Vivado HLS tool Report

Ahmed yahia

Advantages that we can use

* The tool interface is very easy
* We can write c++ code and turn it into Verilog
* Through simulation wave forms can be obtained
* We can target zynq fpga directly
* Specify the type of memory easily (FIFIO ,ROM ,BRAM , RAM )
* Using directives we can easily pipeline the codes
* Gives estimations to frequency , clock cycles and resources utilization
* Compare different solutions at time
* Test the generated code against the c++ testbench
* Useful flags inside the Verilog that help understanding the operation

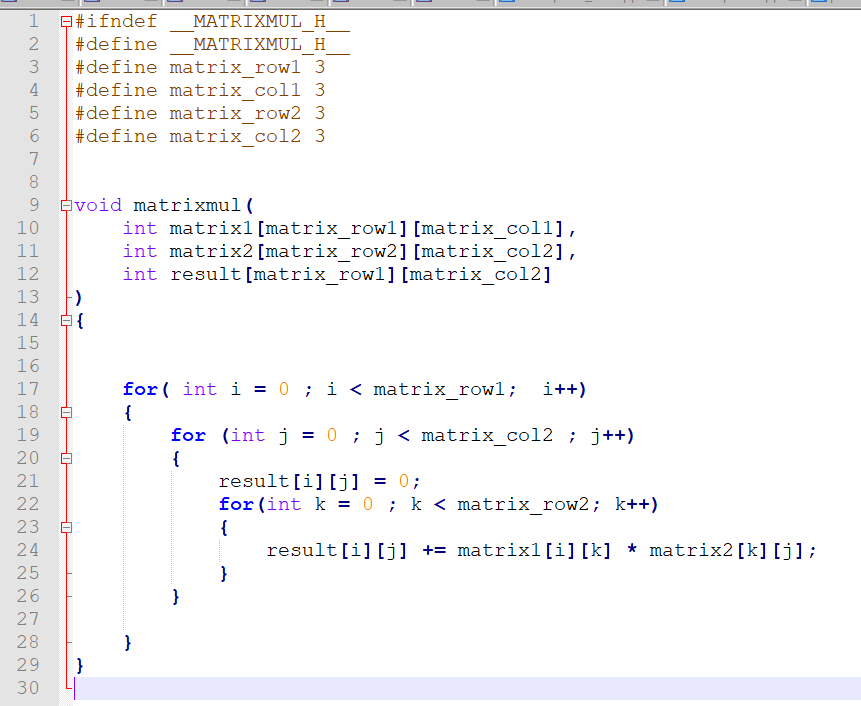
Disadvantage

* The c++ have some limitations (no dynamic arrays -recursion – system command )
* The generated code is not easily readable due using pragamas for targeted devices and the Fsm generated

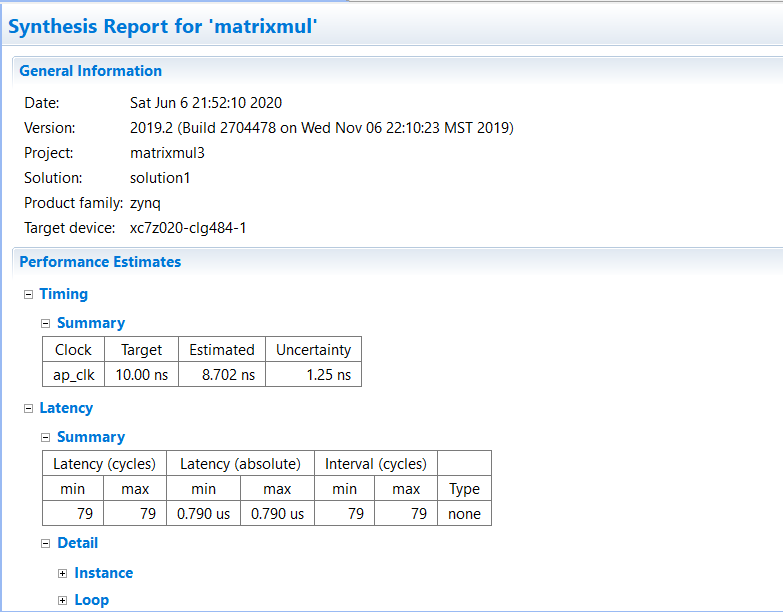
Tested projects

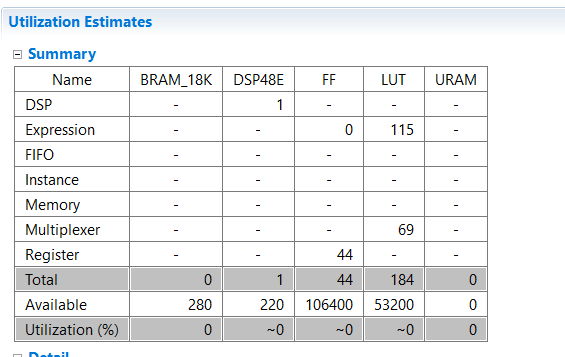
**A. Matrix multiplication**

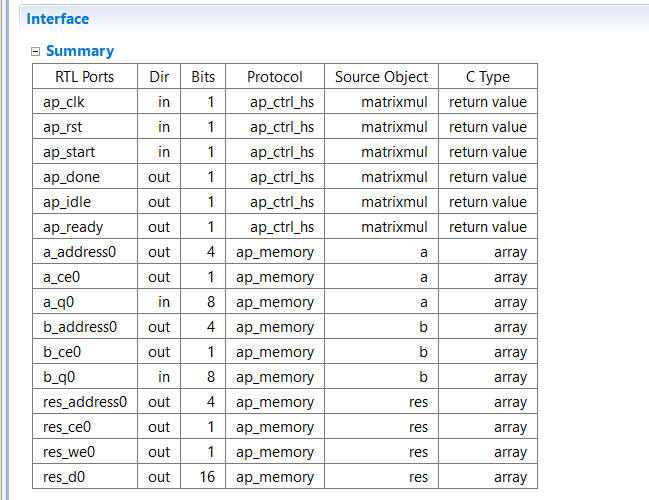
CODE :



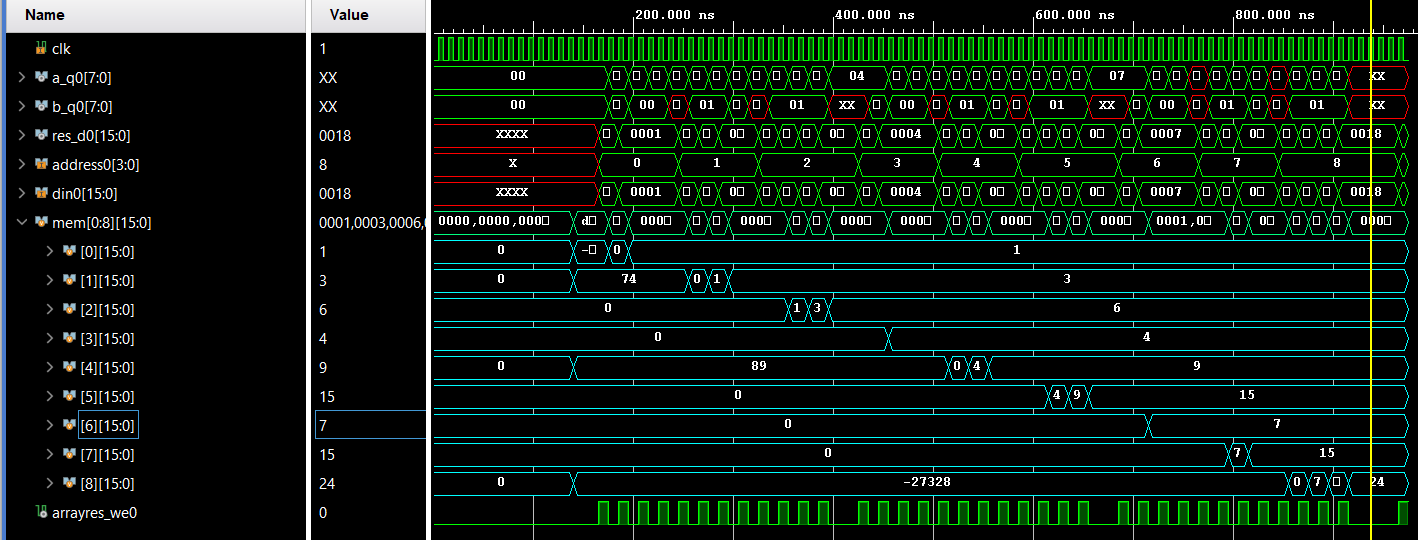
**Synthesized code output report :**



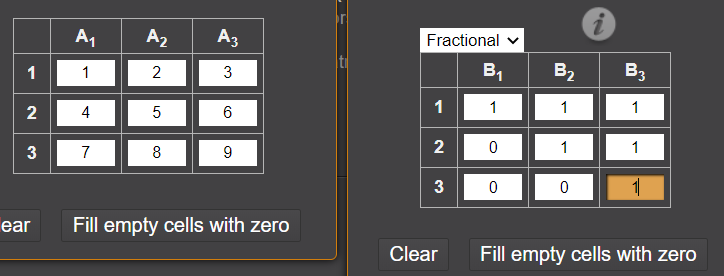


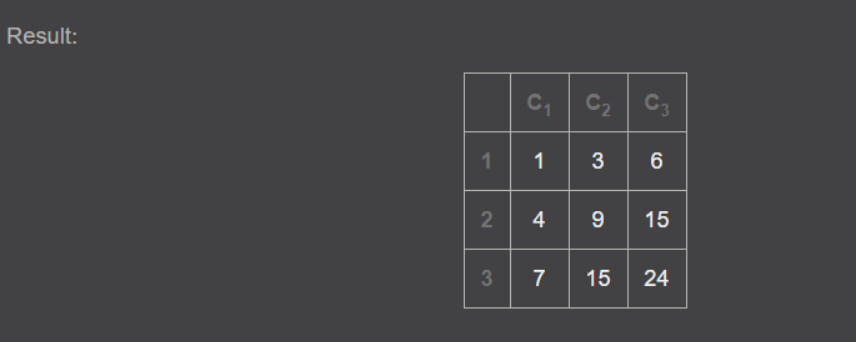


**Simulation screenshot**



The memory in the aqua is output array that has the res\_d0 saved in it .It shows clearly that it is the expected output. By tracing the wave form of inputs and the result the timing of every thing goes as expected .the testbench of matrices are





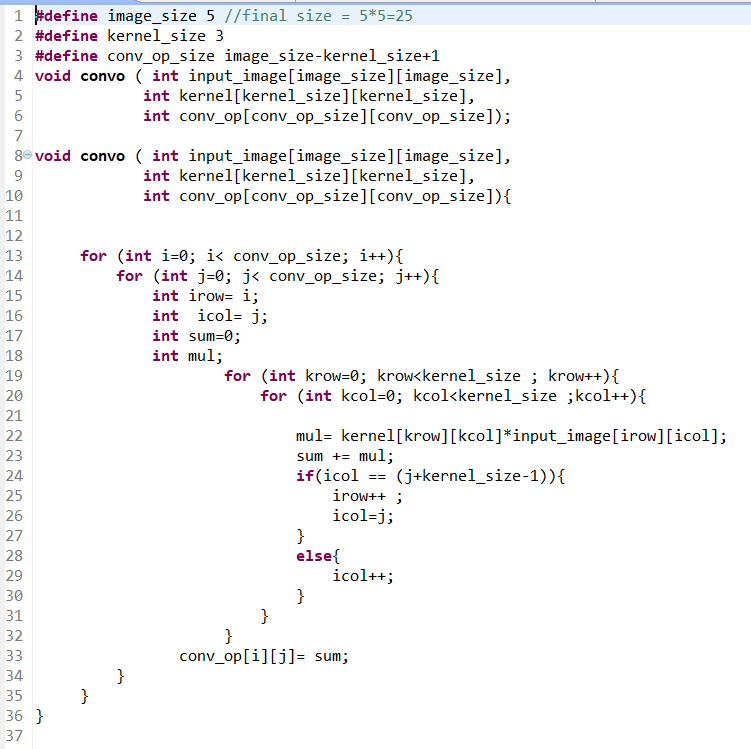
Notes :

-pipelining was used to reduce latency and clock cycles but increased resources

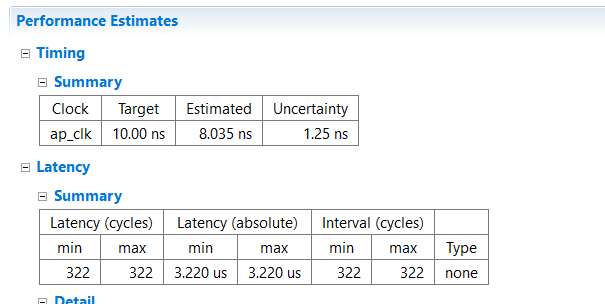
On many levels the inner for loop and the outer one

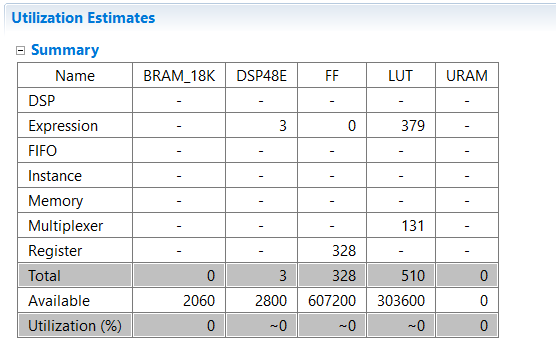
**B.CONVOLUTION**

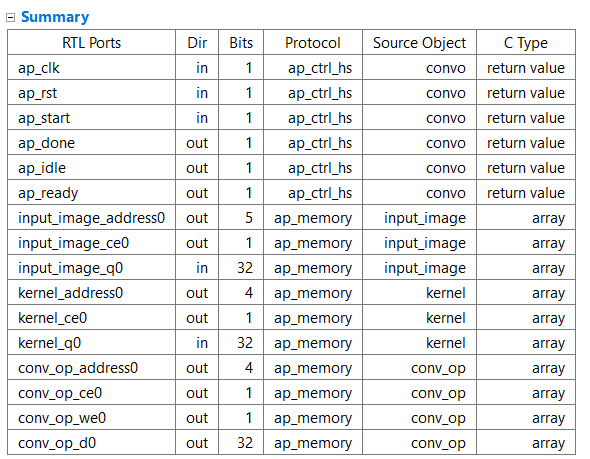
CODE :



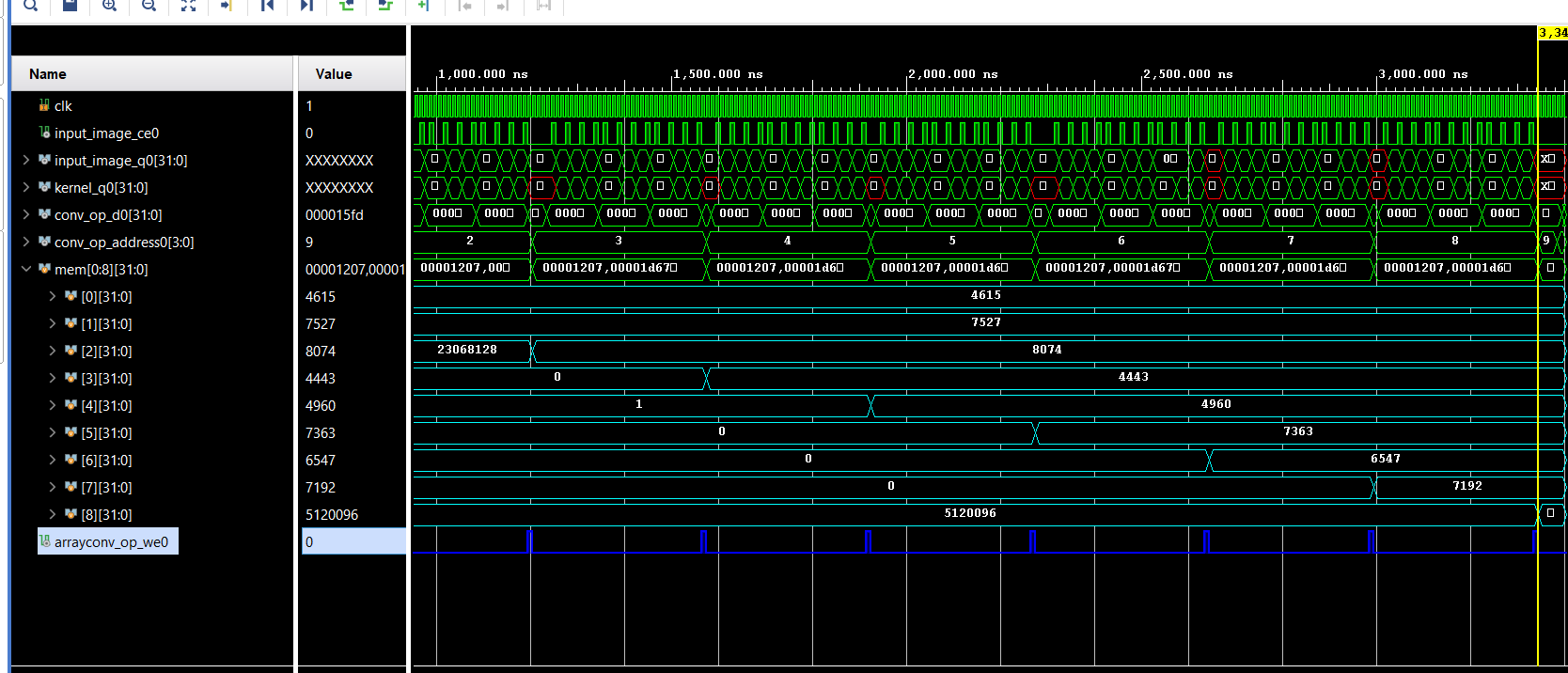
**Synthesized code output report :**



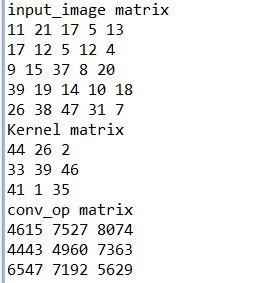




**Simulation screenshot**

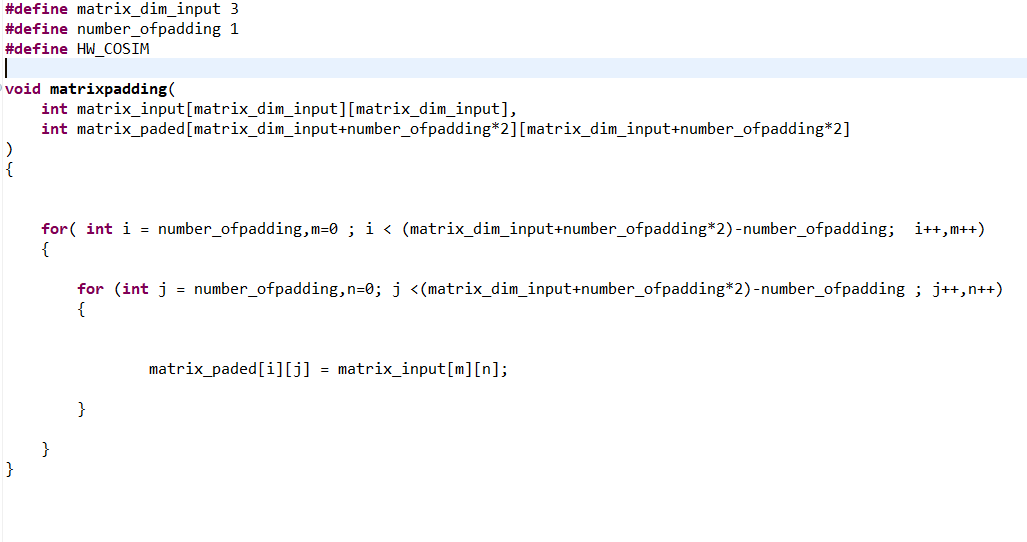


The simulation shows that convolution process is carried out as intend in the c++ code and below output is output of c++ testbench and the output exactly the same. Noting that the we0 (write signal in the output memory (array)is high only when the correct result signals has arrived can be specified by the clock period during project )

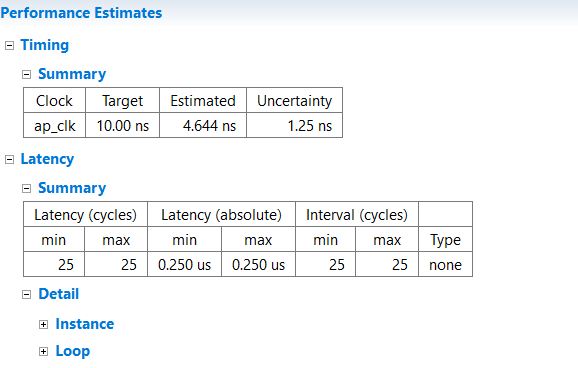


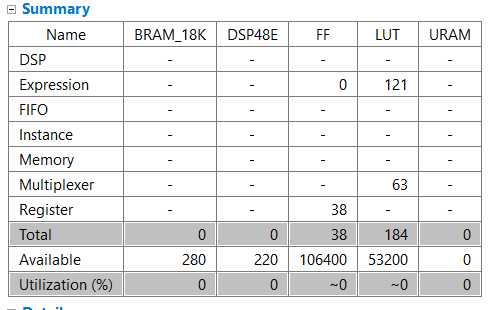
**C. Padding**

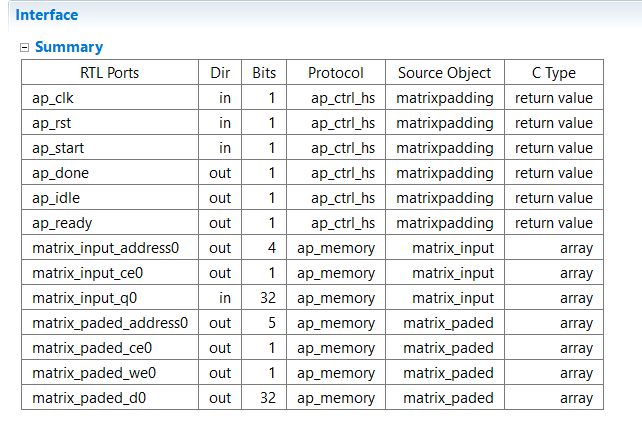
Code



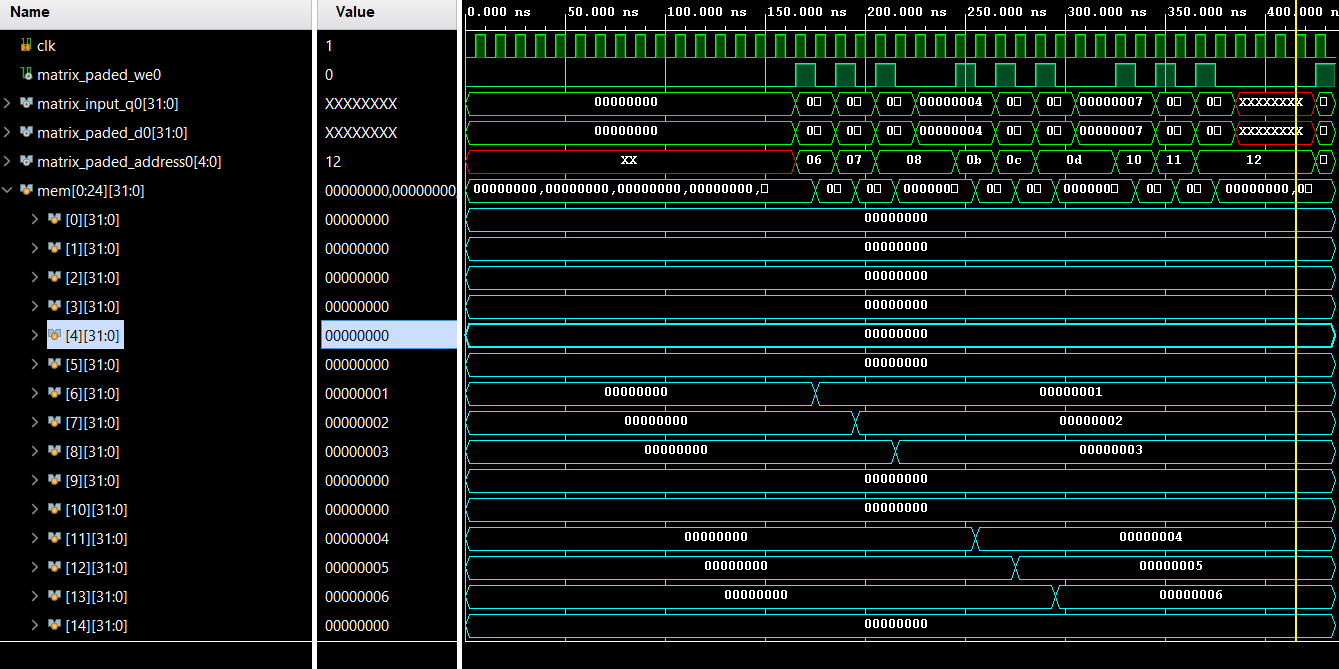
**Synthesized code output report :**



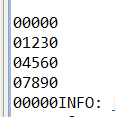




**Simulation screenshot**

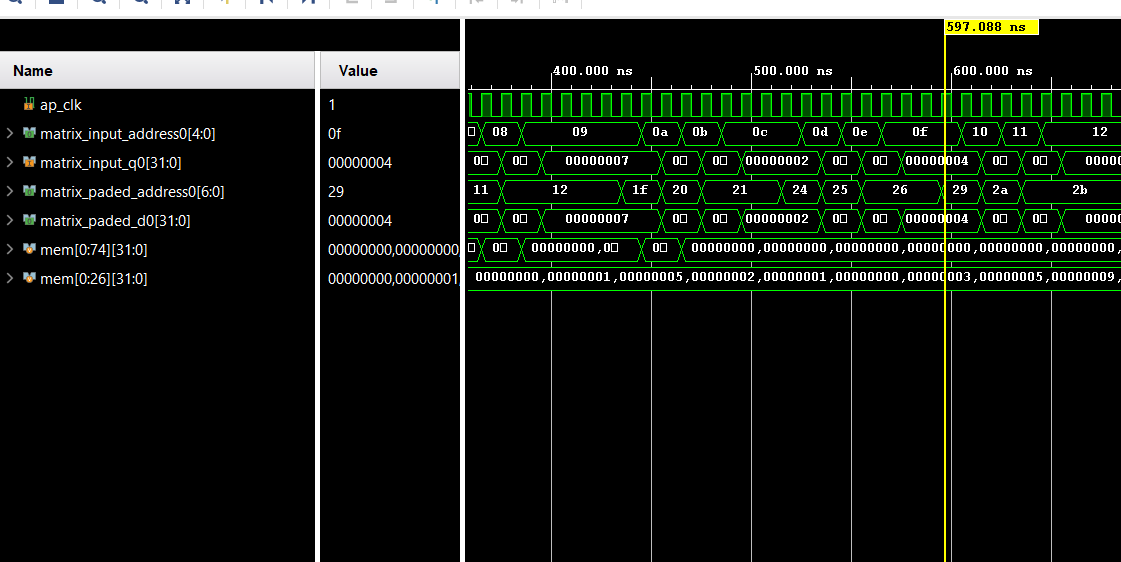


Equal to c++ simulation output



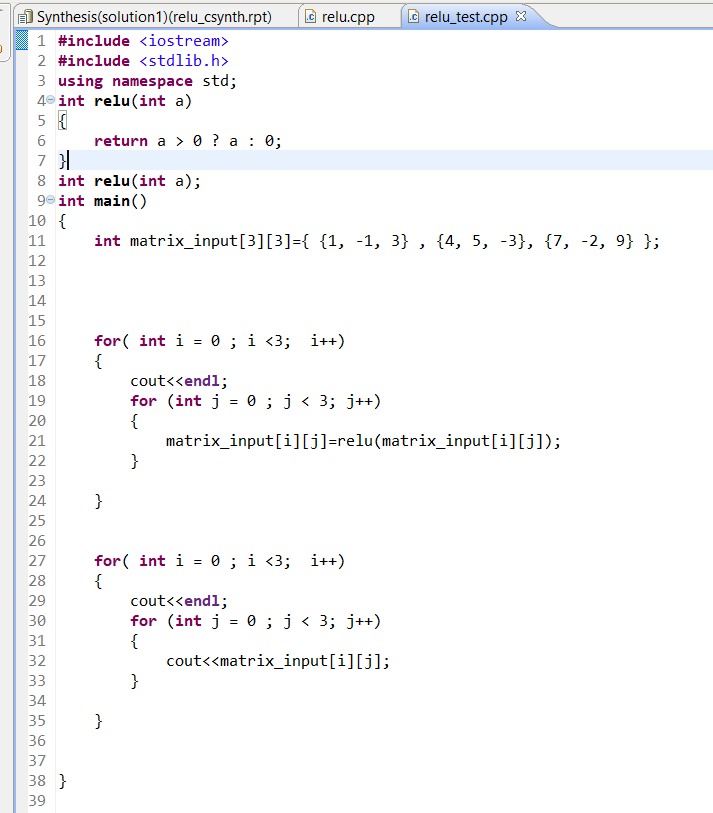
**When the input is 3d Matrix (channels \* columns\*rows)**

Screen shot for the simulation

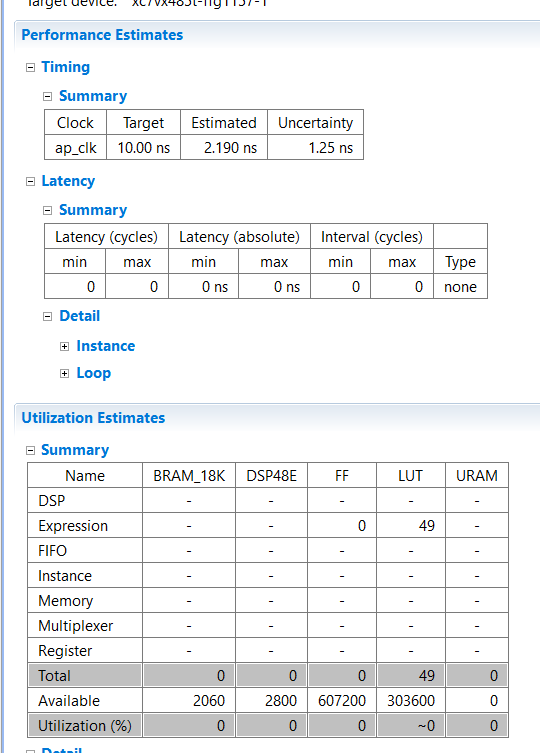


The observation is that the HLS tool put the 3d array as sequence of element after each other **.**

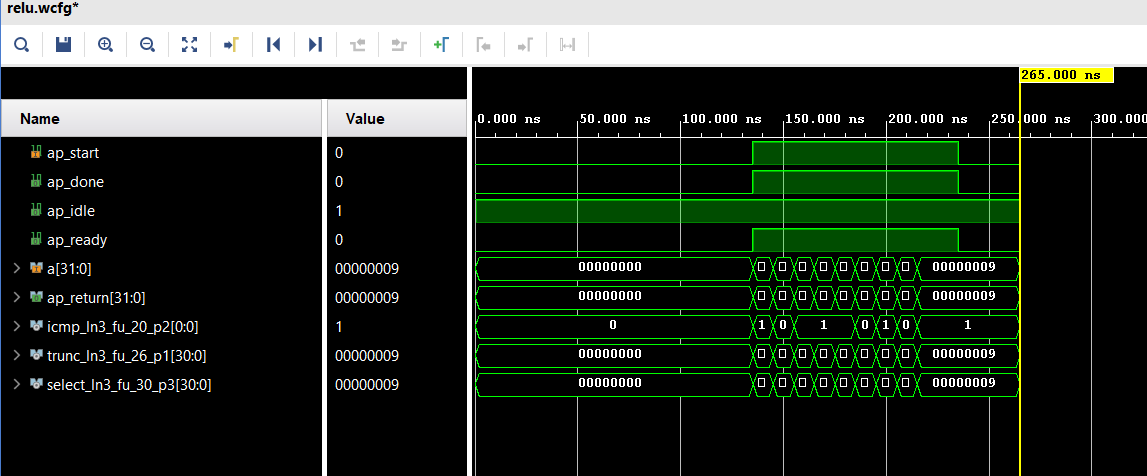
**RELU**



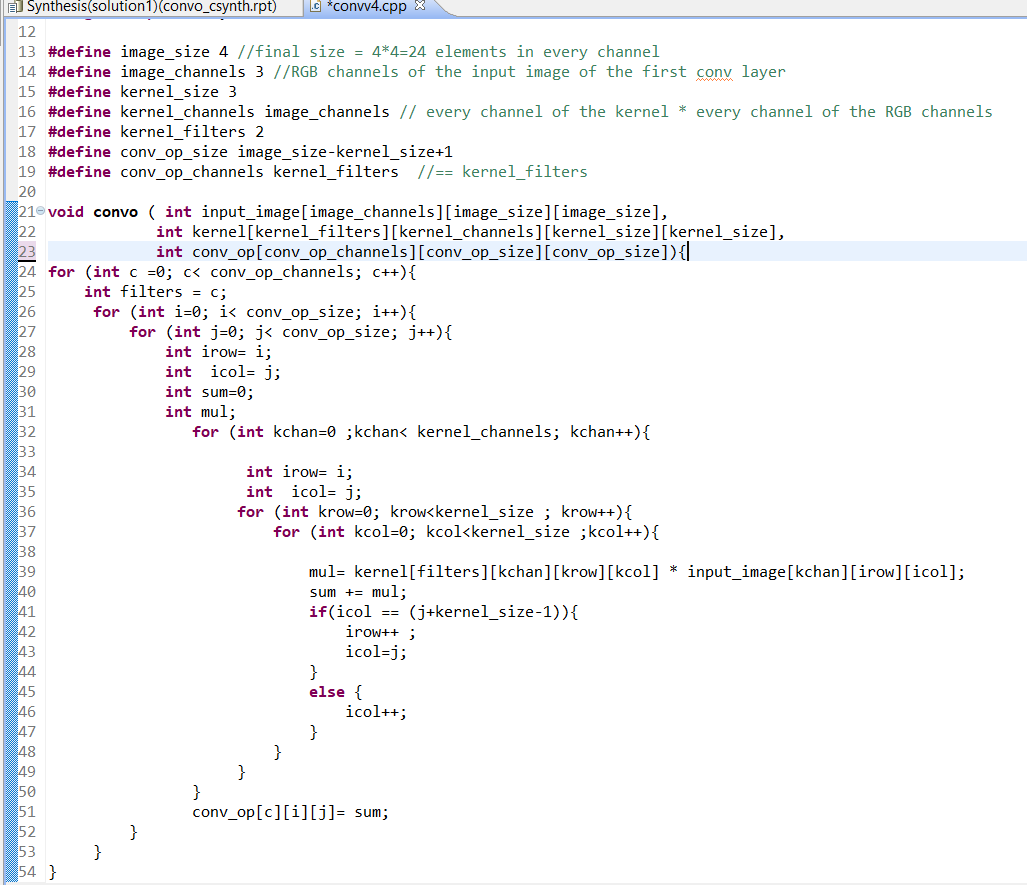
**Simulation report**



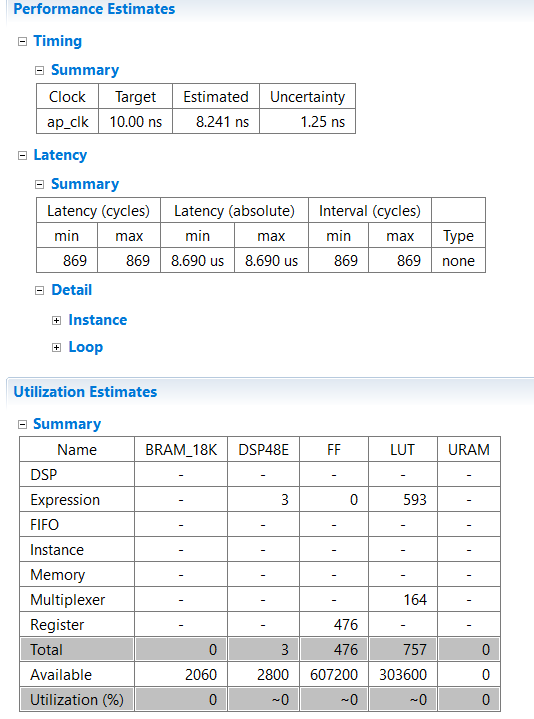
**Simulation screenshot**



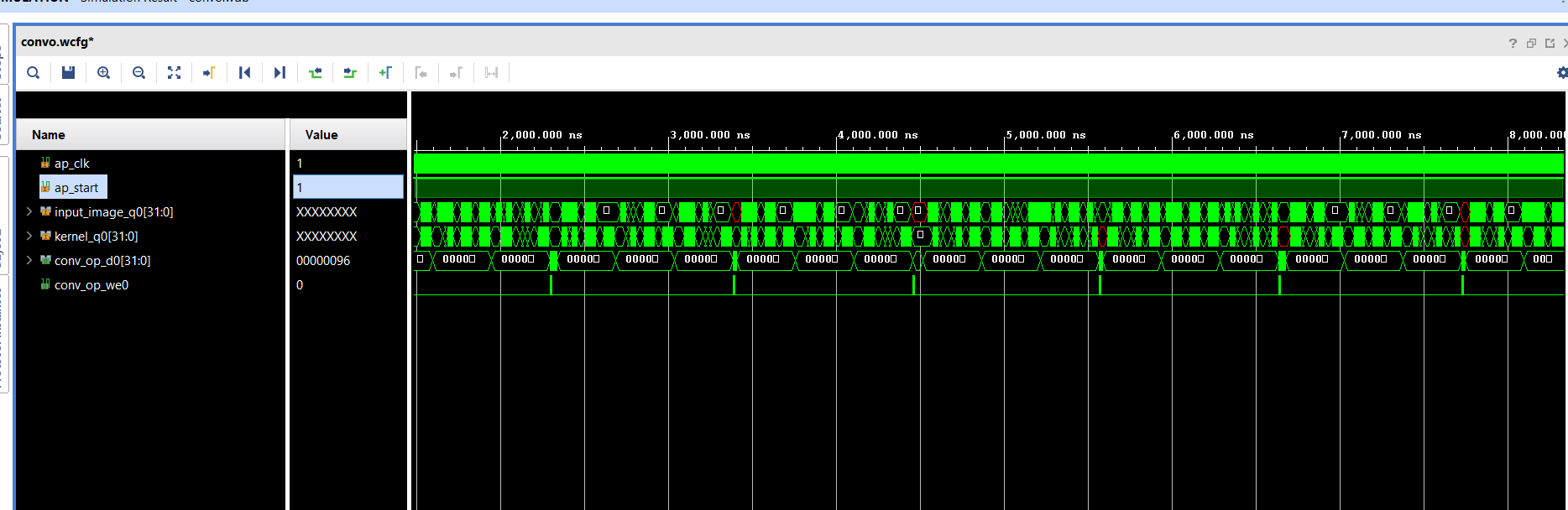
**Convolution (RGB Input images and kernel channels)**code



**Simulation report**

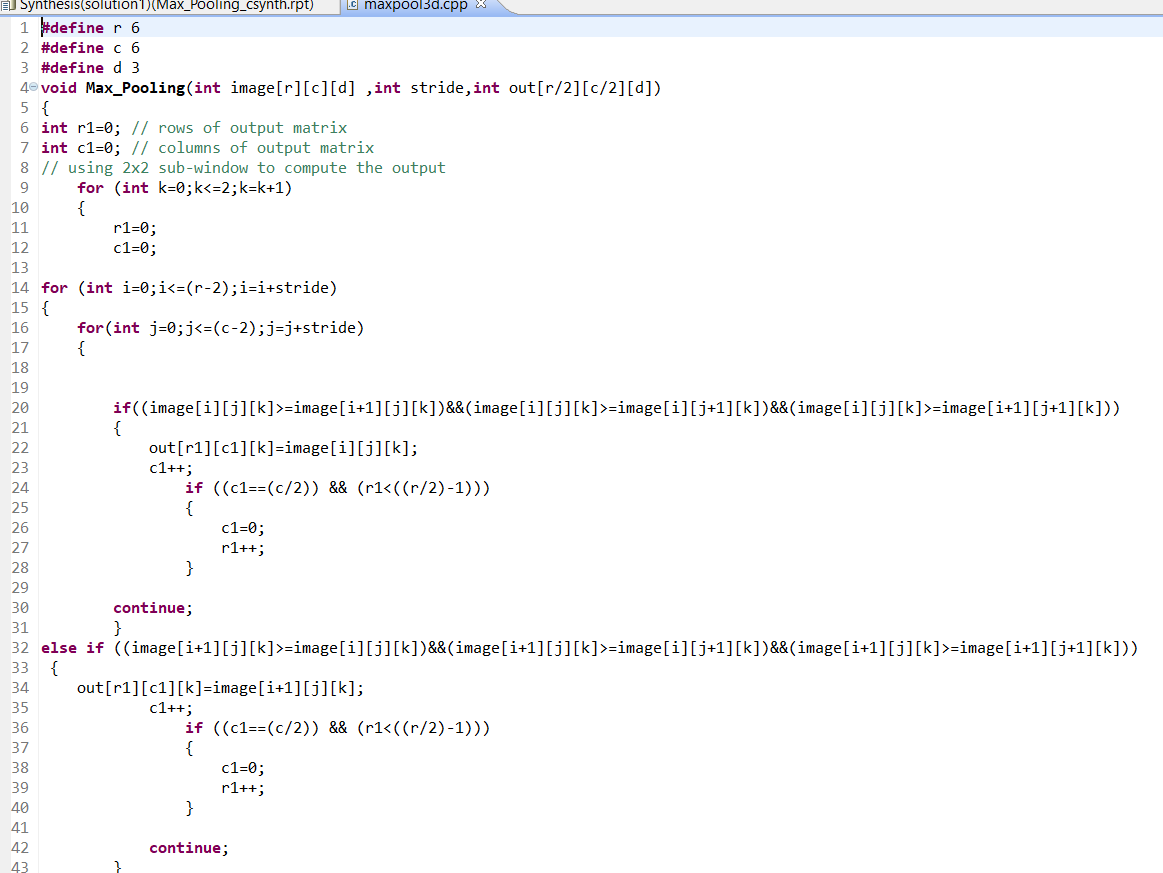


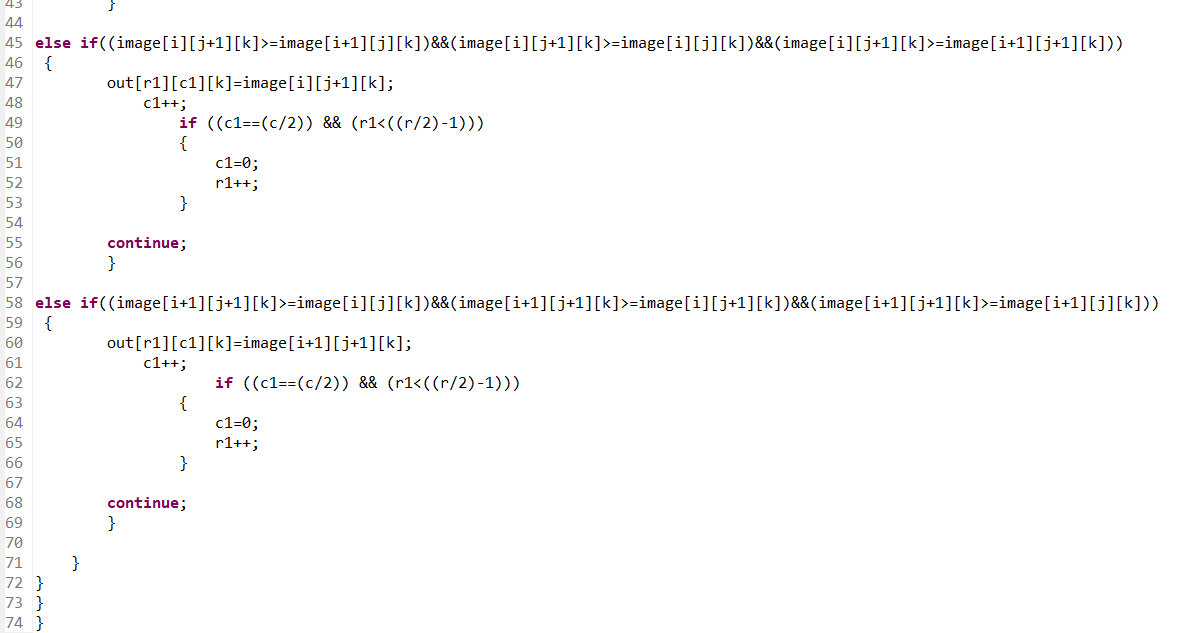
**Simulation screenshot**



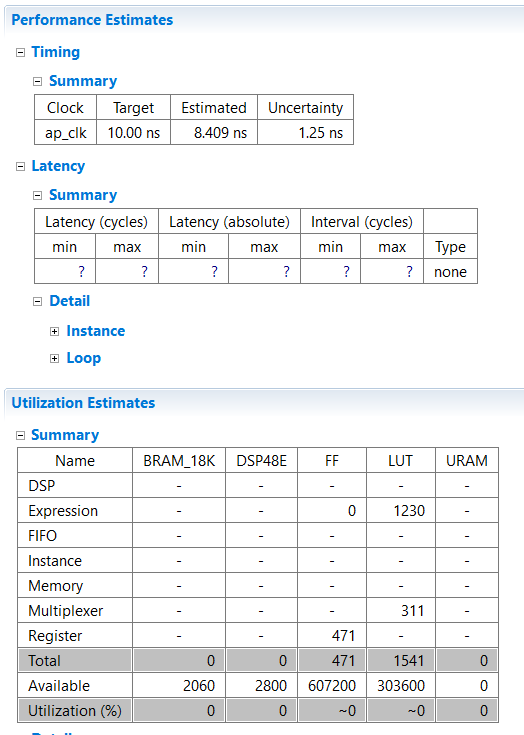
**Max pooling (for multiple feature maps )**

Code

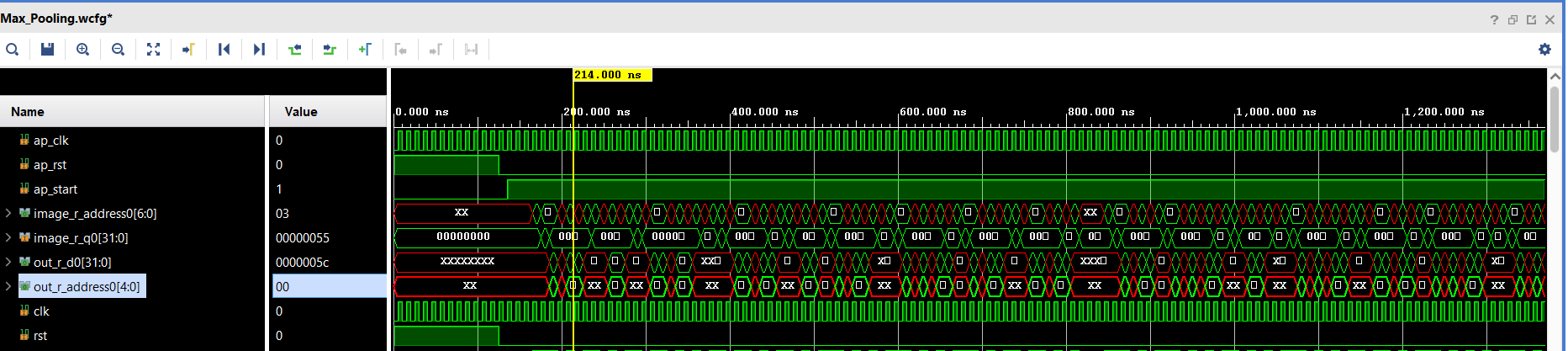




**Simulation report**

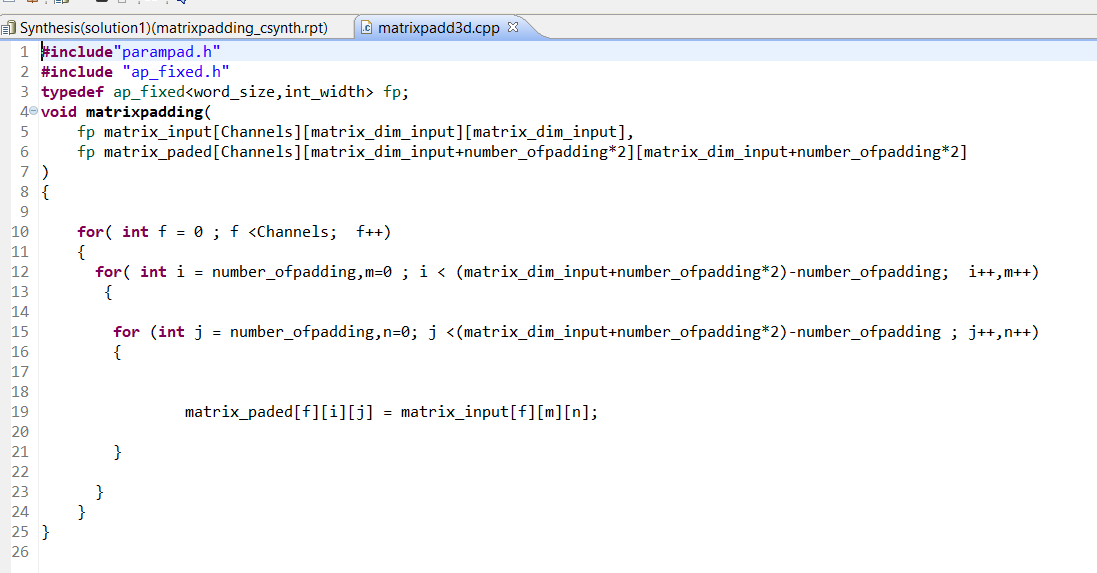


**Simulation waveform**

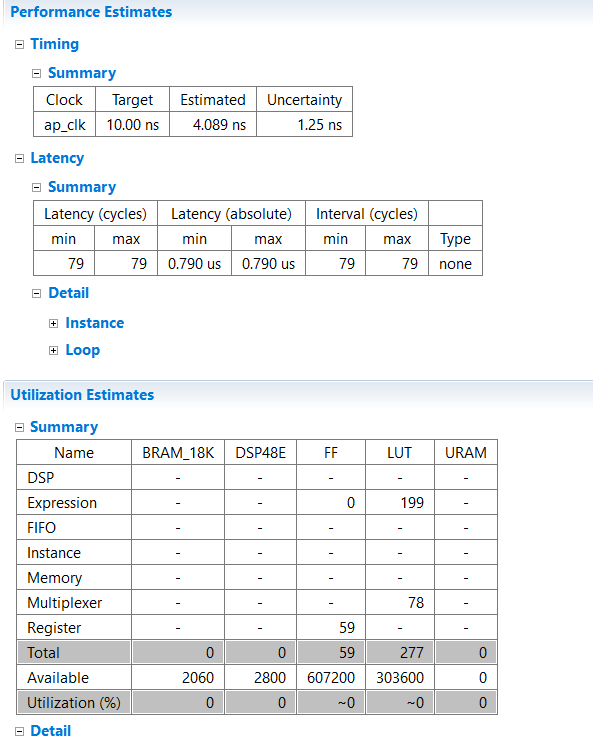


**Padding ( padding image with multiple channels )**

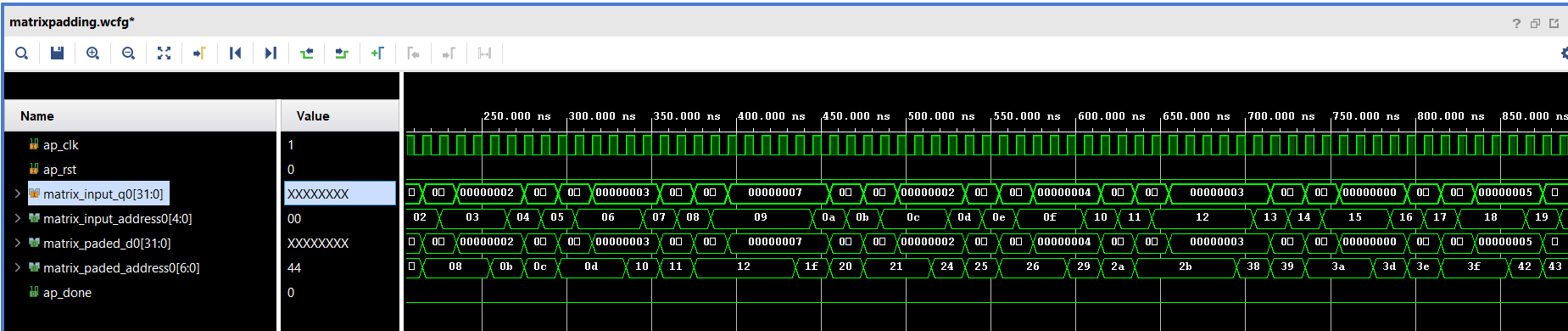
**Code**



**Code Report**



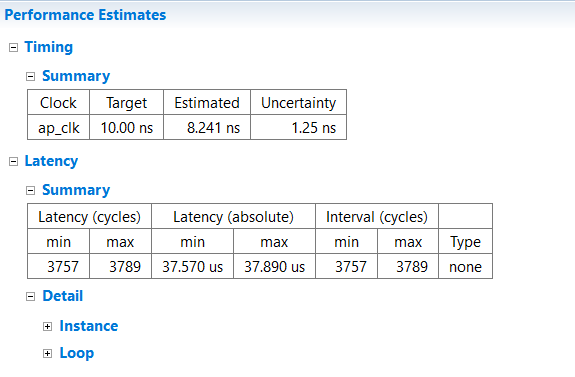
**Simulation waveform**

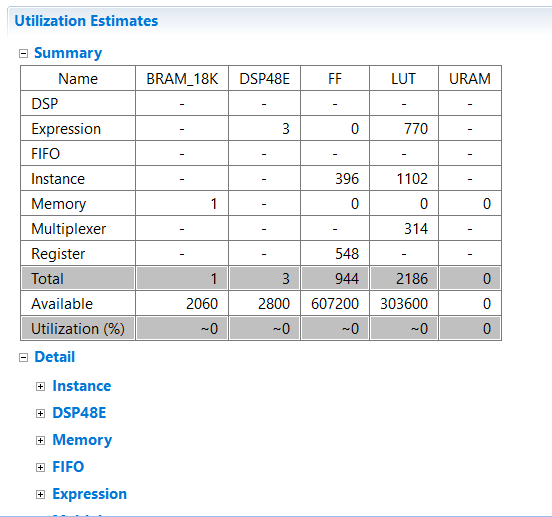


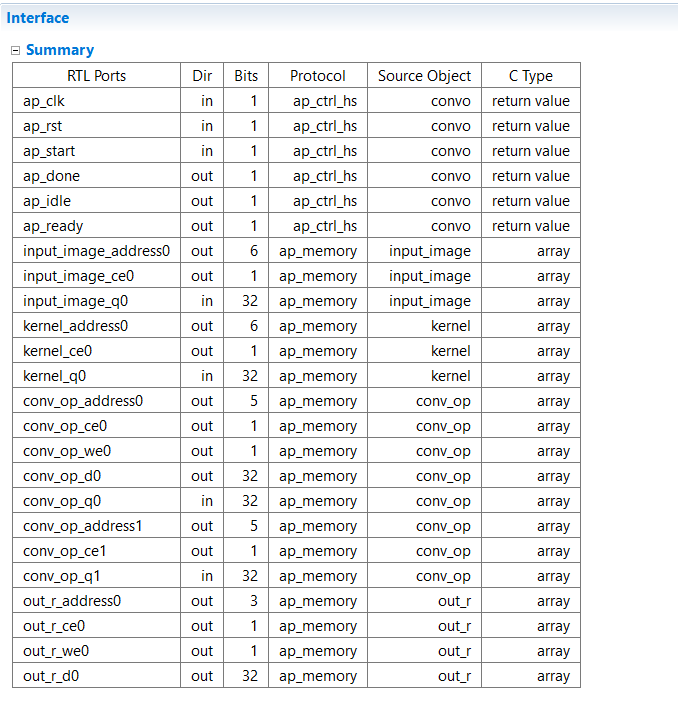
**Integration of (Padding-Convolution-Maxpooling)**

* Start the function by padding the input image so the feature map after convolution is same size if input image
* Then do the convolution with one image RGB ,2 filter with three channels each
* Then do max pooling 2\*2 which reduce the output feature map size by 2
* The code is the integrated parts of the 3 previous parts

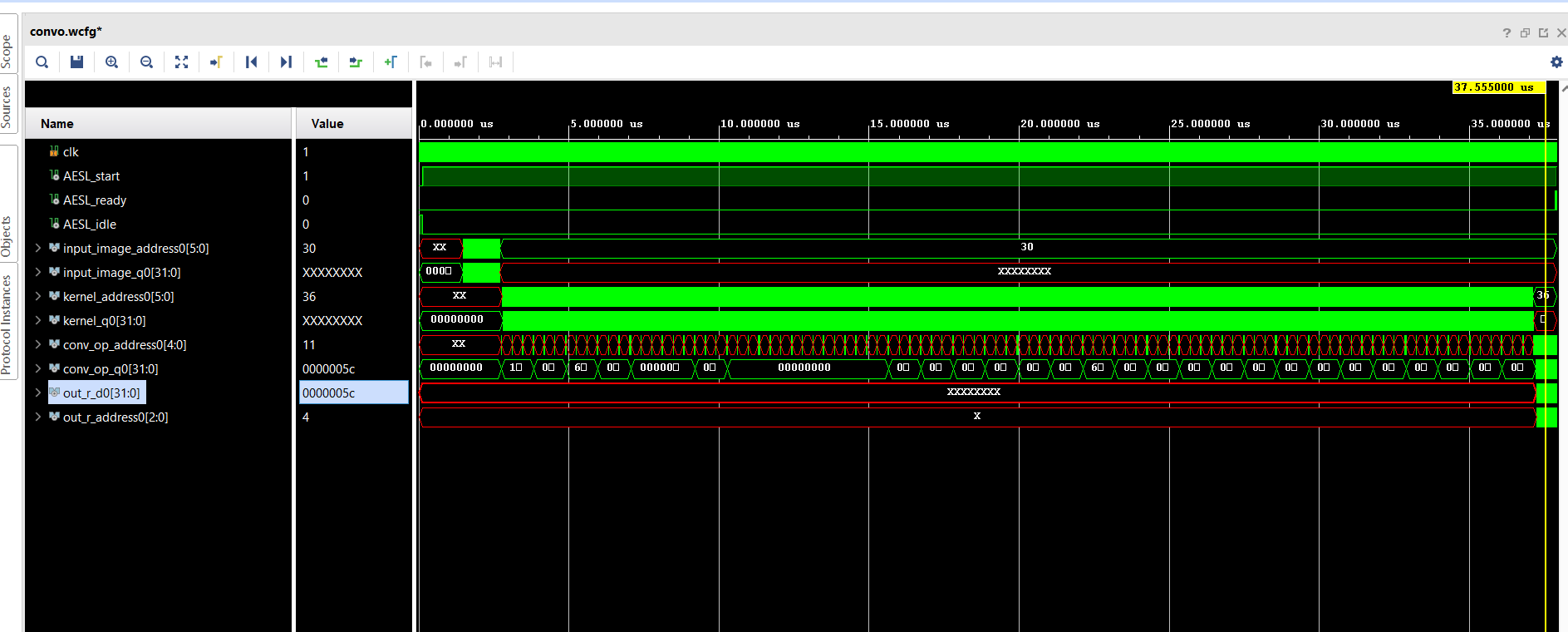
The simulation report







Simulation



We traced the waveform to ensure that the result and operation as expected