به نام خدا

عنوان

پروژه جبرانی امتحان پایانترم درس طراحی سیستم‌های دیجیتال

سوال هشتم میانترم

استاد

دکتر فصحتی

نام و شماره دانشجویی

محمد عرفان محمودی ۴۰۱۱۰۹۹۲۵

الف) ماژول طراحی شده در ادامه قرار داده خواهد شد. صرفا برخی از فرضیات مورد استفاده برای این سوال را ذکر می‌کنیم.

- ظرفیت پارکینگ ۷۰۰ خودرو است.
- هنگامی که جمع ظرفیت اعضای دانشگاه با ظرفیت عموم بیش از ۷۰۰ شد (مثلا در ساعت ۱۶ این مقدار برابر ۱۰۰۰ است) ؛ آنگاه دقت می‌کنیم که تعداد پارک شده های هرکدام از ظرفیتش کمتر باشد و همچنین از ظرفیت کل پارکینگ (۷۰۰ منهای تعداد کل پارک شده ها) هم کمتر باشد . بنابراین مینیمم میگیریم!
- در صورتی که ظرفیت برای گروهی ۰ شود؛ امکان وارد شدن ماشینی از آن گروه وجود ندارد. به عبارتی ماشین امکان ورود دارد اگر و تنها اگر ظرفیت گروه آن بزرگتر از صفر باشد.
- ماژول طراحی شده علاوه بر ورودی خروجی های خواسته شده، خروجی hour نیز دارد که نشان می‌دهد که ساعت چند است!
- خروجی‌های عددی ماژول از جنس integer هستند که می‌دانیم اعداد ۳۲ بیتی اند و خروجی های سیگنالی از جنس reg هستند.
- ورودی های ماژول به صورت آسنکرون عمل می‌کنند و هرلحظه می‌شود یک ماشین وارد یا خارج شود؛ بنابراین، این ماژول را با استفاده از ۴ عدد always می‌سازیم که در ادامه توضیح می دهیم.
- همچنین این ماژول تحت عنوان "CU.v" پیوست شده است.
- در صفحات همراه با عکس، ماژول را توصیف خواهیم کرد:

```
module CU (
    input clk, car_entered, is_uni_car_entered, car_exited, is_uni_car_exited,
    output integer uni_parked_car = 0, parked_car = 0, uni_vacated_space = 0, vacated_space = 0, hour = 8;
    output reg uni_is_vacated_space, is_vacated_space
);

    integer second_counter = 0;
    integer max_uni_vacated_space = 500;
    integer max_vacated_space = 200;
    integer parking_space = 700;

    integer entered_car = 0;
    integer exited_car = 0;
    integer uni_entered_car = 0;
    integer uni_exited_car = 0;

    always @(posedge clk) begin
        if (second_counter == 3600) begin
            second_counter = 0;
            hour = hour + 1;
        end
        second_counter = second_counter + 1;
    end

    always @(hour) begin
        if (hour >= 16)
            max_vacated_space = max_vacated_space + 150;
        else if (hour >= 13)
            max_vacated_space = max_vacated_space + 50;
    end
```

در این قسمت، ورودی و خروجی ها و دو always اول را مشاهده می‌کنید؛ همچنین تعدادی integer جدید برحسب نیاز تعریف شده اند.

Second_counter ، شمارنده ثانیه است و در always اول زیاد می‌شود و هر کلاک را یک ثانیه درنظر می‌گیریم. درنتیجه که هرموقع به ۳۶۰۰ رسید، یک ساعت جلو برویم.

۳ عدد بعدی، حداکثر ظرفیت ممکن برای هر بخش پارکینگ و کل پارکینگ هستند که مقداردهی اولیه (بر اساس ساعت اولیهٔ ۸) شده اند.

۴ عدد بعدی، تعداد ماشین های وارد شده و خارج شده مربوط به هر بخش است؛ این بخش به دلیل مشکل خوردن در سنتز ایجاد شده است؛ در بخش بعدی توضیحات کامل تری در این باره خواهیم داشت.

در always دوم ، با توجه به گفته های سوال، ظرفیت های حداکثری، با توجه به ساعت عوض می‌شوند و این always به تغییرات ساعت حساس است.

```
32      always @(posedge car_entered) begin
33          if (is_uni_car_entered) begin
34              if (uni_vacated_space > 0)
35                  uni_entered_car = uni_entered_car + 1;
36          end
37
38          else begin
39              if (vacated_space > 0)
40                  entered_car = entered_car + 1;
41          end
42      end
43
44      always @(posedge car_exited) begin
45          if (is_uni_car_exited) begin
46              if (uni_parked_car > 0)
47                  uni_exited_car = uni_exited_car + 1;
48          end
49          else begin
50              if (parked_car > 0)
51                  exited_car = exited_car + 1;
52          end
53      end
```

این ۲ بخش مربوط به دریافت اطلاعات ورود و خروج ماشین ها است و از آن ۴ عدد تعریف شده استفاده می‌کند؛ من در ابتدا فقط uni_parked_car و parked_car را به روزرسانی می‌کردم ولی از آنجایی که مقداردهی به یک متغیر در دو always همزمان، میسر نبود، برای هرکدام، دو متغیر جدید تعداد ورودی و خروجی را تعریف کردم.

وجود داشتن دو always جدا از always مربوط به زمان، آسنکرون بودن مدار نسبت به ورود و خروج ماشیت ها را نشان می‌دهد و بنابراین در تست‌بنچ بدون توجه به زمان و فقط با بالا رفتن هر سیگنال، ماشین ها وارد و خارج می‌شوند.

```
55      always @(*) begin
56          parking_space = 700 - parked_car - uni_parked_car;
57          uni_parked_car = uni_entered_car - uni_exited_car;
58          parked_car = entered_car - exited_car;
59
60          if (max_uni_vacated_space - uni_parked_car < parking_space)
61              uni_vacated_space = max_uni_vacated_space - uni_parked_car;
62          else uni_vacated_space = parking_space;
63
64          if (max_vacated_space - parked_car < parking_space)
65              vacated_space = max_vacated_space - parked_car;
66          else vacated_space = parking_space;
67
68          uni_is_vacated_space <= uni_vacated_space > 0;
69          is_vacated_space <= vacated_space > 0;
70      end
71
72  endmodule
```

در نهایت، در این always نهایی که با تغییر هرکدام از مقادیری که در always قبلی تعیین می‌شدند، شروع به‌کار می‌کند؛ ابتدا کل فضای خالی پارکینگ را محاسبه می‌کند.

سپس تعداد ماشین های پارک شده هر بخش را با توجه به تعداد ورودی و خروجی محاسبه می‌کند.

درنهایت فضای خالی مربوط به هر بخش را (که برابر ماکسیمم ظرفیت آن بخش منهای تعداد ماشین های پارک شده آن بخش است) محاسبه می کند و اگر از کل فضای پارکینگ کمتر بود؛ به جای آن، کل فضای پارکینگ را قرار می‌دهد.

- حال برای تست گرفتن از پارکینگمان، فایل "TB.v" را که در پروژه هم پیوست شده است را اجرا می‌کنیم.
در هربخش به طور کامل کامنت‌گذاری شده است که هدف تست آن بخش و نتیجه مورد انتظار را می گوید؛ همچنین هر بخش را به اختصار در ادامه توضیح خواهیم داد؛ با توجه به انتظارات؛ همه چیز درست است؛ ظرفیت ها با توجه به ساعت تغییر می‌کنند و هرگاه ظرفیت هر بخش پر می‌شده؛ دیگر ماشینی اجازه ورود پیدا نمی‌کرد.
همچنین در بخش های ۱۴ و ۱۵ دیگر کل پارکینگ پر می‌شود؛ علارغم اینکه تعداد ماشین های پارک شده هرکدام هنوز کمتر از ظرفیت است؛ چون تعداد کل پارک شده ها به ۷۰۰ رسیده است.

به ترتیب هر بخش را توصیف می‌کنیم:

۱) مقدار اولیه

۲) اضافه کردن یک دانشجو

۳) اضافه کردن یک انسان عادی

۴) اضافه کردن و حذف کردن یک انسان عادی به صورت همزمان

۵) اضافه کردن و حذف کردن یک دانشجو به صورت همزمان

۶) اضافه کردن یک دانشجو و حذف یک انسان عادی

۷) اضافه کردن یک انسان عادی و حذف یک دانشجو

۸) اضافه کردن ۲۹۹ دانشجو و رسیده تعداد دانشجو های پارک شده به ۳۰۰

۹) اضافه کردن ۱۹۹ انسان عادی و پر شدن ظرفیت. همانطور که می‌بینیم خروجی فضای خالی عادی ۰ شده است.

۱۰)اضافه کردن یک انسان عادی دیگر. همانطور که انتظار می‌رفت، ماشینی به پارکینگ اضافه نشد!

۱۱)منتظر ماندن تا زمانی که ساعت ۱۳ شود و بنابراین فضای خالی عادی به ۵۰ برسد.

۱۲)اضافه کردن یک انسان عادی دیگر. همانطور که انتظار می‌رفت، این بار اضافه شد!

۱۳)منتظر ماندن تا زمانی که ساعت ۱۶ شود و بنابراین فضای خالی حداکثری عادی به ۵۰۰ برسد که چون تابحال ۲۰۱ نفر آمده اند باید ۲۹۹ شود؛ اما چون کل طرفیت خالی برابر ۱۹۹ است؛ پس باید ۱۹۹ را (که مینیمم آن هاست) ببینیم که می‌بینیم!

۱۴)اضافه شدن ۱۹۹ ماشین عادی جدید؛ بنابراین ظرفیت کل پارکینگ پر می‌شود و تعداد پارک شده های عادی ۴۰۰ و دانشجویی همان ۳۰۰ است.

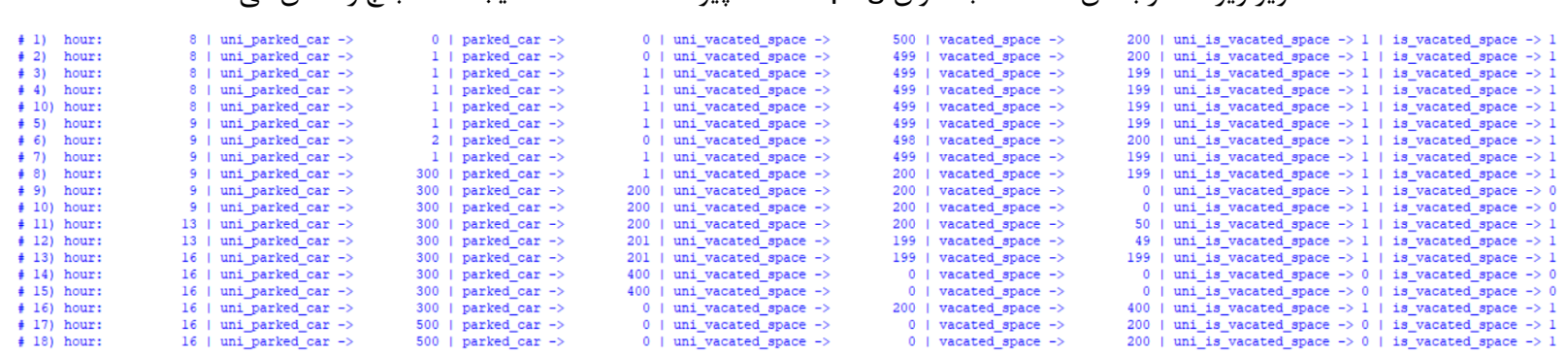
۱۵)اضافه کردن یک انسان عادی دیگر. همانطور که انتظار می‌رفت، ماشینی به پارکینگ اضافه نشد!

۱۶)خارج شدن ۳۰۰ ماشین عادی؛ بنابراین تعدادشان به ۱۰۰ می‌رسد و حالا همه ۵۰۰ ظرفیت دانشجویی قابل استفاده اند که ۳۰۰ تا تابحال پرشده پس فضای خالی باید ۲۰۰ باشد.

۱۷)ورود ۲۰۰ دانشجو و پر شدن ظرفیت دانشجویی.

۱۸)اضافه کردن یک دانشجوی دیگر. همانطور که انتظار می‌رفت، ماشینی به پارکینگ اضافه نشد!

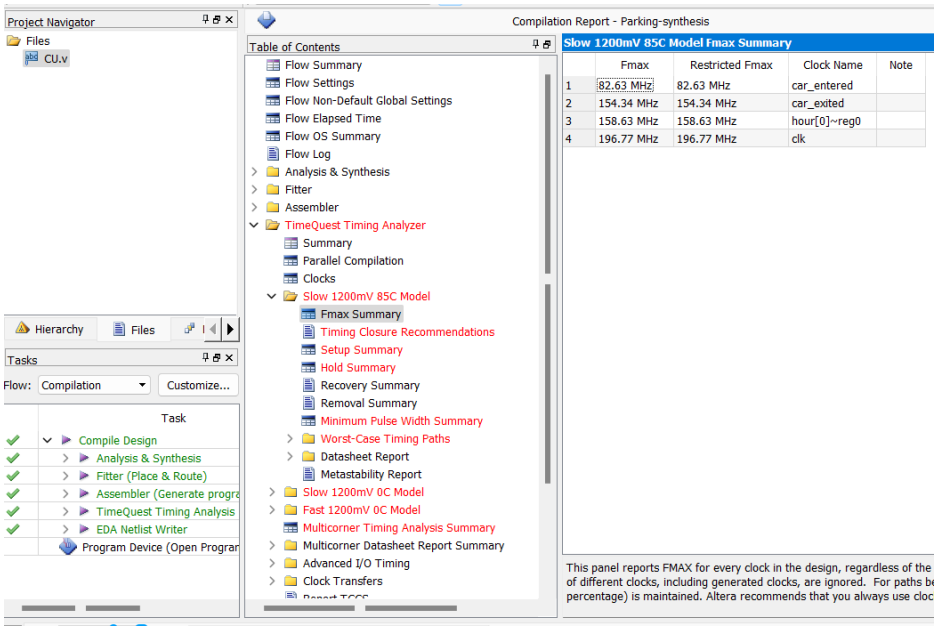
تصویر زیر که در بخش assets با عنوان result.png پیوست شده است؛ نتیجه تست‌بنچ را نشان می‌دهد:



ب)

برای سنتز کد وریلاگ زده شده، از نرم‌افزار کوارتوس و Cyclone II به عنوان FPGA خود استفاده می‌کنیم.

ابتدا فایل وریلاگ CU.v را به پروژه کوارتوس خود اضافه کرده و پس از انتخاب آن به عنوان top level entity پروژه را کامپایل می‌کنیم. برای یافتن گزارشات مربوط به فرکانس و زمانبندی، پس از کامپایل، از پوشه TimeQuestTimingAnalyser، گزینه Fmax summary را باز می‌کنیم.



بیشترین فرکانس برای هر کدام از ورودی‌های مدار ما مطابق جدول بالا است. اما با توجه به اینکه ورودی‌های ما آسنکرون هستند، تعیین یک فرکانس کلی بیشینه برای مدار معنای خاصی ندارد. اما اگر بخواهیم فرکانس بیشینه مشترکی که هیچ ورودی‌ای مشکل‌دار نشود اخذ کنیم، جواب 82.63 خواهد بود.

مشاهده می‌شود که فرکانس ورود با اختلاف زیادی از سایرین کمتر است و گلوگاهی برای بالابردن فرکانس کل می‌باشد. حال برای فهمیدن دلیل این موضوع بررسی می‌کنیم که آیا این اختلاف فرکانس بخاطر ستاپ تایم است یا هلد تایم. با توجه به عکس زیر این اختلاف فرکانس از ستاپ تایم ناشی می‌شود. حال علت آن را بررسی می‌کنیم.

Setup Summary

	Clock	Slack	End Point TNS
1	car_entered	-20.873	-1269.973
2	clk	-11.209	-509.366
3	car_exited	-8.910	-532.422

علت این موضوع به انجام تغییرات موجود در always مربوط به ورودی است. اگر دقت کنیم در always مربوط به ورود خودرو، پس از یک مقایسه با صفر (که نسبتا طولانی است) تغییراتی در تعداد ماشین های پارک شده رخ می دهد و باعث تحریک always آخر می‌شود که این هم با توجه به اینکه تعداد ماشین های پارک شده ممکن است زیاد شده باشد باید ابتدا تعداد پارک شده هر بخش را محاسبه کند و سپس مینیمم بگیرد؛ درحالی که اگر کم شده باشد (خروج ماشین)؛ دیگر نیازی به مینیمم گرفتن برای بقیه وجود ندارد و همان ظرفیت قبلی را صرفا یکی زیاد می‌کند.