گزارش پروژه 3

CAD

امیرحسین علی خانی و صدرا مدائنی اول

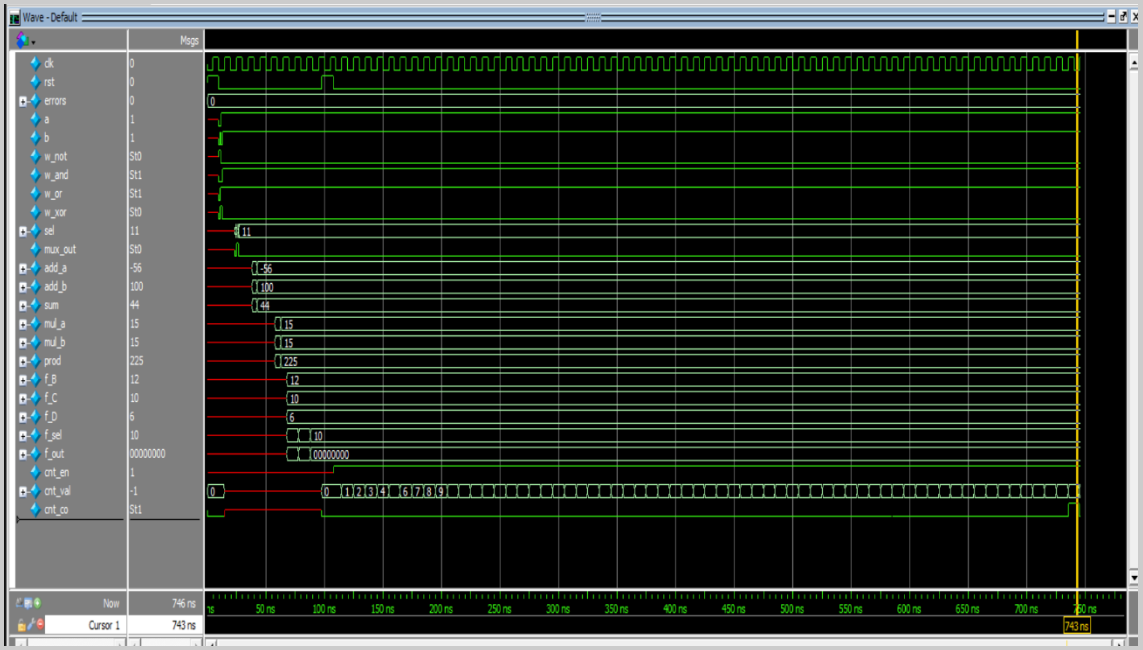
810102479 و 810102564

اولین کاری که کردیم تمام ماژول هایی که با استفاده از c1,c2,s1,s2 ساختیم رو در یک فایل قرار دادیم:

my_utiles.v > f

```
module my_not(input inp, output out);
    c1 inv (.A0(1'b1), .A1(1'b0), .SA(inp), .B0(1'b0), .B1(1'b0), .SB(1'b0), .S0(1'b0), .S1(1'b0), .f(out));
endmodule

module my_and(input a, input b, output out);
    c1 and_g (.A0(1'b0), .A1(b), .SA(a), .B0(1'b0), .B1(1'b0), .SB(1'b0), .S0(1'b0), .S1(1'b0), .f(out));
endmodule

module my_or(input a, input b, output out);
    c1 or_g (.A0(b), .A1(1'b1), .SA(a), .B0(1'b0), .B1(1'b0), .SB(1'b0), .S0(1'b0), .S1(1'b0), .f(out));
endmodule

module my_xor(input a, input b, output out);
    c2 xor_g (.D00(1'b0), .D01(1'b1), .D10(1'b1), .D11(1'b0), .A1(a), .B1(1'b0), .A0(b), .B0(1'b1), .out(out));
endmodule

module my_mux2(input s, input i0, input i1, output out);
    c1 mux (.A0(i0), .A1(i1), .SA(s), .B0(1'b0), .B1(1'b0), .SB(1'b0), .S0(1'b0), .S1(1'b0), .f(out));
endmodule

module my_mux4_one_cell(input [1:0] s, input i0, i1, i2, i3, output out);
    c1 mux4 (
        .S0(s[1]), .S1(1'b0),
        .SA(s[0]), .A0(i0), .A1(i1),
        .SB(s[0]), .B0(i2), .B1(i3),
        .f(out)
    );
endmodule

module half_adder(input a, input b, output sum, output cout);
    my_xor x1(a, b, sum);
    my_and a1(a, b, cout);
endmodule

module full_adder(input a, input b, input cin, output sum, output cout);
    wire not_cin;
    my_not n1(cin, not_cin);
```

برای تست کردنشون تست بنچی نوشتیم که اون ماژول هایی استفاده کردیم و برای صحت درستیش خروجی بگیره و نتیجه شد:



تصویر بالا خروجی شبیه‌سازی ماژول‌های پایه را نشان می‌دهد. همان‌طور که مشاهده می‌شود، سیگنال errors (خط سوم از بالا) در تمام طول شبیه‌سازی مقدار 0 را حفظ کرده است که نشان‌دهنده پاس شدن تمام تست‌هاست. در ادامه می پردازیم به دلیل اینکه چرا هرکدوم از اون سیگنال ها درست است.

1. (**my_adder8**):

در لحظه نشان‌داده شده، ورودی‌ها مقادیر add_a = -56 و add_b = 100 را دارند.خروجی sum مقدار **44** را نشان می‌دهد که نتیجه درسته.

2. (**my_multiplier**):

ورودی‌ها مقادیر ماکزیمم ۴ بیتی یعنی mul_a = 15 (1111) و mul_b = 15 (1111) را دارند.خروجی prod مقدار هگز 11100001 (معادل دسیمال **225**) را نشان می‌دهد

3. (**counter**):

سیگنال cnt_val به درستی با هر لبه کلاک افزایش می‌یابد (دنباله 0, 1, 2, 3, ... در پایین تصویر است).در انتهای شمارش، سیگنال cnt_co (Carry Out) به درستی فعال شده است.

4. **گیت‌های منطقی و مالتی‌پلکسر:**

تغییرات سیگنال‌های w_and, w_or, w_xor و mux_out متناظر با تغییرات ورودی‌ها بوده و تاخیر گیت‌ها در حد نرمال است.

در این پروژه ما میخواستیم یک تولیدکننده هش (Hash Generator) در سطح RTL و به صورت تماماً (Structural) بود.

تمام گیت‌های منطقی (AND, OR, NOT, MUX, XOR) با استفاده از سلول‌های C1 و C2 ساخته شدند.

برای کاهش هزینه، تا حد امکان از سلول C1 (که ۱۰ گیت است) برای پیاده‌سازی MUX و NOT و AND/OR استفاده شد.

به جای استفاده از Full Adder معمولی (۵۵ گیت)، از یک طراحی فشرده مبتنی بر LUT در سلول C2 استفاده شد که هزینه هر بیت را به ۳۲ گیت کاهش داد.

برای جایگزینی عملیات چرخش (Rotate)، یک ضرب‌کننده ۴x۴ طراحی شد.از سلول S2 برای ساخت رجیسترها استفاده شد.

در طراحی‌های معمول، برای انتخاب ورودی رجیستر از یک MUX جداگانه استفاده می‌شود. در این طراحی، با توجه به اینکه سلول S2 دارای پایه‌های انتخاب S0 و S1 است، لاجیک انتخاب ورودی (بین مقدار اولیه و مقدار حلقه) به درون خود رجیستر منتقل شد که باعث شد حدود ۳۲۰ گیت صرفه جویی بشه.

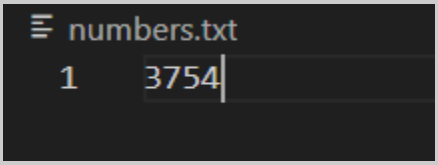
از آنجا که ورودی‌های ضرب‌کننده دارای شیفت‌های مشخصی بودند (ضرب در بیت‌های خاص)، بسیاری از عملیات جمع عملاً جمع با صفر بودند. پس به جای استفاده از ۳ جمع‌کننده کامل ۸ بیتی، یک مدار جمع‌کننده اختصاصی طراحی شد که گیت‌های مربوط به جمع صفرها را حذف کرد.در طبقات اول که کری ورودی وجود نداشت، از Half Adder (۲۲ گیت) به جای Full Adder (۳۲ گیت) استفاده شد.

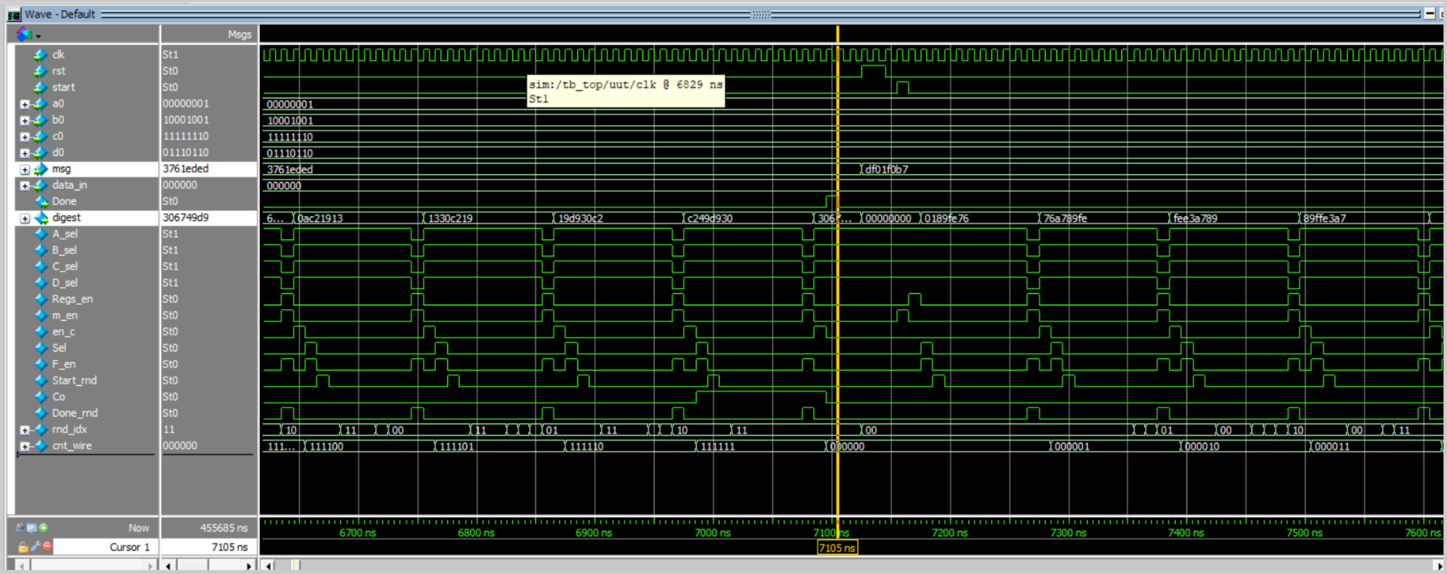
با توجه به اینکه پیام ورودی (msg) در طول پردازش ثابت است، رجیسترهای داخلی پیام حذف شدند و ورودی‌ها مستقیماً به مالتی‌پلکسر انتخاب پیام متصل شدند. یعنی صرفه‌جویی حدود **۴۱۶ گیت**.

تابع F با استفاده از ساختار مالتی‌پلکسری (If-Else) و با سلول ارزان C1 پیاده‌سازی شدند که بسیار کم‌هزینه‌تر از پیاده‌سازی با گیت‌های جداگانه بود.

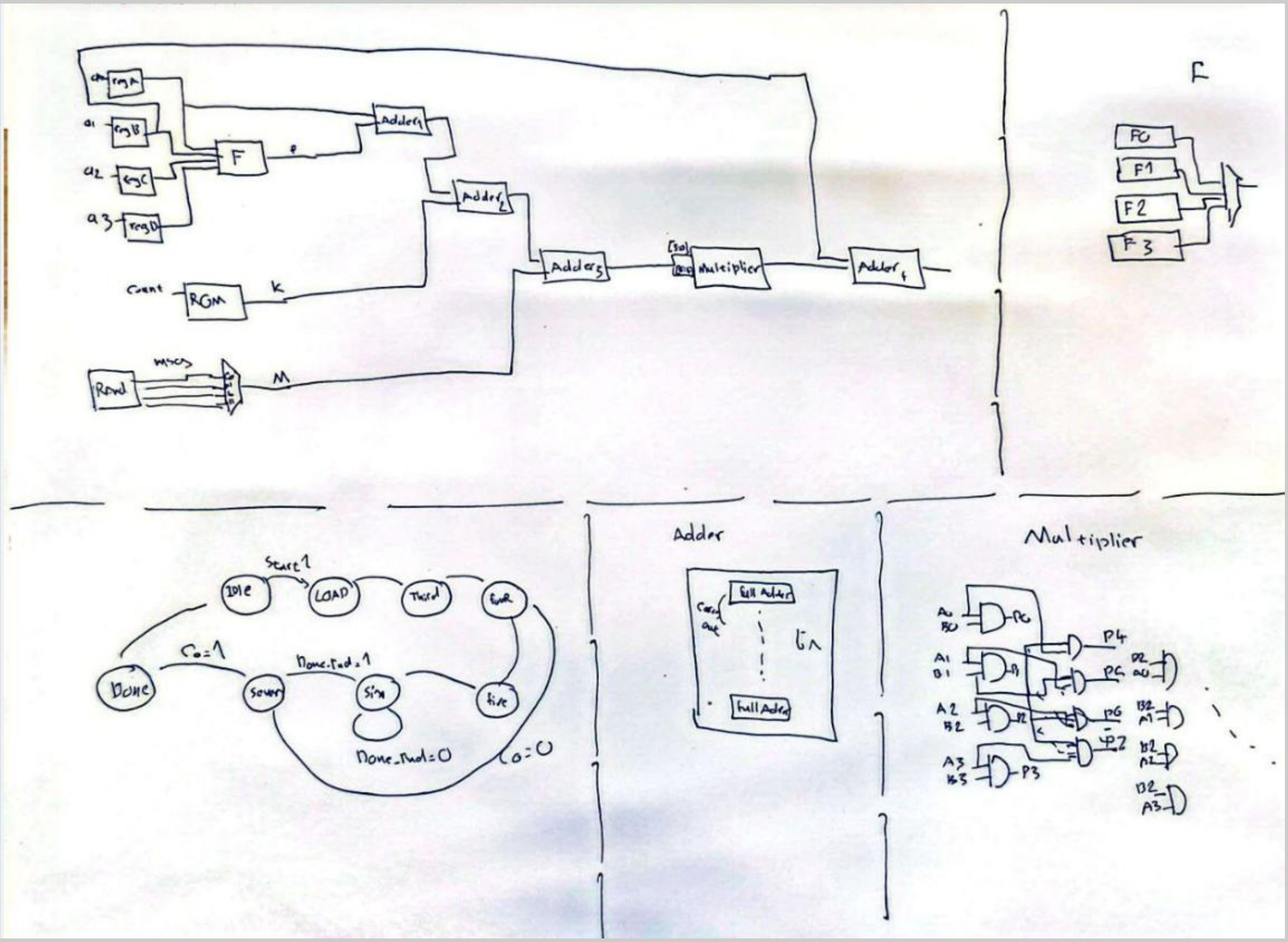
در نهایت با چک کردن خروجی تعداد گیت ها شد 3754 و تمام تست کیس ها درست بود.

خروجی:




Wave - Default
Msgs
clk
rst
start
a0
b0
c0
d0
msg
data_in
Done
digest
A_sel
B_sel
C_sel
D_sel
Regs_en
m_en
en_c
Sel
F_en
Start_rnd
Co
Done_rnd
rnd_idx
cnt_wire
sim:/tb_top/uut/clk @ 6829 ns
St1
Now
455685 ns
Cursor 1
7105 ns

طراحی ما:


F
Adder
Multiplier
RGM
Rnd
F0
F1
F2
F3
Idle
LOAD
Start 1
Done
full Adder

به منظور کاهش تعداد گیت‌ها به ۳۷۵۴، مالتی‌پلکسرهای ورودیِ رجیسترهای A، B، C و D حذف شده و عملکرد آن‌ها با استفاده از پایه انتخاب S0 در سلول S2، به صورت داخلی پیاده‌سازی شده است (منطق ادغام‌شده‌ی مالتی‌پلکسر و رجیستر).