به نام خداوند بخشنده‌ی مهربان

Sharif HardWar 2019

بخش Quartus - بازی Dodge

در این بخش از مسابقه می‌خواهیم با استفاده از نرم‌افزار Quartus و آنچه در استفاده از آن برای ایجاد شماتیک پیکربندی‌های سخت افزاری یاد گرفتید، بازی Dodge را پیاده سازی کنیم.

هدف این بازی کنترل یک ماشین از بین سایر ماشین‌هاست.
در ابتدای بازی، بازیکن ۳ شانس بازی (جان) دارد. با هر تصادف در مسیر یکی از جان‌های بازیکن کاسته می‌شود. همچنین با هر تصادف تمامی ماشین‌های فعلی موجود در صفحه‌ی از بین می‌روند.

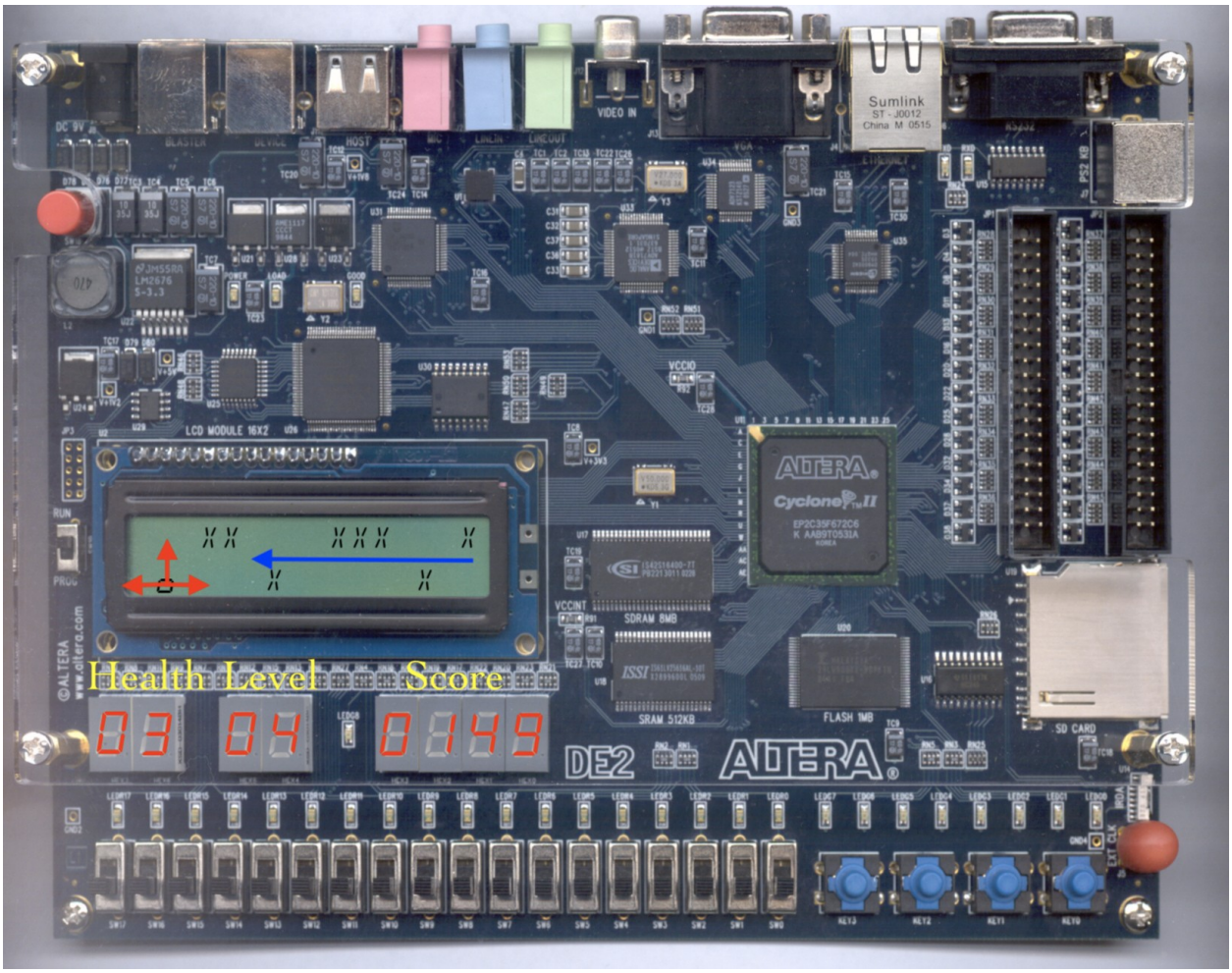
با خارج شدن هر یک از ماشین‌های عبوری مقابل (پس از رد شدن از بازیکن و رسیدن به آن طرف صفحه‌ی بازی) یک امتیاز به امتیازات بازیکن اضافه می‌شود. امتیاز بازیکن حداکثر می‌تواند ۹۹۹۹ شود.
ماشین‌های عبوری باید به نحوی تولید شوند که مسیر عبور بازیکن کاملا بسته نشود.

بازی ۴ درجه‌ی سختی دارد و سختی بازی حین اجرا قابل تنظیم است.
با تمام شدن جان‌های بازی صفحه‌ی نمایش freeze می‌شود.
همچنین می‌توان بازی را restart کرد.

بازیکن می‌تواند ماشین خود را در صفحه‌ی نمایش به جلو یا عقب حرکت دهد یا بین خطوط جابه‌جا کند.

اجرای بازی روی سخت‌افزار

برای اجرای بازی روی سخت افزار، از بورد Altera DE2 استفاده خواهیم کرد. برای نمایش صفحه‌ی بازی از نمایشگر 16x2 بورد استفاده می‌کنیم. ماشین‌های عبوری از سمت راست صفحه‌ی بازی وارد می‌شوند و در ابتدای بازی ماشین بازیکن در انتهای سمت چپ صفحه قرار دارد.



امتیاز کل روی نمایشگر BCD چهارتایی و دشواری و جان‌ها روی دو نمایشگر دوتایی دیگر نمایش داده می‌شوند.
برای restart کردن بازی از یکی از push button های برد استفاده کنید، برای انتقال عمودی از یکی از سوییچ‌ها و برای انتقال افقی از دو push button دیگر استفاده کنید (یکی برای چپ رفتن و دیگری برای راست رفتن) و در نهایت برای تغییر سطح دشواری بازی نیز از push button چهارم استفاده کنید.

پیاده سازی پیکربندی سخت‌افزار

برای پیاده سازی پیکربندی نیاز به ماژول تولید بیت تصادفی دارید. این ماژول در اختیار شما قرار خواهد گرفت. همچنین برای کار با نمایشگرهای بازی رابط‌های مورد نیاز را در اختیار خواهید داشت.

برای پیاده‌سازی پیکربندی رویکرد ماژولار را پی می‌گیریم. از این رو بخش‌های مختلف پیکربندی را در قالب ماژول‌هایی جداگانه پیاده‌سازی و تست خواهید کرد و در نهایت با استفاده از این بخش‌ها پیکربندی نهایی را تشکیل خواهید داد.

ماژول clock_generator:

در این ماژول باید با تقسیم کلاک ۲۷ مگاهرتز fpga، کلاک‌های مورد نیاز بازی را تولید کنیم.

ورودی‌ها:

level[1..0]: این ورودی سطح بازی را در حال حاضر مشخص می‌کند. سطح بازی یکی از اعداد ۰، ۱، ۲ و یا ۳ است که هرچه بیشتر باشد، بازی سخت‌تر خواهد بود.

clk: این ورودی کلاک ۲۷ مگاهرتز fpga است.

pause: هر زمان این سیگنال یک شود، باید بازی متوقف شود. در واقع هر سه خروجی این ماژول باید ثابت بمانند.

خروجی‌ها:

game_clk: این خروجی کلاک اصلی بازی را نشان می‌دهد. (مثلا سرعت ورود مانع‌های جدید را مشخص می‌کند.). از این پس به این کلاک «کلاک بازی» می‌گوییم.
دوره‌ی این کلاک به ازای سطح‌های ۰، ۱، ۲ و ۳ باید به ترتیب نزدیک به ۲,۵، ۱,۲۵، ۰,۶ و ۰,۳ ثانیه باشد.
لازم به ذکر است که سطح بازی به عنوان ورودی به این ماژول داده شده است.

clk_btn: برای این که با هر بار فشار دادن پوش‌باتن‌های fpga دقیقا یک بار عمل کنند، لازم است آن‌ها را با یک کلاک با دوره‌ی حدود ۰,۱۵ ثانیه چک کنیم. (کمی به این قضیه فکر کنید.)
این خروجی باید چنین کلاکی باشد.

clk_swtich: به دلایل مشابهی لازم است که سوییچ‌های fpga نیز با کلاکی نسبتا سریع (با فرکانسی حدود ۱۳ کیلوهرتز) چک شوند. این خروجی نیز باید چنین کلاکی باشد.

ماژول level:

این ماژول باید با ورودی گرفتن سیگنال پوش‌باتن مربوط به تغییر سطح بازی، سطح بازی را کنترل کرده و خروجی دهد.

ورودی‌ها:

clk_btn: همان خروجی clock_generator است.

level_btn: این ورودی نشان می‌دهد که آیا پوش‌باتن مربوط به تغییر سطح بازی، فشرده شده است یا نه. این ماژول باید سر هر clk_btn چک کند که اگر این ورودی صفر بود، سطح بازی را یک واحد افزایش دهد. (به صورتی که اگر در بالاترین سطح بازی باشیم، باید به پایین‌ترین سطح برویم.)

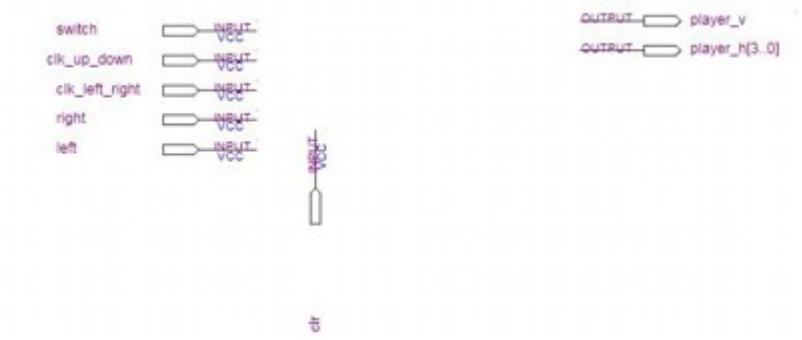
خروجی‌ها:

level[1..0]: سطح بازی را در هر لحظه خروجی می‌دهد.

هفت خروجی مربوط به سون‌سگمنت: این هفت خروجی مربوط به هفت چراغ سون‌سگمنت هستند. نیازی نیست که شما تغییری در آن ایجاد کنید.

* نکته‌ای که وجود دارد این است که سطح بازی عددی بین ۰ تا ۳ است اما عددی که باید روی سون‌سگمنت نمایش داده شود باید یکی از اعداد ۱، ۲، ۳ و ۴ باشد. بنابراین شما باید سطح بازی را یک واحد زیاد کرده و عدد سه بیتی حاصل را به ترتیب رقم کم‌ارزش به پرارزش به ورودی‌های A و B و C ی تراشه‌ی ۷۴۴۸ متصل کنید.

ماژول player:



این ماژول باید مکان فعلی بازیکن را کنترل کرده و خروجی دهد.
ورودی‌ها:

switch: این ورودی وضعیت سوییچ مربوط به مکان عمودی بازیکن را نشان می‌دهد. (در صورتی که ۰ بود بازیکن باید در سطر بالایی باشد.)

right و left: وضعیت پوش‌باتن‌های مربوط به چپ و راست رفتن بازیکن را مشخص می‌کنند. (۰ نشان می‌دهد که پوش‌باتن فشرده است.)

clk_left_right: همان clk_btn خروجی clock_generator است. پوش‌باتن‌های left و right باید با این کلاک کار کنند.

clk_up_down: همان clk_switch خروجی clock_generator است. سوییچ باید با این کلاک کار کند.

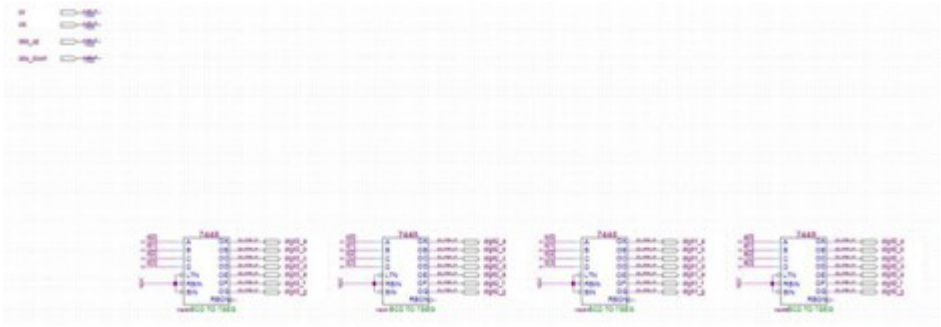
clr: هر زمان این ورودی یک شود، باید بازیکن به مکان اولیه برگردد. (ستون سمت چپ و سطری که سوییچ نشان می‌دهد.)

خروجی‌ها:

player_v: نشان می‌دهد که بازیکن در حال حاضر در کدام سطر قرار دارد. (۰ یعنی سطر بالایی و ۱ یعنی سطر پایینی)

player_h[3..0]: نشان می‌دهد که در حال حاضر بازیکن در کدام ستون قرار دارد. ستون سمت چپ شماره‌ی ۰ و ستون سمت راست شماره‌ی ۱۵ است.

ماژول point:



در این ماژول باید امتیاز بازی، که عددی ۴ رقمی در مبنای ۱۰ است، شمرده شود.
امتیازدهی به این صورت است که به ازای هر مانعی که از سمت چپ صفحه‌ی بازی خارج می‌شود، یک امتیاز دریافت می‌کنیم.

ورودی‌ها:

clk: همان خروجی game_clk از clock_generator است.

obs_up: نشان می‌دهد که در این کلاک، مانعی از سمت چپ سطر بالایی خارج شد یا نه.

obs_down: نشان می‌دهد که در این کلاک، مانعی از سمت چپ سطر پایینی خارج شد یا نه.

clr: هرگاه این ورودی یک باشد، باید امتیاز بازیکن صفر شود.

خروجی‌ها:

خروجی‌های این ماژول در واقع چهار رقم عدد امتیاز هستند که هر کدام به ۷ بیت سون‌سگمنت تبدیل شده اند. به عنوان مثال a نشان‌دهنده‌ی رقم یکان امتیاز به صورت باینری است.

ماژول collision:

وظیفه‌ی این ماژول این است که تشخیص دهد که آیا بازیکن به مانعی برخورد کرده است یا خیر.
علاوه بر این این ماژول باید تعداد جان‌های بازیکن را نیز کنترل کرده و خروجی دهد.

ورودی‌ها:

Obs[1..0][15..0]: نشان می‌دهد که در هر خانه از بازی، مانع وجود دارد یا خیر.

Player_v و player_h[3..0]: همان مکان بازیکن است که ماژول player خروجی می‌دهد.

Clk: همان کلاک بازی (game_clk) است.

Clr: باید تعداد جان‌های بازیکن را به تعداد اولیه‌ی ۳ برگرداند.

خروجی‌ها:

Collision: نشان می‌دهد که آیا بازیکن در این لحظه با مانعی برخورد دارد یا خیر. لازم به ذکر است که اگر برخوردی رخ داد باید یکی از جان‌های بازیکن کم شود.

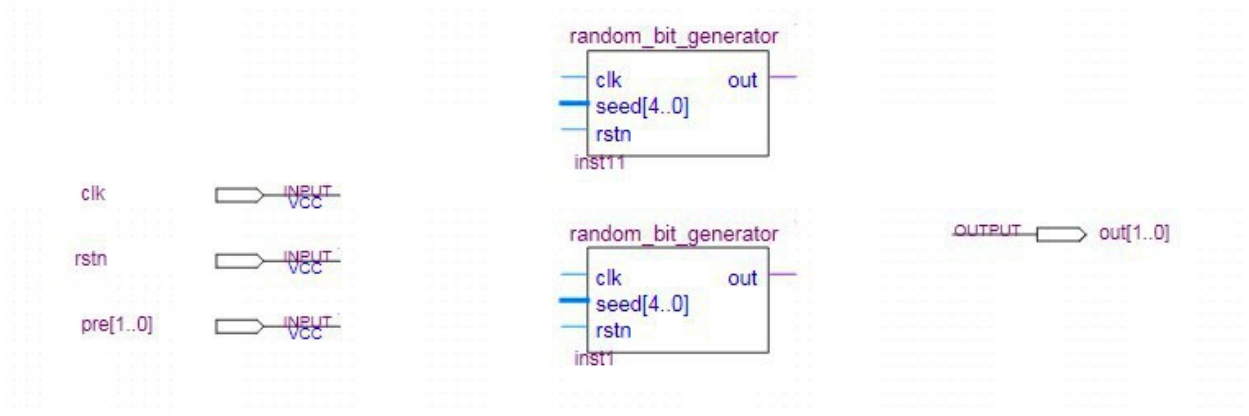
Lose: هر گاه تعداد جان‌های بازیکن به صفر رسید، این خروجی باید یک شود.

خروجی‌های مربوط به سون‌سگمنت تعداد جان: برای این قسمت کافی است تعداد جان‌ها را در lives[1..0] بریزید تا در سون‌سگمنت نمایش داده شود.

* دقت کنید که اگر حتی collision در زمانی غیر از لبه‌ی کلاک رخ داد نیز باید تشخیص داده شود.

ماژول obstacle_generator:

این ماژول قرار است در هر کلاک از سمت راست صفحه‌ی بازی دو خانه‌ی جدید وارد کند. (یکی بالا و یکی پایین) به این صورت که هر کدام از این دو خانه می‌توانند مانع باشند یا نباشند. همچنین باید توجه کنیم که مسیر بازیکن نباید به طور کامل بسته شود. بنابراین، اولا نباید هر دو خانه‌ی جدید مانع باشند، ثانیا خانه‌های جدید به دو خانه‌ی قبلی نیز بستگی دارند.



ورودی‌ها:

Clk: همان کلاک بازی است.

Pre[1..0]: دو خانه‌ی قبلی هستند. Pre[0] خانه‌ی سطر بالایی و pre[1] خانه‌ی سطر پایینی.

Rstn: ماژول‌های random_bit_generator یک ورودی seed[4..0] ورودی می‌گیرند و به ازای هر seed یک دنباله را خروجی می‌دهند. شما باید هر بار که rstn صفر شد، seed هر دو random_bit_generator را تغییر دهید. علاوه بر این دقت کنید که seed نمی‌تواند به طور کامل صفر باشد.
همچنین این ورودی را مستقیما به rstnهای ورودی random_bit_generator وصل کنید.

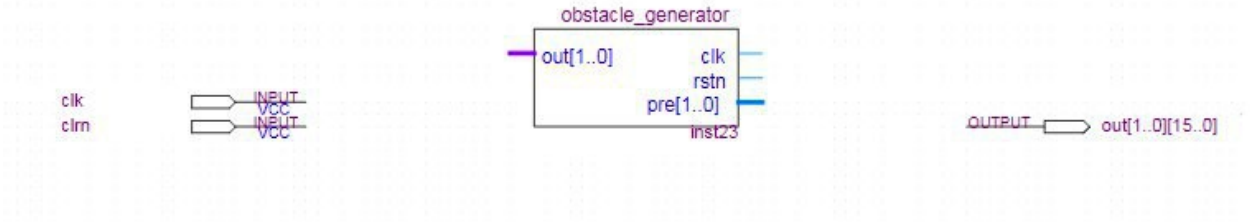
خروجی‌ها:

Out[1..0]: دو خانه‌ی جدید را خروجی می‌دهد. Out[0] و out[1] نشان می‌دهند که به ترتیب آیا در خانه‌ی بالایی و پایینی مانع هست یا نه.

* برای تولید out[1..0] از دو بیت تولید شده توسط random_bit_generatorها استفاده کنید اما در عین حال توجه کنید که نباید مسیر بازیکن به طور کامل بسته شود.

ماژول obstacles:

وظیفه‌ی این ماژول این است که وضعیت کنونی صفحه‌ی بازی (وجود یا عدم وجود مانع در هر خانه) را ذخیره کرده و خروجی دهد.



ورودی‌ها:

Clk: همان کلاک بازی است.

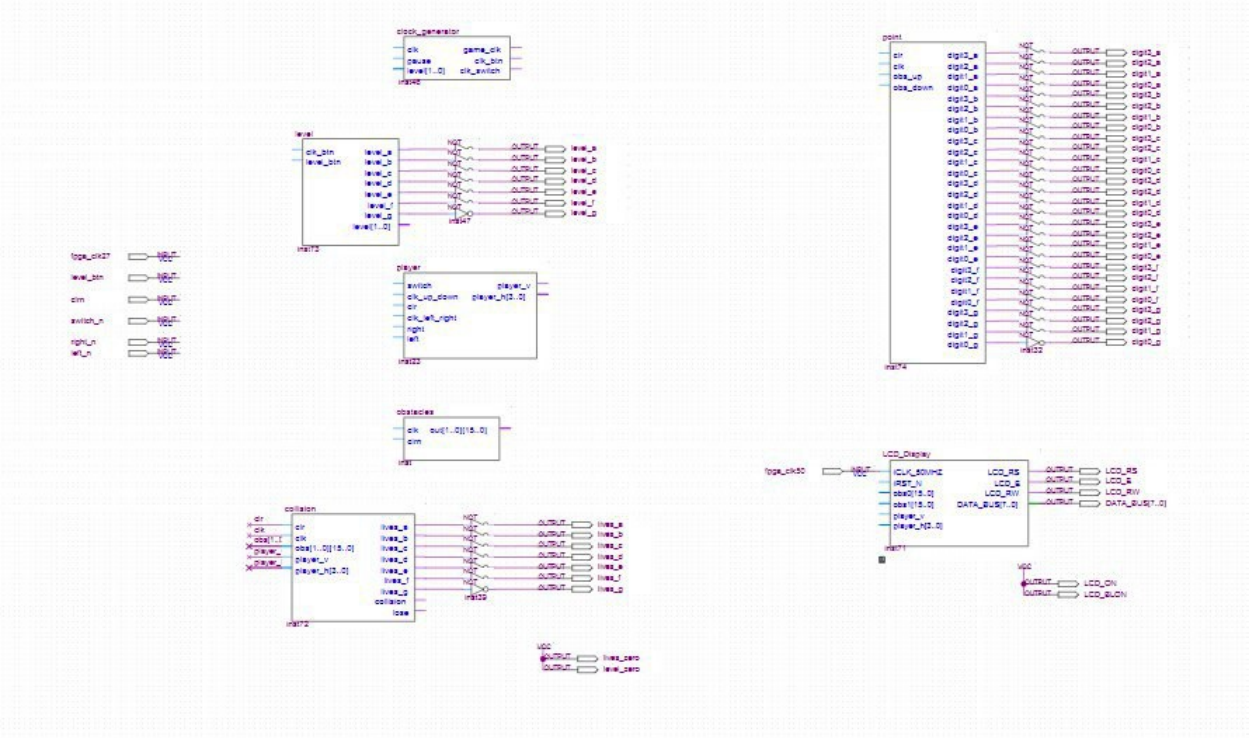
Clrn: هر گاه این ورودی صفر شد، باید صفحه‌ی بازی کاملا خالی شده و seed رندوم‌های obstacle_generator نیز تغییر کنند.

خروجی‌ها:

Out[1..0][15..0]: وضعیت کنونی خانه‌های صفحه‌ی بازی را نشان می‌دهد.

ماژول map:

Map ماژول اصلی بازی است که تمام ماژول‌های قبلی را کنار هم می‌گذارد و به هم ارتباط می‌دهد.



ورودی‌ها:

Fpga_clk27: کلاک ۲۷مگاهرتز fpga است.

Fpga_clk50: کلاک ۵۰مگاهرتز fpga است.

Level_btn: در صورتی که صفر باشد یعنی پوش‌باتن مربوط به سطح بازی فشرده شده است.

Clrn: در صورتی که صفر باشد یعنی پوش‌باتن مربوط به ریست کردن بازی فشرده شده است.

Switch_n: در صورتی که صفر باشد یعنی سوییچ در حالت پایین قرار دارد و برعکس.

right_n: در صورتی که صفر باشد یعنی پوش‌باتن مربوط به حرکت بازیکن به سمت راست فشرده شده است.

left_n: در صورتی که صفر باشد یعنی پوش‌باتن مربوط به حرکت بازیکن به سمت چپ فشرده شده است.

خروجی‌ها:

خروجی‌های این ماژول به پین‌های fpga متصل می‌شوند و نیازی به تغییرشان نیست.

*** توضیحاتی راجع به ماژول LCD_Display:**

از این ماژول استفاده کنید تا المان‌های بازی (مثل بازیکن و موانع) را روی LCD نمایش دهید:

ورودی‌ها:

کلاک ۵۰مگاهرتز fpga که متصل شده است.

iRST_N: السی‌دی را خالی می‌کند. ورودی clrn را مستقیما به آن وصل کنید.

Obs0[15..0]: موانع سطر بالایی را به آن وصل کنید.

Obs1[15..0]: موانع سطر پایینی را به آن وصل کنید.

Player_v: شماره‌ی سطر بازیکن را به آن وصل کنید.

Player_h[3..0]: شماره‌ی ستون بازیکن را به آن وصل کنید.