Dot-Product Controller (DPC)

Project

This project focuses on designing and implementing a **Vector Dot Product System** on the Basys3 FPGA, building upon the Multiply-Accumulate (MAC) unit from the previous assignment. The dot product is defined as the sum of a[i]b[i] for all index i in vector a and b.

The system is divided into the following components:

1. **Input Handling (Switches & Controls):**
   - Vectors **A** and **B** are provided via slider switches SW7–SW0.
   - SW8–SW9 select the index (0–3) of the element.
   - SW12 and SW13 act as write enables for A and B respectively, while SW14 and SW15 act as read enables.
   - The **BTNC button** resets the system and displays "-rSt" for 5 seconds.
2. **Dot Product Computation (Controller + MAC):**
   - A controller manages data entry, initiates the MAC operation, and sequences through elements using a counter (0–4).
   - When all elements are written, the controller enables the MAC to accumulate results.
   - Overflow conditions are detected, and the display shows "OFLO" if the result exceeds 16 bits.
3. **Output Display:**
   - Results are shown on the **4-digit 7-segment display** in hexadecimal.
   - During read operations, individual elements or the final dot product can be viewed.
4. **Deliverables:**
   - **Verilog modules** implementing the controller and system integration.
   - **Testbench** with waveforms showing reset, normal operation, and overflow.
   - **Constraint files (.xdc) and bitstream:** FPGA pin mapping and machine code.
   - **Short report** (4-5 pages) documenting design approach, simulation, synthesis, and schematics.

Design Explanation and implementation

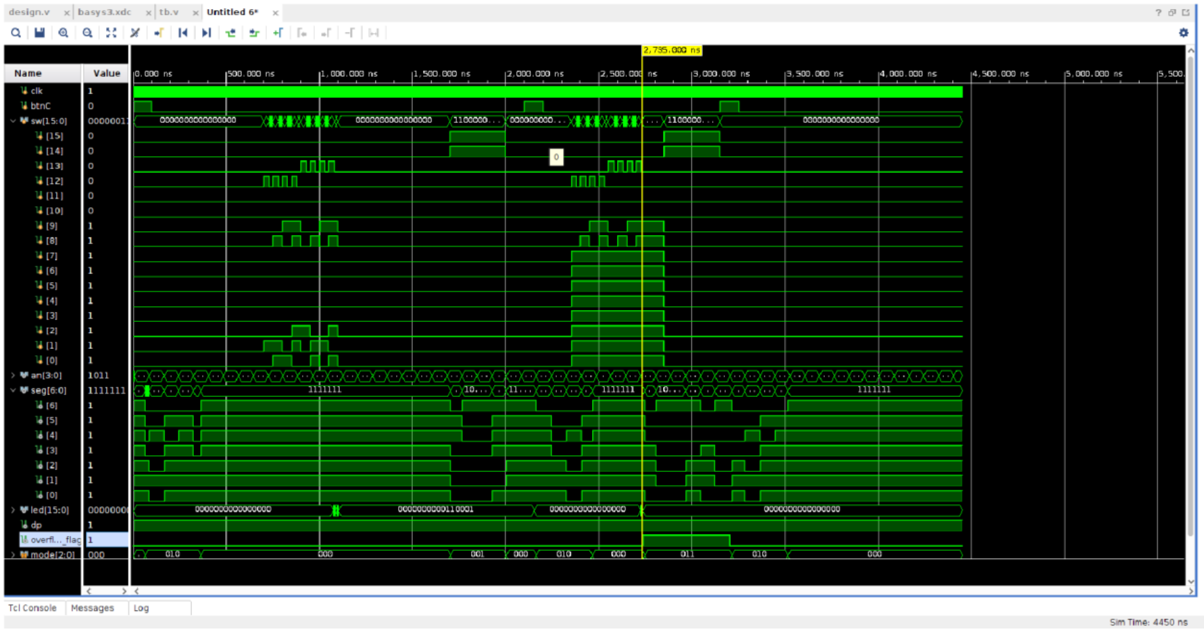
The design aims to implement the **dot product of two 4-element vectors** on the Basys3 FPGA board using the previously built **Multiply-Accumulate (MAC) unit** as the core computational block. The system is organized into three main parts: **input interface**, **controller unit**, and **output display**.

1. **Input Interface:**

- The elements of vectors **A** and **B** are entered using the slider switches SW7–SW0, which allow 8-bit values.
- The position of the element (index 0–3) is selected using switches SW8–SW9.
- Write enables (SW12 for A and SW13 for B) store the values into their respective vector registers. Similarly, read enables (SW14 for A and SW15 for B) control the display of stored elements.
- A reset input (BTNC) clears all registers and initializes the system, briefly showing "-rSt" on the display to confirm reset.

2. **Controller Unit:**
    - The controller supervises data flow into the MAC unit. Once all elements of A and B are stored, it initiates a **counter-based sequencing mechanism**.
    - The counter iterates from 0 to 3, successively feeding pairs of elements A[i], B[i] into the MAC. On each clock cycle, the product of the current pair is accumulated into a running sum.
    - After the counter reaches its maximum, a flag dot_product_computed is set high. The accumulated result represents the final dot product.
    - Overflow detection is included: if the computed value exceeds 16 bits, an **overflow flag** is raised and the display shows "OFLO".
3. **Output Display:**
    - The **7-segment display** is multiplexed to show either individual vector elements (during read operations) or the final dot product. Values are displayed in **hexadecimal format**, supporting compact representation of larger numbers.
    - The **LED** continuously reflects the value in accumulator if not overflowed.

Simulation Result

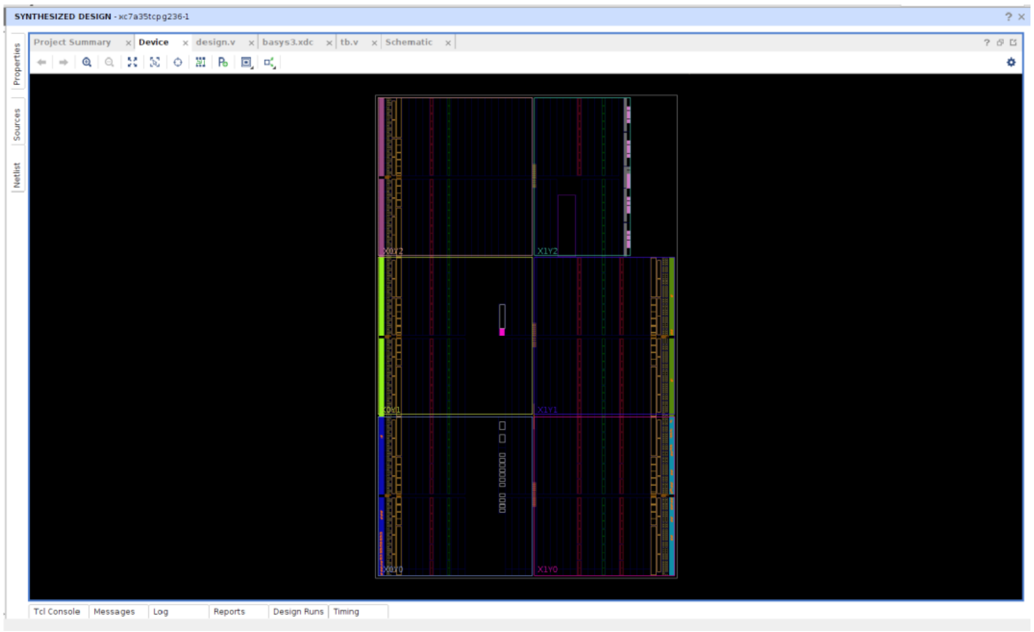


Name	Value
clk	1
btnC	0
sw[15:0]	0000001
[15]	0
[14]	0
[13]	0
[12]	0
[11]	0
[10]	0
[9]	1
[8]	1
[7]	1
[6]	1
[5]	1
[4]	1
[3]	1
[2]	1
[1]	1
[0]	1
an[3:0]	1011
seg[6:0]	1111111
[6]	1
[5]	1
[4]	1
[3]	1
[2]	1
[1]	1
[0]	1
led[15:0]	0000000
dp	1
overfl..._flag	1
mode[2:0]	000

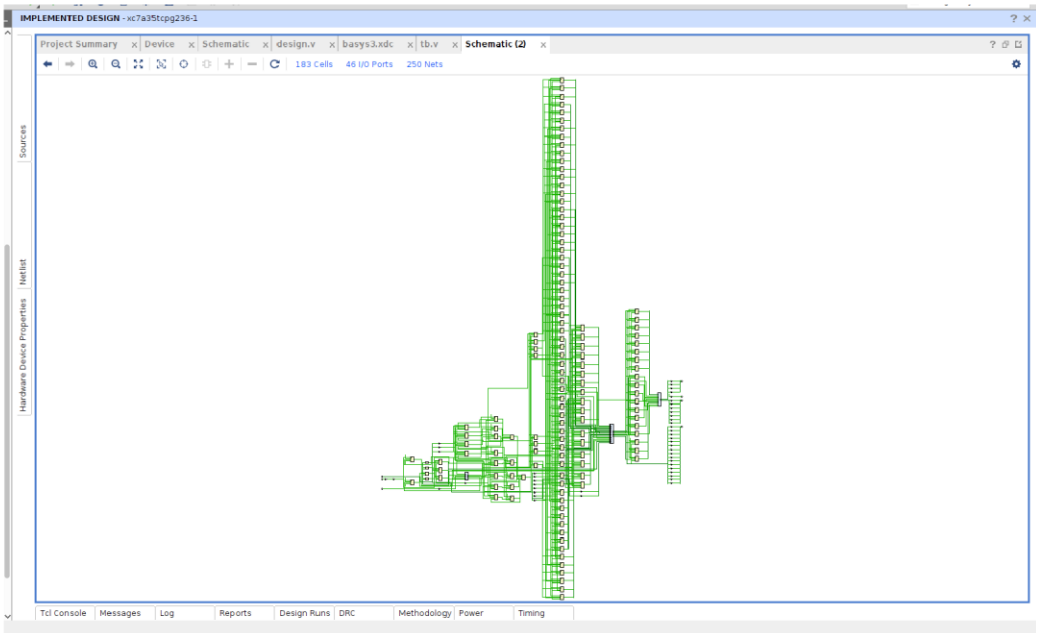
In the shown design simulation, I have represented:

1. **Reset function:** controlled by the btnC button, in the simulation time period, that button is pressed 3 times, and all the values of accumulator (represented by the led array), overflow (represented by overflow_flag), A, B and almost everything is resetted. The display works correctly as the segment shows some deviation for first 20 cycles (small specifically for the simulation). Corresponds to the mode 10 of display (i.e representing "-rst" in the display).
2. **Normal Working**: after the first successful, mode resets to 00 (blank), and after a while the data of A={2,3,1,6} and B={4,3,2,5}. As the all the 8 bits are filled, we can see a spontaneous change in led representing the accumulator value to the dot product within next 4 clock cycles (value of 49 = 110001 (in binary)). After with the value is flushed by the next reset.
3. **Overflow:** after 2nd reset we again set high values of A={255,255,255,255} and B={255,255,255,255} which led to overflow in the accumulator value (exceeding its 16 bits). As you can see in figure, the yellow line shows that as long as the last value is set, overflow_flag becomes 1 within next 4 clock cycles. Also the mode of display goes to 011 representing "OFLO" on the seven segment display.

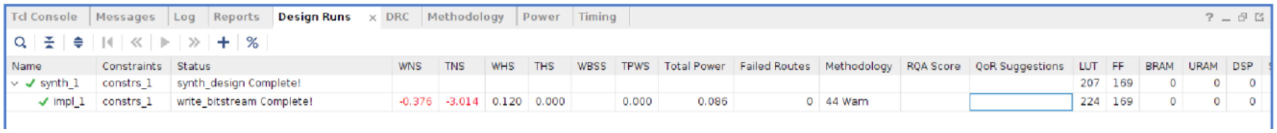
Design suite



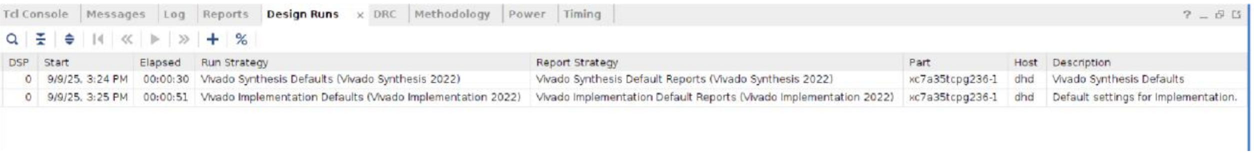
Schematic Diagram (of combined gates)



Synthesis Report



Name	Constraints	Status	WNS	TNS	WHS	THS	WBSS	TPWS	Total Power	Failed Routes	Methodology	RQA Score	QoR Suggestions	LUT	FF	BRAM	URAM	DSP
✓ synth_1	constrs_1	synth_design Complete!												207	169	0	0	0
✓ impl_1	constrs_1	write_bitstream Complete!	-0.376	-3.014	0.120	0.000		0.000	0.086	0	44 Warn			224	169	0	0	0



DSP	Start	Elapsed	Run Strategy	Report Strategy	Part	Host	Description
0	9/9/25, 3:24 PM	00:00:30	Vivado Synthesis Defaults (Vivado Synthesis 2022)	Vivado Synthesis Default Reports (Vivado Synthesis 2022)	xc7a35tcpg236-1	dhd	Vivado Synthesis Defaults
0	9/9/25, 3:25 PM	00:00:51	Vivado Implementation Defaults (Vivado Implementation 2022)	Vivado Implementation Default Reports (Vivado Implementation 2022)	xc7a35tcpg236-1	dhd	Default settings for Implementation.

Number of

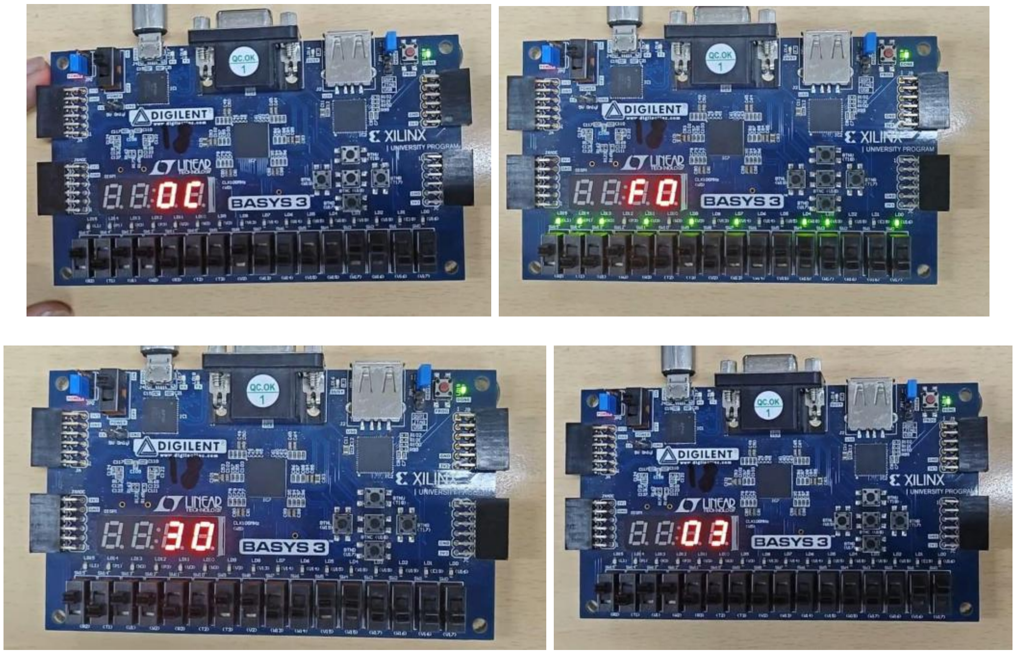
1. LUTs=224
2. Flip flops=169
3. BRAM=0
4. DSP=0

Programming the Basys 3™ FPGA Board

After generating the bitstream of the 4-digit display system, using the constraint files and source file:

- **Constraint file** containing the data of mapping and active switches/7-segment displays.
  - **Source file** containing the logic according to which the board will be programmed.
- **The .bit file** is programmed inside the Basys 3™ FPGA Board.

We used 14 different switches for input (SW0-SW9 and SW12-SW15) and 4 seven-segment displays with 7 cathode segments plus 4 anode controls for the output.



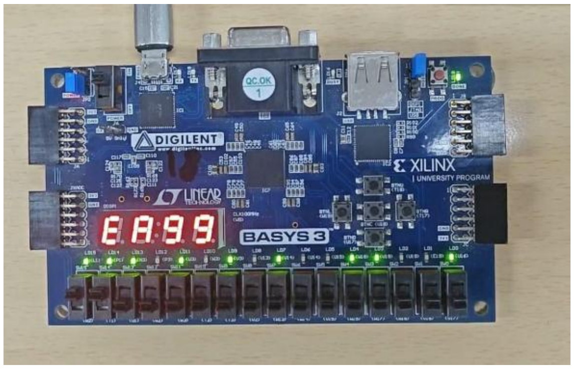
Setting up:

(i) 03 to A[0] and B[0]

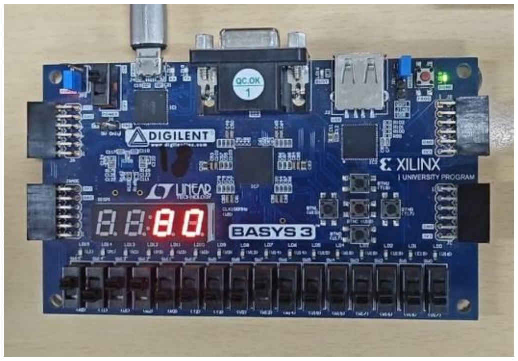
(ii) 0C (12 in decimal) to A[1] and B[1]

(iii) F0 (240 in decimal) to A[2] and B[2]

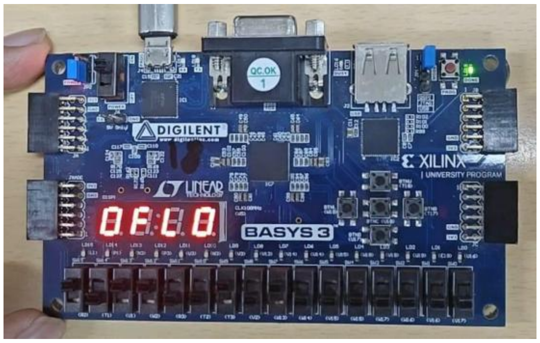
(iv) 30 (48 in decimal) to A[3] and B[3]



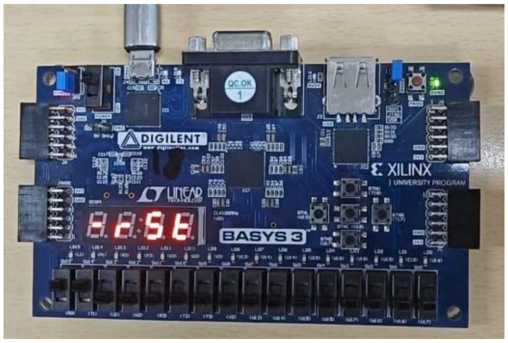
Showing the result of Dot product



setting all the values of A and B vector to 80 (128 in decimal notation) to get overflow



As long as all the values are set to be 80, “OFLO” shows up on the 7 segment display.



Resetting the board after all the operations on it.

Thanking you!