**به نام خدا**



**عنوان**

پروژه جبرانی امتحان پایانترم درس طراحی سیستم‌های دیجیتال

سوال هشتم میانترم

**استاد**

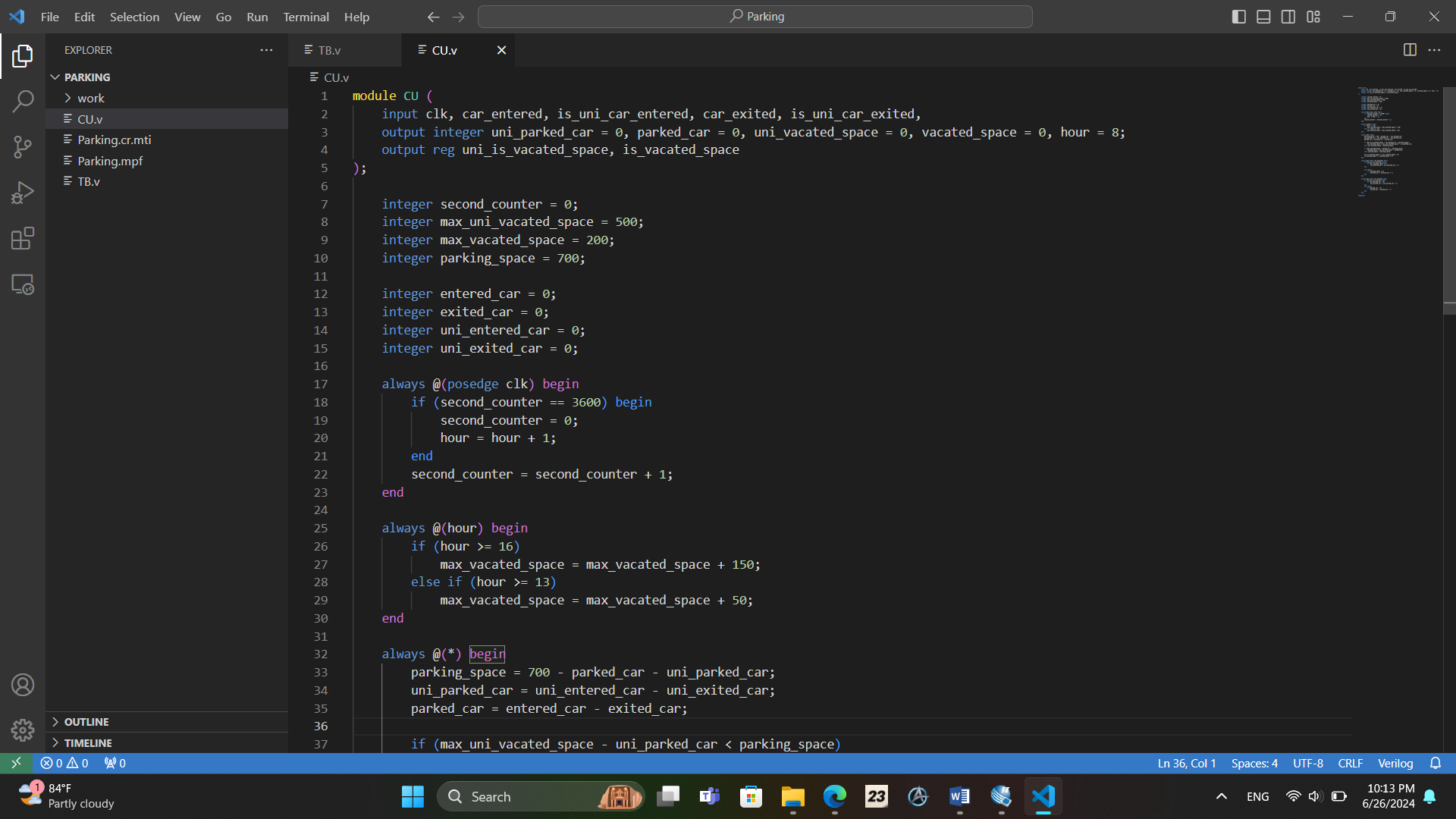
دکتر فصحتی

**نام و شماره دانشجویی**

محمد عرفان محمودی 401109925

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

* ظرفیت پارکینگ 700 خودرو است.
* هنگامی که جمع ظرفیت اعضای دانشگاه با ظرفیت عموم بیش از 700 شد ( مثلا در ساعت 16 این مقدار برابر 1000 است ) ؛ آنگاه دقت می‌کنیم که تعداد پارک شده های هرکدام از ظرفیتش کمتر باشد و همچنین از ظرفیت کل پارکینگ ( 700 منهای تعداد کل پارک شده ها ) هم کمتر باشد . بنابراین مینیمم میگیریم!
* در صورتی که ظرفیت برای گروهی 0 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 32 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 4 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:



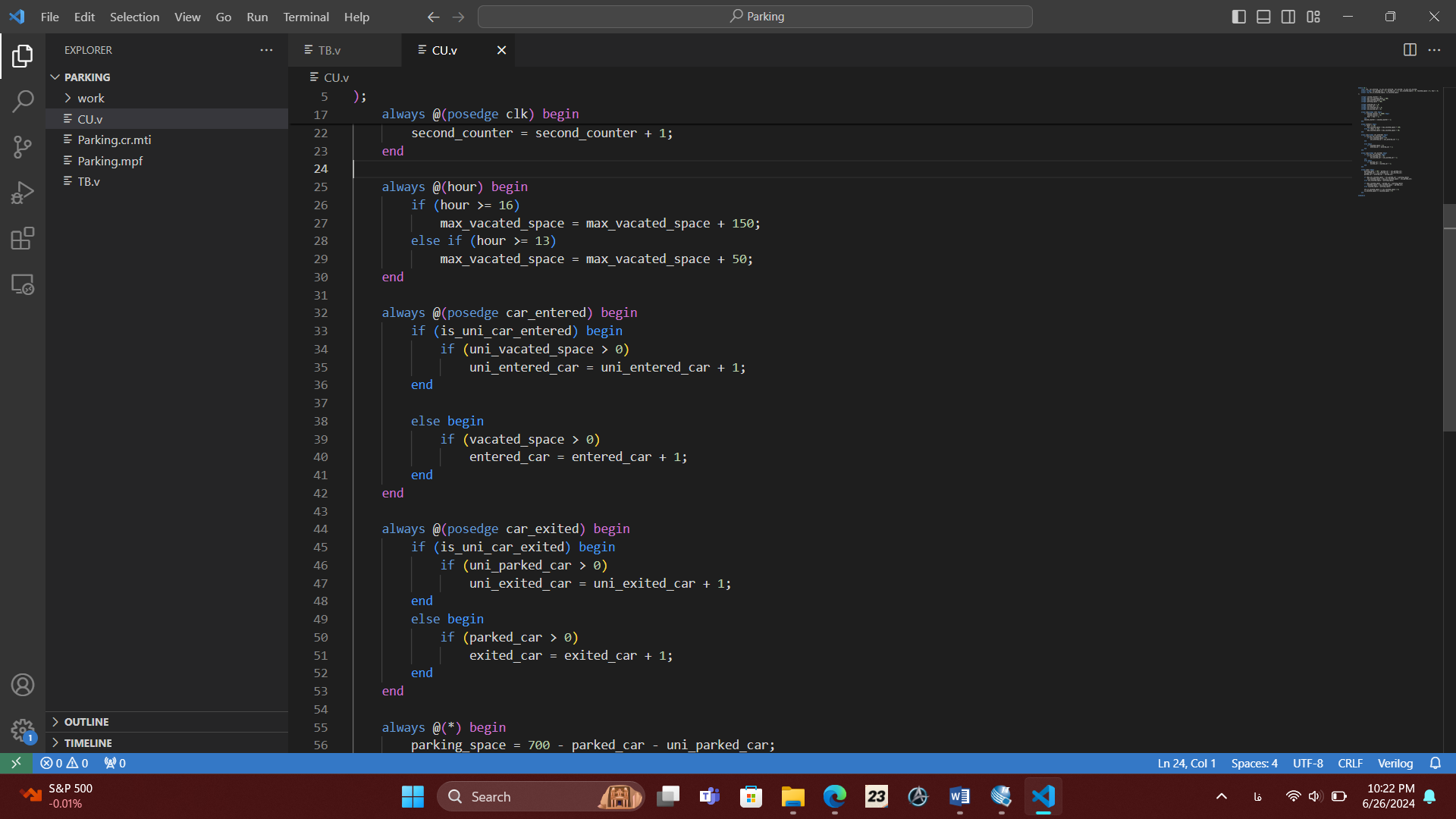
در این قسمت، ورودی و خروجی ها و دو always اول را مشاهده می‌کنید؛ همچنین تعدادی integer جدید برحسب نیاز تعریف شده اند.

Second\_counter ، شمارنده ثانیه است و در always اول زیاد می‌شود و هر کلاک را یک ثانیه درنظر می‌گیریم. درنتیجه که هرموقع به 3600 رسید، یک ساعت جلو برویم.

3 عدد بعدی، حداکثر ظرفیت ممکن برای هر بخش پارکینگ و کل پارکینگ هستند که مقداردهی اولیه (بر اساس ساعت اولیۀ 8 ) شده اند.

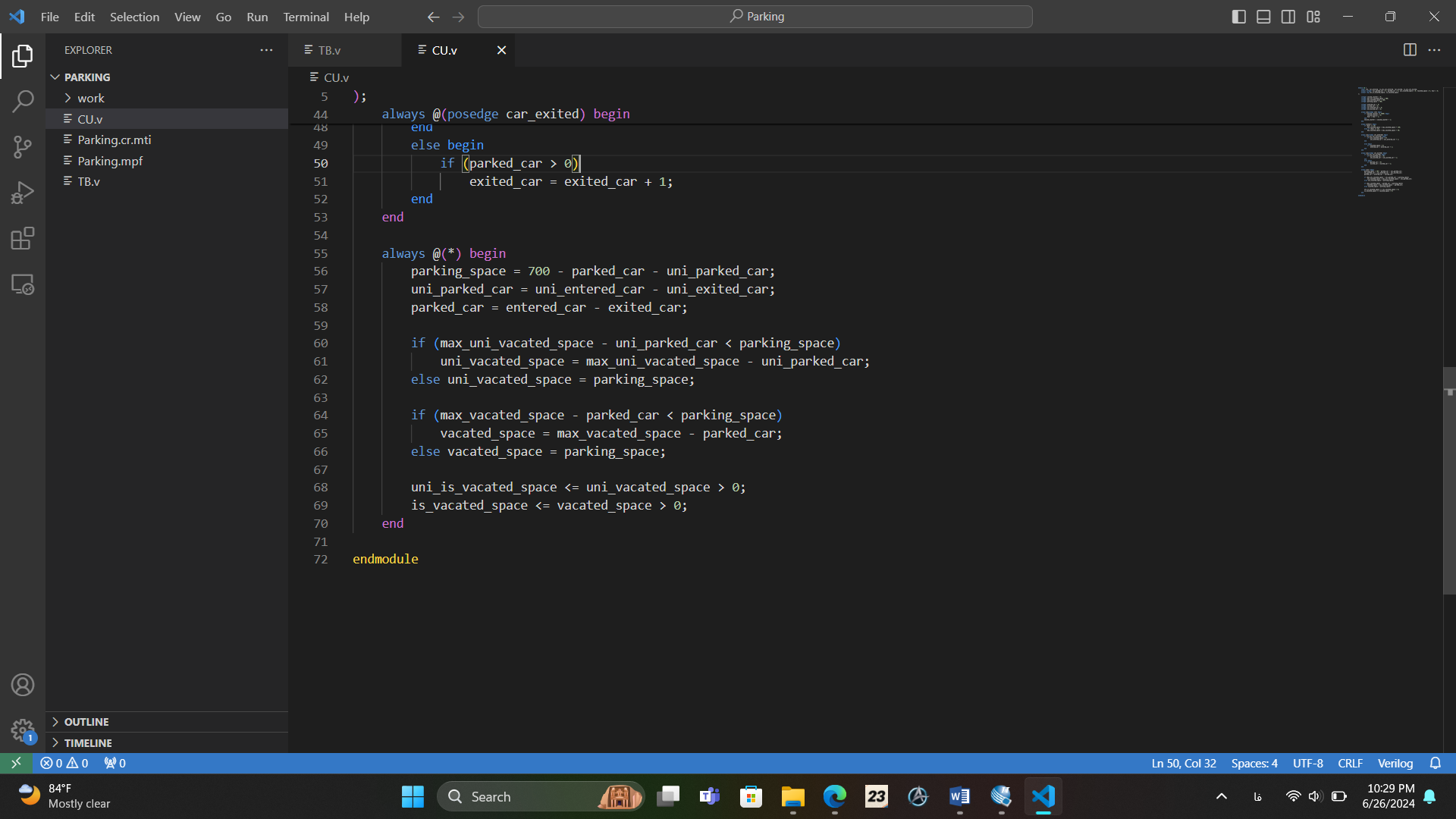
4 عدد بعدی، تعداد ماشین های وارد شده و خارج شده مربوط به هر بخش است؛ این بخش به دلیل مشکل خوردن در سنتز ایجاد شده است؛ در بخش بعدی توضیحات کامل تری در این باره خواهیم داشت.

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.



این 2 بخش مروبط به دریافت اطلاعات ورود و خروج ماشین ها است و از آن 4 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.



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.

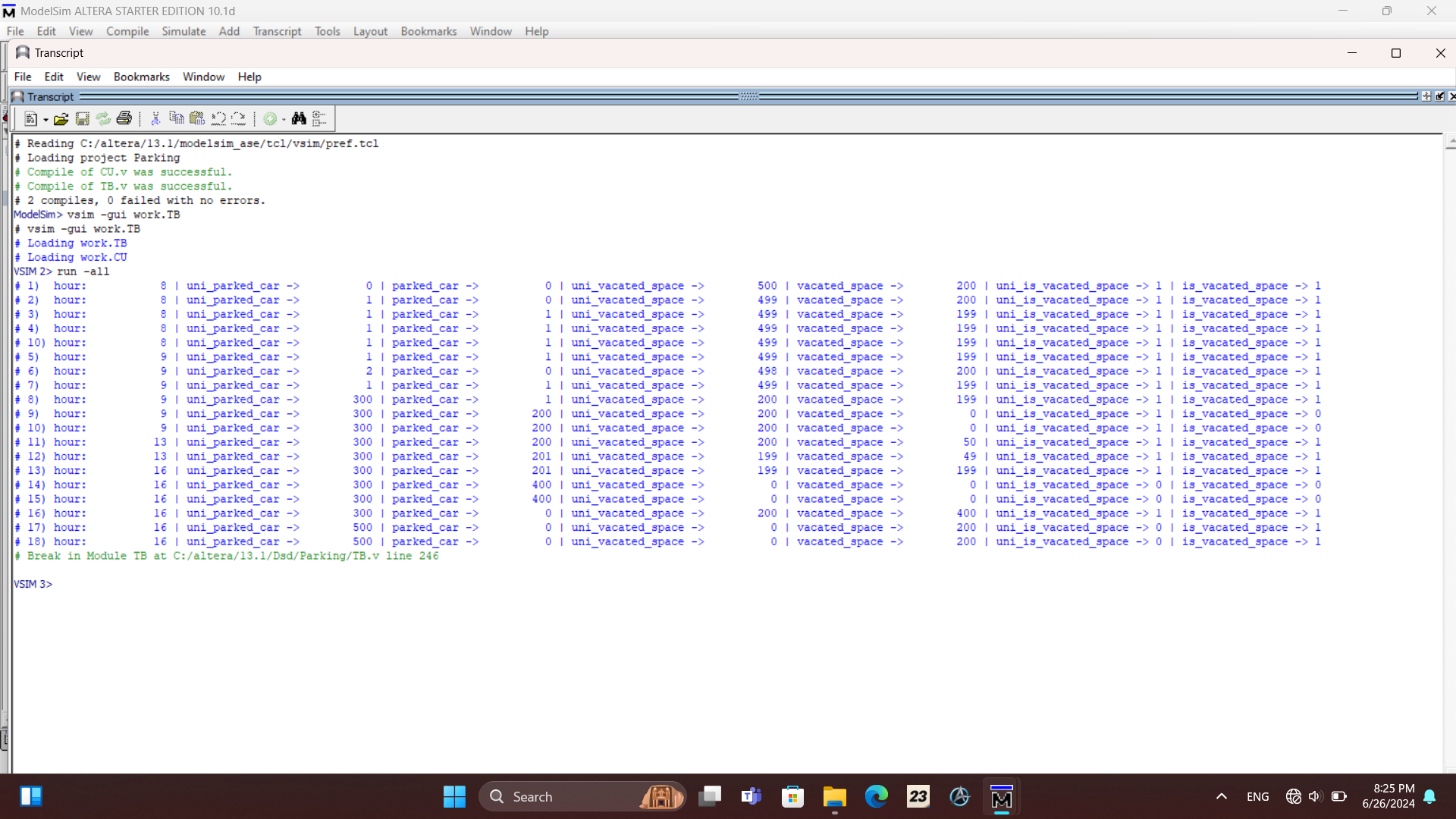
درنهایت فضای خالی مربوط به هر بخش را ( که برابر ماکسیمم ظرفیت آن بخش منهای تعداد ماشین های پارک شده آن بخش است) محاسبه می کند و اگر از کل فضای پارکینگ کمتر بود؛ به جای آن، کل فضای پارکینگ را قرار می‌دهد.

* 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 14 و 15 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 700 رسیده است.

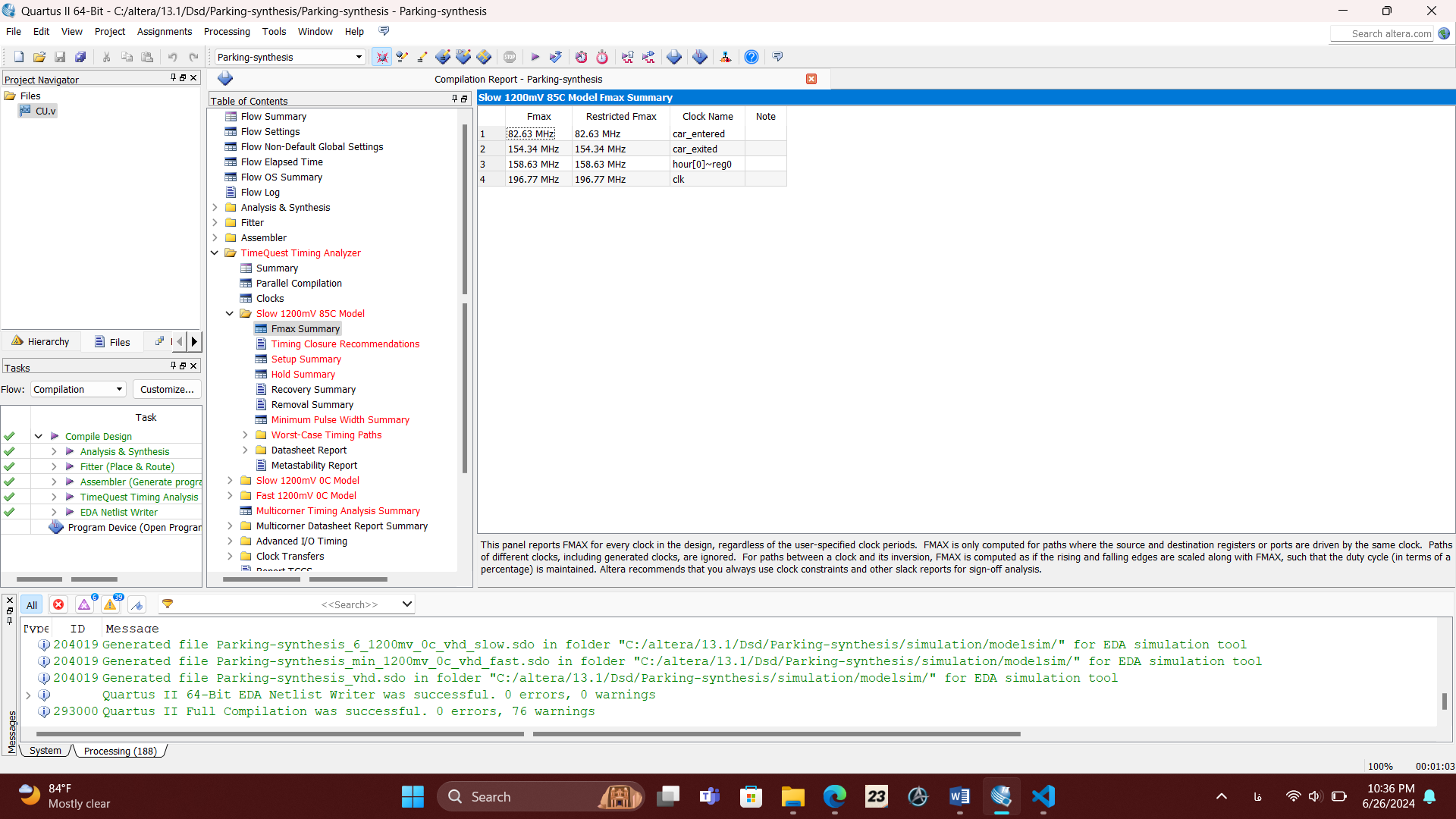
به ترتیب هر بخش را توصیف می‌کنیم می‌کنیم:

1. مقدار اولیه
2. اضافه کردن یک دانشجو
3. اضافه کردن یک انسان عادی
4. اضافه کردن و حذف کردن یک انسان عادی به صورت همزمان
5. اضافه کردن و حذف کردن یک دانشجو به صورت همزمان
6. اضافه کردن یک دانشجو و حذف یک انسان عادی
7. اضافه کردن یک انسان عادی و حذف یک دانشجو
8. اضافه کردن 299 دانشجو و رسیده تعداد دانشجو های پارک شده به 300
9. اضافه کردن 199 انسان عادی و پر شدن ظرفیت. همانطور که می‌بینیم خروجی فضای خالی عادی 0 شده است.
10. اضافه کردن یک انسان عادی دیگر. همانطور که انتظار می‌رفت، ماشینی به پارکینگ اضافه نشد!
11. منتظر ماندن تا زمانی که ساعت 13 شود و بنابراین فضای خالی عادی به 50 برسد.
12. اضافه کردن یک انسان عادی دیگر. همانطور که انتظار می‌رفت، این بار اضافه شد!
13. منتظر ماندن تا زمانی که ساعت 16 شود و بنابراین فضای خالی حداکثری عادی به 500 برسد که چون تابحال 201 نفر آمده اند باید 299 شود؛ اما چون کل طرفیت خالی برابر 199 است؛ پس باید 199 را ( که مینیمم آن هاست) ببینیم که می‌بینیم!
14. اضافه شدن 199 ماشین عادی جدید؛ بنابراین ظرفیت کل پارکینگ پر می‌شود و تعداد پارک شده های عادی 400 و دانشجویی همان 300 است.
15. اضافه کردن یک انسان عادی دیگر. همانطور که انتظار می‌رفت، ماشینی به پارکینگ اضافه نشد!
16. خارج شدن 300 ماشین عادی؛ بنابراین تعدادشان به 100 می‌رسد و حالا همه 500 ظرفیت دانشجویی قابل استفاده اند که 300 تا تابحال پرشده پس فضای خالی باید 200 باشد.
17. ورود 200 دانشجو و پر شدن ظرفیت دانشجویی.
18. 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:

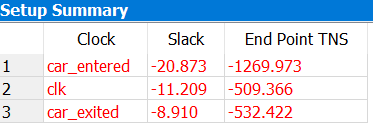
برای سنتز کد وریلاگ زده شده، از نرم‌افزار کوارتوس و Cyclone II به عنوان FPGA خود استفاده می‌کنیم.

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.



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.