notation:

All variables, except i and digest, represent 16-bit words.
iv0..iv3 : initialization vector
h0..h3 : intermediate hash result
digest: output value
ki : round constant
w[i]: message block words, i=0..3 for a single message block, w[0] represents the least significant word of the message block.
not : one's complement of a 16-bit word.
xor, and, or : Boolean operations on 16-bit words.
leftrotate(a, r) : rotation of the variable **a** by **r** positions to the left
a || **b:** a concatenated with b.

Block Diagram:



All registers have clock, reset, and enable even if these inputs are not shown in the diagram. Enable signals are specified in the table on the next page. a, b, c, d signals mean output of the respective register.

Interface:

Assume the following interface to your circuit.

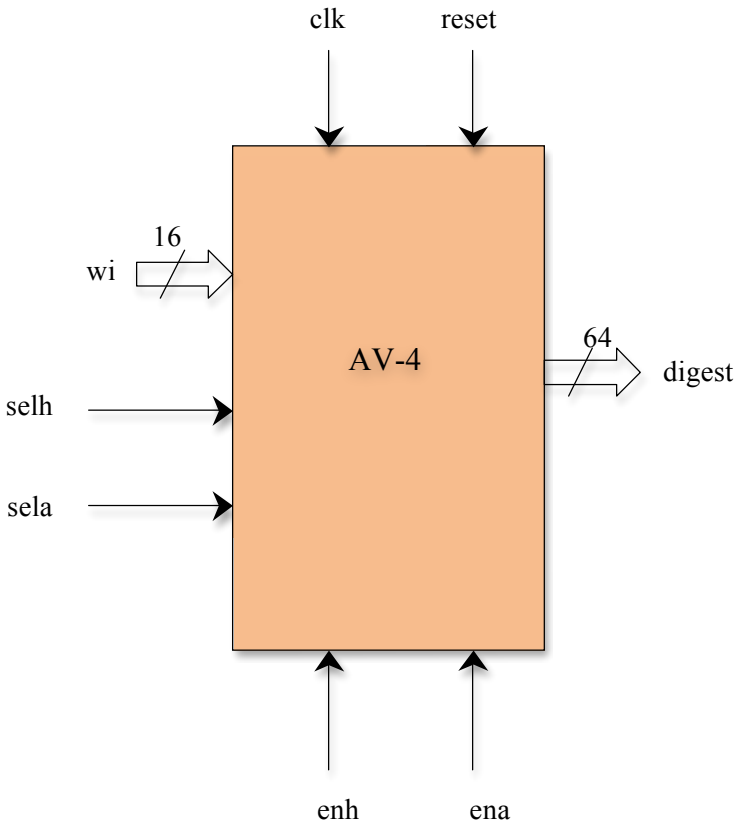


Table of input/output ports:

Port	Mode	Width	Function
clk	Input	1	System clock.
reset	Input	1	Asynchronous system reset – clears all internal register.
enh	Input	1	Common enable for the registers h0, h1, h2 and h3.
ena	Input	1	Common enable for registers a, b, c and d.
selh	Input	1	Select signal to choose between initialization and loading in new value to registers h0, h1, h2, and h3.
sela	Input	1	Select signal to choose between initialization and loading in new value to registers a, b, c and d.
wi	Input	16	A subsequent word of a message block, starting from w[0] and ending with w[3].
digest	Output	64	digest = h0 || h1 || h2 || h3.

Timing Requirements:

Assume that

- **one clock cycle is used for the once-per-message initialization:**
  `h0 = iv0; h1 = iv1; h2 = iv2; h3 = iv3;`
- **one clock cycle is used for the once-per-block initialization:**
  `a = h0; b = h1; c = h2; d = h3;`
- **one round of the main for loop of the pseudocode executes in one clock cycle; there are a total of 4 rounds.**
- **one clock cycle is used for the once-per-block finalization:**
  `h0 = h0 + a; h1 = h1 + b; h2 = h2 + c; h3 = h3 + d;`

As a result, hashing of the message M, consisting of N 64-bit blocks (each block=4 16-bit words) should last 1+(1+4+1)*N clock cycles. The hash value is then written to the destination circuit in one additional clock cycle.

Constants:

iv0 = 0xefcd
iv1 = 0x98ba
iv2 = 0x1032
iv3 = 0xdcfe
ki = 0xab89

Inputs, Outputs and Intermediate Values:

Input Message Words:

Block 1:
w[0] = 0x9e10
w[1] = 0x7d9d
w[2] = 0x372b
w[3] = 0xb682

Block 2:
w[0] = 0x1d35
w[1] = 0x42a4
w[2] = 0x19d6
w[3] = 0x101e

Output:

digest = 0xe36b86735ac67b8c

Intermediate values of the variables a-d, and h0-h3 during the circuit operation:
All values are represented in hexadecimal notation.

Clock cycle	a	b	c	d	h0	h1	h2	h3
0	0000	0000	0000	0000	0000	0000	0000	0000
1	0000	0000	0000	0000	efcd	98ba	1032	dcfe
2	efcd	98ba	1032	dcfe	efcd	98ba	1032	dcfe
3	dcfe	6197	98ba	1032	efcd	98ba	1032	dcfe
4	1032	c304	6197	98ba	efcd	98ba	1032	dcfe
5	98ba	07ce	c304	6197	efcd	98ba	1032	dcfe
6	6197	adab	07ce	c304	efcd	98ba	1032	dcfe
7	6197	adab	07ce	c304	5164	4665	1800	a002
8	5164	4665	1800	a002	5164	4665	1800	a002
9	a002	9207	4665	1800	5164	4665	1800	a002
10	1800	db8a	9207	4665	5164	4665	1800	a002
11	4665	42c6	db8a	9207	5164	4665	1800	a002
12	9207	400e	42c6	db8a	5164	4665	1800	a002
13	9207	400e	42c6	db8a	e36b	8673	5ac6	7b8c

Timing Waveform:

Design Requirements

The combinational portion of the circuit should be described using the dataflow VHDL code, and the sequential portion of the circuit should be described using the synthesizable behavioral code. Your code should infer a circuit that requires a minimum amount of FPGA resources. The target clock frequency should be 50 MHz.

Tasks

Perform the following tasks:

1. Write a synthesizable VHDL code representing the described above circuit.
2. Write a testbench verifying the operation of your circuit for inputs shown given above.
3. Perform functional simulation of your circuit and use it to debug your VHDL code. Take screen shots of the waveform.
4. Synthesize your circuit.
5. Implement your circuit using
   FPGA family: Spartan3E,
   Device: XC3S1600E
   Speed Grade: -4.
6. Run the static timing analysis of your circuit.
7. Based on the circuit block diagram and the implementation reports, determine the most critical path in your circuit and the circuit maximum clock frequency.
8. Based on the implementation reports and the report from the static timing analysis, determine the number of CLB slices, Logic Cells, LUTs, D flip-flops, and pins used by the circuit.
9. Perform timing simulation of your circuit. Take screen shots of the waveform.

Deliverables

1. VHDL code of your entire circuit fulfilling the requirements specified in the Design Requirements section above.
2. VHDL code of your testbench.
3. Timing waveforms from the functional and timing simulations demonstrating the correct operation of your circuit. Take screen shots and include in the report.
4. Description of the critical path of your circuit.
5. FPGA resource utilization (as defined in Task 8 above)
6. Minimum clock period and maximum clock frequency of your circuit.