# ASIC & FPGA HW(3)

## Parham Gilani – 400101859

```
module alu_IPCore (
    input wire aclk,
    input wire [31:0] value_a,
    input wire [31:0] value_b,
    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,
    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 result = (func == 2)? mult_result : add_result;
    assign error = (func == 2)? (mult_error != 0) : (add_error != 0);
    assign ready = (func == 2)? mult_ready : add_ready;

    Multiplication mult_core (
        .aclk(aclk), // input aclk
        .s_axis_a_tvalid(1), // input s_axis_a_tvalid
        .s_axis_a_tready(), // output s_axis_a_tready
        .s_axis_a_tdata(value_a), // input [31 : 0] s_axis_a_tdata
        .s_axis_b_tvalid(1), // input s_axis_b_tvalid
        .s_axis_b_tready(), // output s_axis_b_tready
        .s_axis_b_tdata(value_b), // input [31 : 0] s_axis_b_tdata
        .m_axis_result_tvalid(mult_ready), // output m_axis_result_tvalid
        .m_axis_result_tready(1), // input m_axis_result_tready
        .m_axis_result_tdata(mult_result), // output [31 : 0] m_axis_result_tdata
        .m_axis_result_tuser(mult_error) // output [1 : 0] m_axis_result_tuser
    );

    Addition_Subtraction add_core (
        .aclk(aclk), // input aclk
        .s_axis_a_tvalid(1), // input s_axis_a_tvalid
        .s_axis_a_tready(), // output s_axis_a_tready
        .s_axis_a_tdata(value_a), // input [31 : 0] s_axis_a_tdata
        .s_axis_b_tvalid(1), // input s_axis_b_tvalid
        .s_axis_b_tready(), // output s_axis_b_tready
        .s_axis_b_tdata(value_b), // input [31 : 0] s_axis_b_tdata
        .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_tvalid(add_ready), // output m_axis_result_tvalid
        .m_axis_result_tready(1), // input m_axis_result_tready
        .m_axis_result_tdata(add_result), // output [31 : 0] m_axis_result_tdata
        .m_axis_result_tuser(add_error) // output [1 : 0] m_axis_result_tuser
    );

endmodule
```

در این قسمت از IP Core جمع و ضرب کننده Floating Point استفاده کردم و مقادیر را به جفت جمع کننده و ضرب کننده دادم تا مقادیر را محاسبه کند و با توجه به نوع عملیات مقادیر خروجی را ست کردم.
برای پاور هم کل توان مصرف شده برابر 388mW است که 210mW ان Dynamic و 178mW ان Quiescent است و در دمای 25.4 C کار میکند. منابع مصرف شده برابر 1072 رجیستر و 983 تا LUT و 37 تا مموری و 1187 تا فلیپ فلاپ و 101 تا IOB و 2 تا DSP48 است. ماکسیمم فرکانس کلاک تست شده برای این مدار تقریبا برابر 463MHz است. (2.16ns)

```
NET "aclk" TNM_NET = "aclk";
TIMESPEC "TS_aclk" = PERIOD "aclk" 2.16 ns HIGH 50%;
```

| Device | |
|---|---|
| Family | Virtex7 |
| Part | xc7vx330t |
| Package | ffg1761 |
| Temp Grade | Commercial |
| Process | Typical |
| Speed Grade | -1 |

| Environment | |
|---|---|
| Ambient Temp (C) | 25.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 250 |
| Heat Sink | Medium Profile |
| Custom TSA (C/W) | NA |
| Board Selection | Medium (10"x10") |
| # of Board Layers | 12 to 15 |
| Custom TJB (C/W) | NA |
| Board Temperature (C) | NA |

| Characterization | |
|---|---|
| Production | v1.0,2012-07-11 |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.037 | 1 | --- | --- |
| Logic | 0.024 | 983 | 204000 | 0 |
| Signals | 0.037 | 1593 | --- | --- |
| DSPs | 0.003 | 2 | 1120 | 0 |
| IOs | 0.108 | 101 | 700 | 14 |
| Leakage | 0.178 | | | |
| Total | 0.388 | | | |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 1.2 | 84.6 | 25.4 |

| Supply Summary | | Total | Dynamic | Quiescent |
|---|---|---|---|---|
| Source | Voltage | Current (A) | Current (A) | Current (A) |
| Vccint | 1.000 | 0.193 | 0.106 | 0.087 |
| Vccaux | 1.800 | 0.035 | 0.006 | 0.030 |
| Vcco18 | 1.800 | 0.053 | 0.052 | 0.001 |
| Vccbram | 1.000 | 0.002 | 0.000 | 0.002 |
| Vccadc | 1.710 | 0.020 | 0.000 | 0.020 |

| | Total | Dynamic | Quiescent |
|---|---|---|---|
| Supply Power (W) | 0.388 | 0.210 | 0.178 |

در این قسمت صرفا کد های ضرب و جمع را دستی پیاده سازی کردم. پاور در در کل 213mW است که Dynamic 35mW و Quiescent 177mW است و در دمای 25.2 C کار میکند. منابع مصرف شده نیز 512 رجیستر و LUT 820 و 0 مموری و IOB 101 و DSP48 2 است. ماکسیمم فرکانس کلاک برای این مدار تقریبا برابر 204MHz است. (4.9ns) کد قسمت اول را تست نکردم چون تابع اماده داشت ولی این کد تست شده است و در ارور وقتی رخ میدهد که underflow یا overflow یا exception رخ دهد.

```
module alu (
    input wire aclk,
    input wire [31:0] value_a,
    input wire [31:0] value_b,
    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,
    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 result = (func == 2)? mult_result : add_result;
    assign error = (func == 2)? (mult_error != 0) : (add_error != 0);
    assign ready = (func == 2)? mult_ready : add_ready;

    Addition_Subtraction add_core(  // needs 6 clocks
        .aclk(aclk),
        .a_operand(value_a),
        .b_operand(value_b),
        .AddBar_Sub(func == 1),
        .result(add_result),
        .error(add_error),
        .ready(add_ready)
    );

    Multiplication mult_core(  // needs 6 clocks
        .aclk(aclk),
        .a_operand(value_a),
        .b_operand(value_b),
        .error(mult_error),
        .result(mult_result),
        .ready(mult_ready)
    );

endmodule

module Multiplication(  // needs 6 clocks···
endmodule

module Addition_Subtraction(  // needs 6 clocks···
endmodule

module priority_encoder(···
endmodule
```

```
NET "aclk" TNM_NET = "aclk";
TIMESPEC "TS_aclk" = PERIOD "aclk" 4.9 ns HIGH 50%;
```

```
ModelSim> run -all
# i = 1 : 12153524 + c0895e81 = c0f6a17f  :   error = 0   ,   ready = 1
#
# i = 2 : 8484d609 + b1f05663 = b1f05663  :   error = 0   ,   ready = 1
#
# i = 3 : 06b97b0d + 46df998d = 46df998d  :   error = 0   ,   ready = 1
#
# i = 4 : b2c28465 + 89375212 = b2c28465  :   error = 0   ,   ready = 1
#
# i = 5 : 00f3e301 + 06d7cd0d = 06d7dc4b  :   error = 0   ,   ready = 1
#
# i = 1 : 3b23f176 - 1e8dcd3d = 3b5c0e8a  :   error = 0   ,   ready = 1
#
# i = 2 : 76d457ed - 462df78c = 76aba813  :   error = 0   ,   ready = 1
#
# i = 3 : 7cfde9f9 - e33724c6 = 7cfde9f9  :   error = 0   ,   ready = 1
#
# i = 4 : e2f784c5 - d513d2aa = e2887b3b  :   error = 0   ,   ready = 1
#
# i = 5 : 72aff7e5 - bbd27277 = 72aff7e5  :   error = 0   ,   ready = 1
#
# i = 1 : 8932d612 * 47ecdb8f = 91a576ba  :   error = 0   ,   ready = 1
#
# i = 2 : 793069f2 * e77696ce = ff800000  :   error = 1   ,   ready = 1
#
# i = 3 : f4007ae8 * e2ca4ec5 = 7f800000  :   error = 1   ,   ready = 1
#
# i = 4 : 2e58495c * de8e28bd = cd703638  :   error = 0   ,   ready = 1
#
# i = 5 : 96ab582d * b2a72665 = 09dfc070  :   error = 0   ,   ready = 1
#
# ** 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
```

| Device | |
|---|---|
| Family | Virtex7 |
| Part | xc7vx330t |
| Package | ffg1761 |
| Temp Grade | Commercial |
| Process | Typical |
| Speed Grade | -1 |

| Environment | |
|---|---|
| Ambient Temp (C) | 25.0 |
| Use custom TJA? | No |
| Custom TJA (C/W) | NA |
| Airflow (LFM) | 250 |
| Heat Sink | Medium Profile |
| Custom TSA (C/W) | NA |
| Board Selection | Medium (10"x10") |
| # of Board Layers | 12 to 15 |
| Custom TJB (C/W) | NA |
| Board Temperature (C) | NA |

| Characterization | |
|---|---|
| Production | v1.0,2012-07-11 |

| On-Chip | Power (W) | Used | Available | Utilization (%) |
|---|---|---|---|---|
| Clocks | 0.009 | 1 | --- | --- |
| Logic | 0.003 | 820 | 204000 | 0 |
| Signals | 0.007 | 1305 | --- | --- |
| DSPs | 0.001 | 2 | 1120 | 0 |
| IOs | 0.016 | 101 | 700 | 14 |
| Leakage | 0.177 | | | |
| Total | 0.213 | | | |

| Thermal Properties | Effective TJA (C/W) | Max Ambient (C) | Junction Temp (C) |
|---|---|---|---|
| | 1.2 | 84.8 | 25.2 |

| Supply Source | Summary Voltage | Total Current (A) | Dynamic Current (A) | Quiescent Current (A) |
|---|---|---|---|---|
| Vccint | 1.000 | 0.107 | 0.022 | 0.086 |
| Vccaux | 1.800 | 0.030 | 0.001 | 0.030 |
| Vcco18 | 1.800 | 0.008 | 0.007 | 0.001 |
| Vccbram | 1.000 | 0.002 | 0.000 | 0.002 |
| Vccadc | 1.710 | 0.020 | 0.000 | 0.020 |

| Supply Power (W) | Total | Dynamic | Quiescent |
|---|---|---|---|
| | 0.213 | 0.035 | 0.177 |

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.