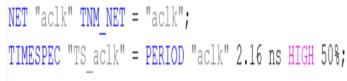
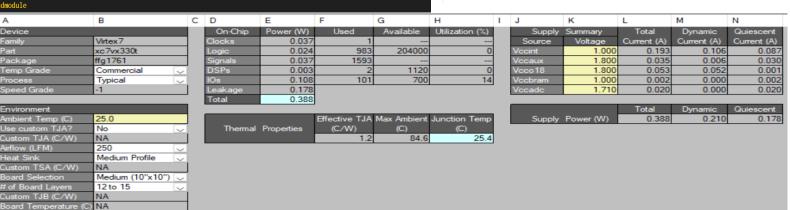
## **ASIC & FPGA HW(3)**

## Parham Gilani - 400101859

```
le alu IPCore
input wire aclk.
input wire [31:0] value b,
output wire [31:0] result,
output wire error,
output wire ready
wire [31:0] add_result, mult_result;
wire mult_ready, add_ready;
wire [1:0] mult_error, add_error;
assign error = (func == 2)? (mult_error != 0) : (add_error != 0);
assign ready = (func == 2)? mult_ready : add_ready;
    .s_axis_b_tready(), // output s_axis_b_tready
.s_axis_b_tdata(value_b), // input [31 : 0] s_axis_b_tdata
     . \verb|m_axis_result_tvalid| (\verb|mult_ready|), // \verb| output m_axis_result_tvalid|
     .s_axis_b_tready(), // output s_axis_b_tready
    .s_axis_operation_tvalid(1), // input s_axis_operation_tvalid
     .s_axis_operation_tready(), // output s_axis_operation_tready
     .s_axis_operation_tdata(func == 2), // input [7 : 0] s_axis_operation_tdata
      .m_axis_result_tdata(add_result), // output [31 : 0] m_axis_result_tdata
```

v1.0.2012-07-11





```
در این قسمت صرفا کد های ضرب و جمع
 dule alu (
  input wire aclk,
                                                                                                را دستی پیاده سازی کردم. پاور در در کل
  input wire [31:0] value a,
  input wire [31:0] value b,
                                                                                                  213mW است که Dvnamic 35mW
  input wire [1:0] func,
                            // 10 func is for multiply , 01 func is for subtraction , 00 func is for addition
  output wire [31:0] result.
                                                                                                      و Ouiescent 177mW است و در
  output wire error,
  output wire ready
                                                                                                   دمای 25.2 C کار میکند. منابع مصرف
                                                                                                   شده نیز 512 رجیستر و 200 LUT و 0
  wire [31:0] add_result, mult_result;
  wire mult_ready, add_ready;
                                                                                                 ممورى و IOB 101 و DSP48 2 است.
  wire [1:0] mult_error, add_error;
                                                                                                      ماکسیمم فرکانس کلاک برای این مدار
  assign result = (func == 2)? mult_result : add_result;
                                                                                                   تقريبا برابر 204MHz است. (4.9ns)
  assign error = (func == 2)? (mult_error != 0) : (add_error != 0);
  assign ready = (func == 2)? mult_ready : add_ready;
                                                                                                     كد قسمت اول را تست نكريم چون تابع
                                                                                                   اماده داشت ولی این کد تست شده است و
     .aclk(aclk),
     .a_operand(value_a),
                                                                                                 در ارور وقتی رخ میدهد که underflow
     .b_operand(value_b),
     .AddBar_Sub(func == 1),
                                                                                                    یا overflow یا exception رخ دهد.
     .result(add_result),
     .error(add error),
     .ready(add_ready)
                                                                                               NET "aclk" TNM NET = "aclk";
                                                                                               TIMESPEC "TS aclk" = PERIOD "aclk" 4.9 ns HIGH 50%;
                                                                                                    m>run -all
1 : 12153524 + c0895e81 = c0f6al7f :
     .aclk(aclk),
     .a_operand(value_a),
                                                                                                i = 2 : 8484d609 + b1f05663 = b1f05663 :
     .b_operand(value_b),
                                                                                                       : 06b97b0d + 46df998d = 46df998d :
      .error(mult_error),
                                                                                                        b2c28465 + 89375212 = b2c28465 :
     .result(mult_result),
                                                                                                        00f3e301 + 06d7cd0d = 06d7dc4b :
      .ready(mult_ready)
                                                                                                        e2f784c5 - d513d2aa = e2887b3b :
                                                                                                       : 72aff7e5 - bbd27277 = 72aff7e5 :
module Multiplication( // needs 6 clocks·
                                                                                                                                                              ready = 1
                                                                                                       : 8932d612 * 47ecdb8f = 91a576ba :
                                                                                                                                                              readv = 1
                                                                                                     2 : 793069f2 * e77696ce = ff800000 :
                                                                                                                                                              ready = 1
                                                                                                     3 : f4007ae8 * e2ca4ec5 = 7f800000 :
module Addition Subtraction( // needs 6 clocks...
                                                                                                     4 : 2e58495c * de8e28bd = cd703638 :
                                                                                                    5 : 96ab582d * b2a72665 = 09dfc070 : error = 0
                                                                                                ** Note: $stop : C:/Users/gilan/OneDrive/Desktop/alu_tb.v(38)
Time: 1500 ns | Iteration: 0 | Instance: /alu_tb
Break at C:/Users/gilan/OneDrive/Desktop/alu_tb.v line 38
odule priority_encoder(
                    В
                    Virtex7
                                                                                     204000
                                                              0.003
                    ffg1761
                                                              0.016
                     Typical
                     Medium Profile
                     Medium (10"x10")
                    12 to 15
                    v1.0,2012-07-11
```

## IP core Benefits:

- 1. Time saving: IP cores are pre-designed and pre-verified functional blocks that can be easily integrated into a larger design. This saves significant development time as engineers don't need to start from scratch for every component.
- 2. Cost effective: Developing custom hardware components from scratch can be expensive due to the resources required for design, verification, and testing. IP cores offer a cost-effective alternative by providing ready-made solutions that can be licensed or purchased for use.
- 3. Quality and reliability: IP cores are typically developed and tested by experienced teams or companies with expertise in the specific domain. This ensures high quality and reliability of the cores, reducing the risk of errors or failures in the final product.
- 4. Flexibility: IP cores can be easily customized and configured to meet the specific requirements of a design. This flexibility allows engineers to tailor the functionality, performance, and features of the cores to suit their application needs.
- 5. Interoperability: IP cores are often designed to industry standards and specifications, making them compatible and interoperable with other components and systems. This simplifies integration and reduces compatibility issues in complex designs.
- 6. Scalability: IP cores enable scalability in design by providing reusable building blocks that can be easily replicated or modified to support different configurations or performance levels. This facilitates the development of scalable and adaptable systems.
- 7. Focus on innovation: By leveraging pre-designed IP cores for standard functionalities, engineers can focus their efforts on developing innovative features and differentiation in their designs. This accelerates time-to-market and enhances competitiveness in the market.

8. Risk mitigation: Using proven and well-tested IP cores helps mitigate the risks associated with complex hardware design projects. By relying on established solutions, engineers can reduce the likelihood of design errors, delays, and cost overruns.

Overall, the use of IP cores offers significant advantages in terms of time, cost, quality, flexibility, interoperability, scalability, innovation, and risk mitigation, making them an essential component in modern digital design workflows.