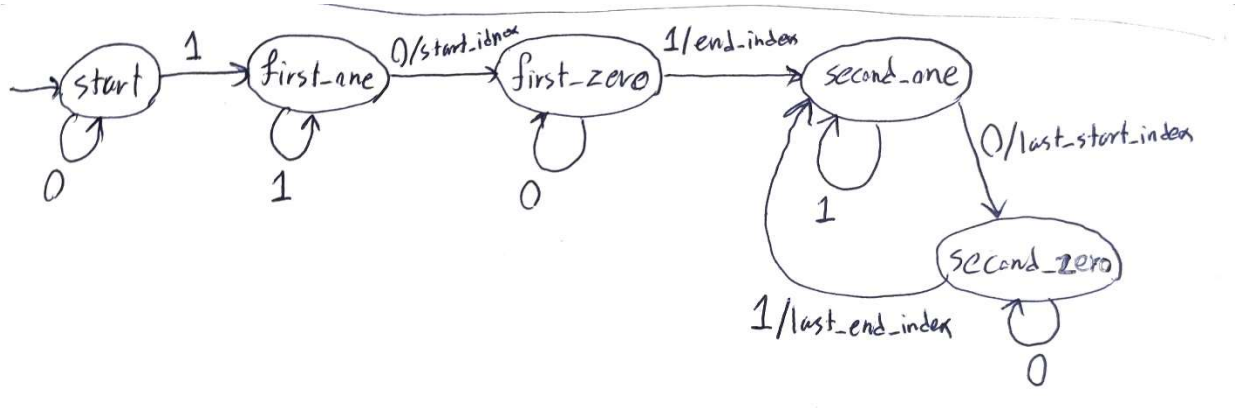
به نام خدا

فرزان رحمانی ۹۹۵۲۱۲۷۱

گزارش تمرین سری سوم طراحی کامپیوتری سیستم های دیجیتال

سوال اول



توضیح کد سوال:
کد سوال یک شبیه‌سازی یک ماشین حالت (Finite State Machine) است که با استفاده از زبان VHDL پیاده‌سازی شده است. این ماشین حالت ورودی‌هایی را دریافت می‌کند و بر اساس آنها به یک توالی از حالت‌ها پیروی می‌کند تا خروجی‌های مورد نظر را تولید کند.

ماشین حالت از ورودی‌های زیر استفاده می‌کند:

- `clk`: سیگنال ساعت برای مدیریت روند زمانی.
- `reset`: سیگنال بازنشانی برای ریست کردن ماشین حالت.
- `input`: ورودی که یک بردار از بیت‌ها است با طول ۲۱ بیت(می توانیم هر عدد دیگری نیز بذاریم).

و خروجی‌های زیر را تولید می‌کند:

- `start_index`: برداری ۵ بیتی که مشخص می‌کند شروع محدوده‌ی یک.
- `end_index`: برداری ۵ بیتی که مشخص می‌کند پایان محدوده‌ی یک.
- `last_start_index`: برداری ۵ بیتی که مشخص می‌کند شروع محدوده‌ی دو.
- `last_end_index`: برداری ۵ بیتی که مشخص می‌کند پایان محدوده‌ی دو.

کد اصلی، شامل ماشین حالت و منطق مربوط به تغییر حالت‌ها و محاسبه خروجی‌ها است. در هر حالت، ورودی بررسی می‌شود و با توجه به آن و وضعیت فعلی، حالت بعدی مشخص می‌شود. در نهایت، خروجی‌های محاسبه شده در حالت‌های مورد نظر قرار می‌گیرند.

توضیح کد تست:

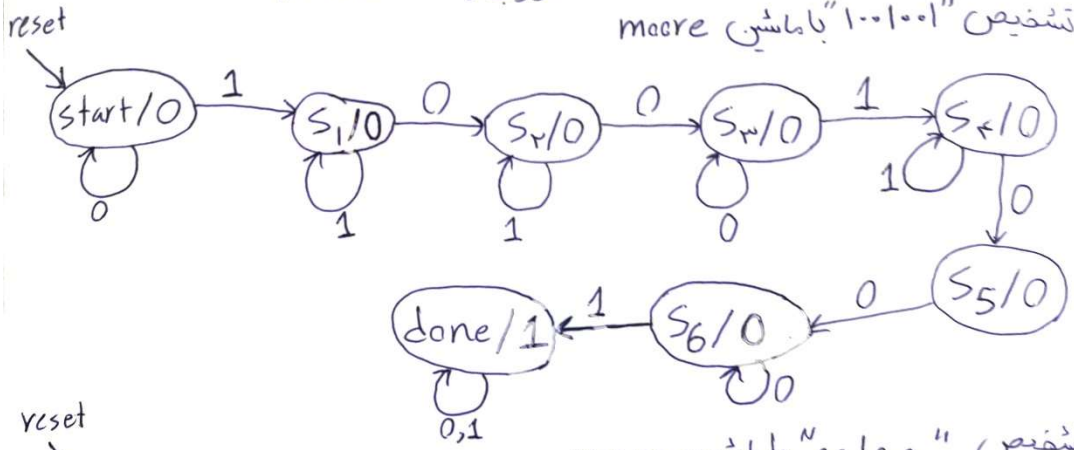
کد تست نیز برای تست و اعتبارسنجی ماشین حالت استفاده می‌شود. در این کد، سیگنال‌های ورودی و خروجی تعریف شده و ماشین حالت در قالب یک کامپوننت مورد استفاده قرار می‌گیرد. سپس با تغییر سیگنال‌های ورودی، عملکرد ماشین حالت و خروجی‌های آن بررسی می‌شود.

پس از اجرای کد تست، سیگنال‌های ورودی و خروجی طی زمان مشخص شده تغییر می‌کنند. این تغییرات موجب تغییر حالت‌ها و محاسبه خروجی‌های متناظر با آنها می‌شود. در نهایت، از محیط شبیه‌سازی برای بررسی صحت عملکرد ماشین حالت استفاده می‌شود.

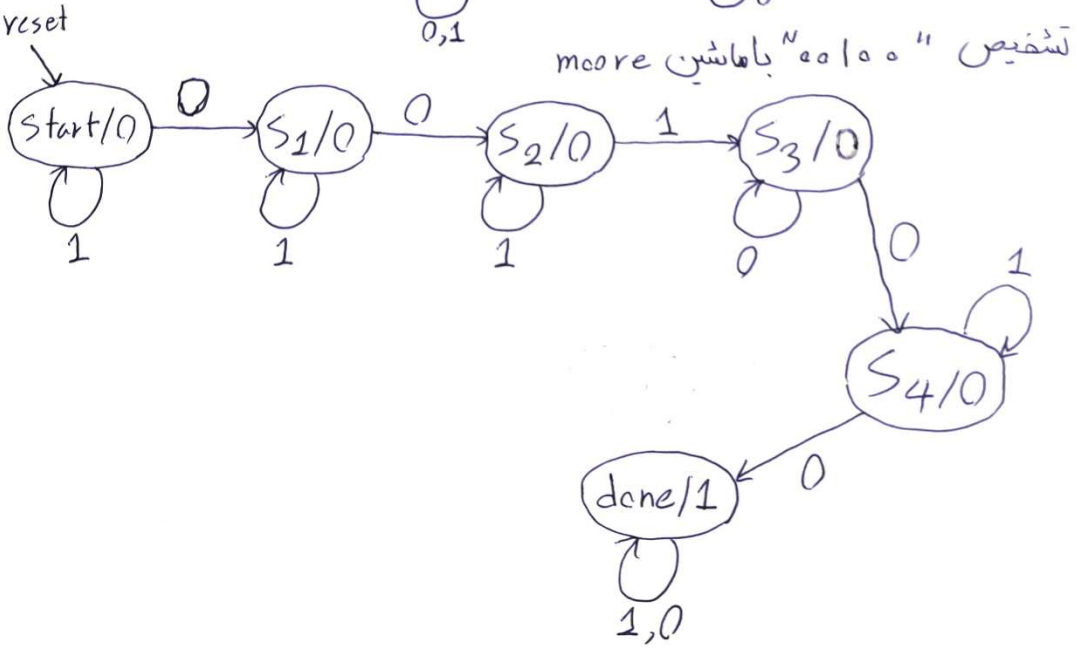
سوال دوم

سوال دوم: از دو ماشین برای تشخیص دو زیررشته "۰۰۱۰۰" و "۱۰۰۱۰۰۱" استفاده می‌کنیم.
سپس با توجه به اینکه کدام یک یافت شده است خروجی را مشخص می‌کنیم.

تشخیص "۱۰۰۱۰۰۱" با ماشین moore



تشخیص "۰۰۱۰۰" با ماشین moore



آر ماشین اول در حالت done بود خروجی را معکوس می‌کنیم. در غیر این صورت آر ماشین دوم در حالت done بود خروجی را با یک جمع می‌کنیم. در غیر این صورت خروجی را تغییر نمی‌دهیم.

توضیح کد سوال:

کد بال یک ماشین حالت (Finite State Machine) با نام `q2_fsm` را در زبان VHDL پیاده‌سازی کردم. این ماشین حالت ورودی‌هایی را دریافت می‌کند و بر اساس آنها به یک توالی از حالت‌ها پیروی می‌کند تا خروجی‌های مورد نظر را تولید کند.

در موارد زیر، کدهای مهم و توضیحات مربوط به آنها آمده است:

۱. `generic (n: positive := 17)`: یک پارامتر ژنریک به نام `n` با مقدار پیش‌فرض ۱۷ تعریف شده است.

۲. `port`: ورودی‌ها و خروجی‌های ماشین حالت در این قسمت تعریف شده است. این ورودی‌ها و خروجی‌ها عبارتند از:
- `clk`: سیگنال ساعت برای مدیریت روند زمانی.
- `reset`: سیگنال بازنشانی برای ریست کردن ماشین حالت.
- `input`: ورودی که یک بردار از بیت‌ها است با طول `n-1`.
- `output`: خروجی که یک بردار از بیت‌ها است با طول `n-1`.

۳. `type state_type is (start, s1, s2, s3, s4, s5, s6, done)`: یک نوع داده به نام `state_type` تعریف شده است که حالت‌های ماشین حالت را نشان می‌دهد.

۴. `type state_type_2 is (start_2, s1_2, s2_2, s3_2, s4_2, done_2)`: یک نوع داده به نام `state_type_2` تعریف شده است که حالت‌های دوماشین حالت را نشان می‌دهد.

۵. `signal state: state_type;` و `signal state_2: state_type_2;`: سیگنال‌های `state` و `state_2` برای نگهداری حالت‌های فعلی ماشین حالت و دوماشین حالت تعریف شده‌اند.

۶. `signal index: unsigned(4 downto 0);` و `signal index_2: unsigned(4 downto 0);`: سیگنال‌های `index` و `index_2` برای نگهداری مق

ادیر فهرست‌ها و دومقدارهای فعلی تعریف شده‌اند.

۷. `signal temp_output: std_logic_vector(n-1 downto 0);` و `signal temp_output_2: std_logic_vector(n-1 downto 0);`: سیگنال‌های `temp_output` و `temp_output_2` برای نگهداری خروجی‌های موقتی ماشین حالت و دوماشین حالت تعریف شده‌اند.

۸. در بلوک `process (clk, reset)`، فعالیت‌های ماشین حالت و دوماشین حالت در هر لبه صعودی ساعت و در صورتی که سیگنال بازنشانی روشن باشد انجام می‌شود.

۹. در بلوک `if reset = '1' then`، وضعیت ماشین حالت و دوماشین حالت در حالت بازنشانی قرار می‌گیرند.

۱۰. در بلوک `elsif rising_edge(clk) then`، وضعیت ماشین حالت و دوماشین حالت در صورتی که ماشین حالت در حالت کاری باشد به‌روزرسانی می‌شود.

۱۱. در بلوک ` output <= (not input) when state = done else std_logic_vector(unsigned(input) + 1) when state_2 = done_2 else input;`، خروجی ماشین حالت و دوماشین حالت تعیین می‌شود. اگر ماشین حالت در حالت `done` باشد، خروجی آن برابر با `not input` است. در غیر این صورت، اگر دوماشین حالت در حالت `done_2` باشد، خروجی آن برابر با `std_logic_vector(unsigned(input) + 1)` است. در غیر این صورت، خروجی برابر با ورودی است.

این کد به طور خاص برای دو ماشین حالت مجزا طراحی شده است که هر کدام به صورت جداگانه ورودی‌ها را بررسی کرده و خروجی‌های مربوطه را تولید می‌کنند.

توضیح کد تست:

کد تست **q2_fsm_tb** برای تست و اعتبارسنجی واحد طراحی شده با نام **q2_fsm** است. در این کد، واحد طراحی **q2_fsm** به عنوان یک کامپوننت تعریف شده است و از آن استفاده می‌شود.

در بخشهای زیر توضیحات مربوط به کد تست آمده است:

۱. **component q2_fsm**: تعریف کامپوننت **q2_fsm** که بر اساس آن واحد طراحی **q2_fsm** تست می‌شود. این کامپوننت دارای ورودی‌ها و خروجی‌های مشخصی است که باید با ورودی‌ها و خروجی‌های کامپوننت مورد تست مطابقت داشته باشد.

۲. **signal clk, reset, input, output**: تعریف سیگنال‌های مورد استفاده در کد تست، که به ورودی‌ها و خروجی‌های کامپوننت متصل می‌شوند.

۳. **UUT: q2_fsm**: نمونه‌ای از کامپوننت **q2_fsm** با نام **UUT** تعریف شده است. این نمونه برای تست واحد طراحی استفاده می‌شود.

۴. **UUT: q2_fsm generic map (n => n)**: تعیین مقادیر ژنریک برای نمونه کامپوننت **UUT**، به طوری که مقدار ژنریک **n** با مقدار **n** تعریف شده در کد تست برابر باشد.

۵. **UUT: q2_fsm port map (clk => clk, reset => reset, input => input, output => output)**: اتصال سیگنال‌های تست به پورت‌های کامپوننت **UUT**.

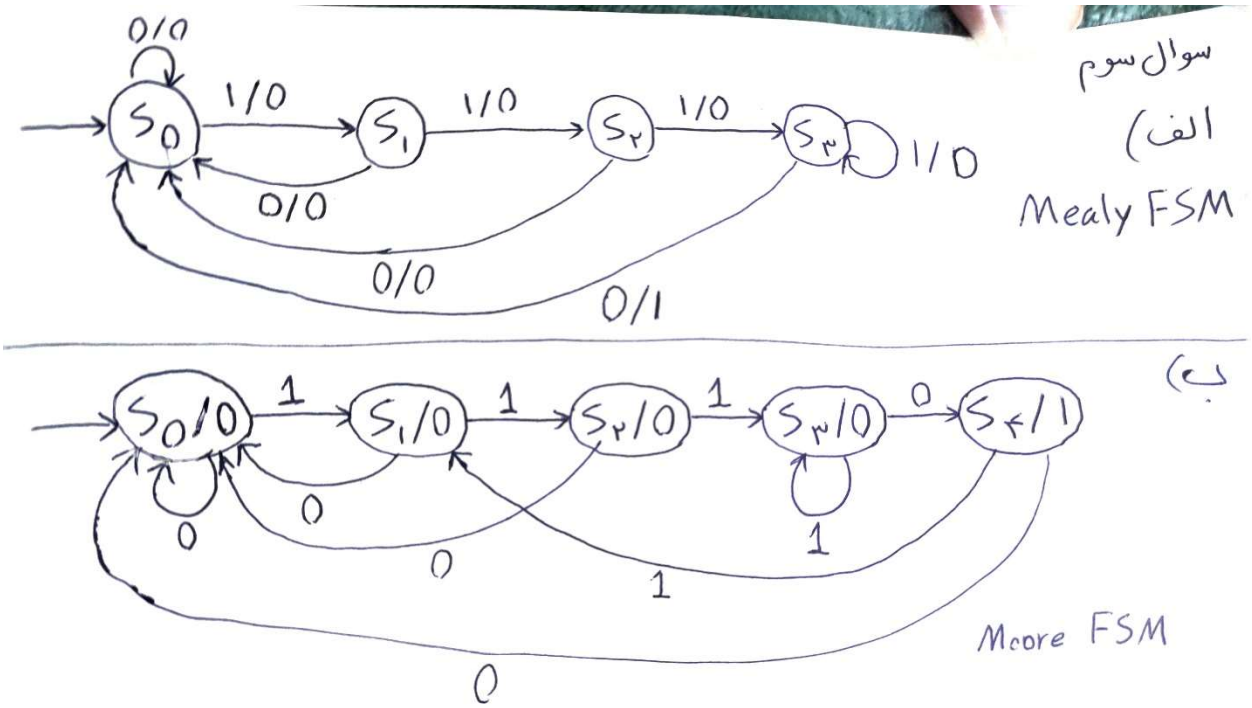
۶. **process**: یک فرایند برای تولید سیگنال ساعت **clk** با فرکانس مشخص شده است. سیگنال ساعت ابتدا به ۰ تنظیم می‌شود و سپس بعد از ۵ نانوثانیه به ۱ تغییر می‌کند. سپس بعد از ۵ نانوثانیه دیگر، دوباره به ۰ تغییر می‌کند. این فرایند برای ایجاد ساعت مناسب برای تست استفاده می‌شود.

۷. **process**: یک فرایند تست که ورودی‌ها و ریست را تنظیم می‌کند و سپس صبر می‌کند تا خروجی‌ها تولید شوند. در این فرایند، تست‌های مختلف با استفاده از تنظیمات متفاوت برای ورودی‌ها انجام می‌شود و نتایج تولید شده بررسی می‌شود. این فرایند شامل چند بلاک **wait** برای ایجاد دوره‌های ساعت و تنظیم ورودی‌ها و ریست است.

۸. **configuration**: تنظیمات کد تست برای استفاده از واحد طراحی **q2_fsm** تعریف شده است.

کد تست **q2_fsm_tb** برای اجرا و اعتبارسنجی واحد طراحی **q2_fsm** استفاده می‌شود. در این کد، سیگنال‌های ورودی مورد نیاز برای تست تنظیم شده و نتایج تولید شده بررسی می‌شوند. این کد می‌تواند به عنوان یک مثال از نحوه استفاده از واحد طراحی **q2_fsm** و تست عملکرد آن استفاده شود.

سوال سوم



الف)

توضیح کد سوال:

این کد یک ماشین حالت Mealy با عملکردی مشخص بر اساس ورودی‌ها و ساعت را نشان می‌دهد. در این ماشین حالت، وضعیت در حالت‌های s0 ، s1، s2 و s3 تغییر می‌کند و خروجی متناسب با هر وضعیت و ورودی مشخص می‌شود.

نکته‌های مهم در کد عبارتند از:

۱. **type state_type is (s0, s1, s2, s3);**: تعریف یک نوع با نام **state_type** که شامل حالت‌های s0 ، s1، s2 و s3 است. این نوع برای نگهداری وضعیت ماشین حالت استفاده می‌شود.

۲. :**signal state : state_type;**تعریف یک سیگنال به نام **state**از نوع **state_type**برای نگهداری وضعیت فعلی ماشین حالت.

۳. :**process (clk, reset)**یک فرایند که به تغییرات ساعت و ورودی را واکنش نشان می‌دهد.

۴. :**if reset = '1' then**بررسی وضعیت ریست. اگر ریست برابر با ۱ باشد، ماشین حالت به وضعیت اولیه s0 برمی‌گردد.

۵. :**elsif rising_edge(clk) then**بررسی لبه صعودی ساعت. اگر لبه صعودی ساعت رخ دهد، فرایند حالت جاری را بررسی می‌کند و با توجه به وضعیت و ورودی‌های مشخص، وضعیت بعدی را تعیین می‌کند.

۶. :**case state is**یک ساختار switch-case برای بررسی حالت فعلی ماشین.

۷. هر حالت در بلاک case مشخص شده است و با توجه به وضعیت فعلی و ورودی، وضعیت بعدی تعیین می‌شود. برای مثال، اگر حالت s0 باشد و ورودی ۱ باشد، وضعیت به s1 تغییر می‌کند.

۸. :**output <= '1' when (state = s3 AND input = '0') ELSE '0';**تعیین خروجی ماشین حالت. اگر حالت فعلی s3 باشد و ورودی ۰ باشد، خروجی ۱ است، در غیر این صورت خروجی ۰ است.

این کد یک ماشین حالت Mealy است که ورودی را در نظر می‌گیرد و وضعیت و خروجی را بر اساس ورودی و تغییرات ساعت تعیین می‌کند.

توضیح کد تست:

کد تست ماشین حالت Mealy را توضیح می‌دهد. در این تست بنچ، ماشین حالت Mealy که قبلاً تعریف شده است، تست می‌شود. ورودی‌ها و ساعت با استفاده از سیگنال‌های **clk**، **reset**و **input**تعیین می‌شوند و خروجی ماشین نیز با سیگنال **output**مشاهده می‌شود.

نکته‌های مهم در کد عبارتند از:

۱. تعریف سیگنال‌های **clk**، **reset**، و **input**برای تست ماشین حالت.Mealy

۲. :**UUT : mealy_fsm port map**نقشه‌برداری پورت‌های ماشین حالت Mealy به سیگنال‌های تعریف شده در تست بنچ.

۳. :**process**یک فرایند برای تنظیم ساعت.

۴. تنظیم سیگنال **clk**به '۰' و '۱' با فاصله زمانی ۵ nsبه منظور ایجاد ساعت.

۵. دومین فرایند برای تنظیم سیگنال‌های **reset**و **input**به منظور ارسال ورودی‌های تستی به ماشین حالت.

۶. تنظیم سیگنال **reset**به '۱' برای ریست ماشین، سپس تنظیم ورودی‌های تستی با تاخیرهای زمانی ۱۰ns.

۷. پس از تنظیم ورودی‌های تستی، فرایند منتظر می‌ماند و تست بنچ پایان می‌یابد.

در این کد تست، ورودی‌های **input**به صورت توالی '۰'، '۱'، '۱'، '۱'، '۰'، '۱'، '۰'، '۱'، '۱'، '۰' تنظیم می‌شوند و خروجی ماشین را مشاهده می‌کنیم.

توجه داشته باشید که در کد تست، سیگنال **output**برای مشاهده خروجی ماشین تنظیم نشده است. بنابراین، خروجی ماشین در این کد تست قابل مشاهده نیست و تنها ورودی‌ها ارسال می‌شوند.

ب)

توضیح کد سوال:
کد ماشین حالت Moore را توضیح می‌دهد. در این کد، ماشین حالت Moore با پورت‌های **clk**، **reset**، **input**، و **output**تعریف شده است. ماشین حالت به وسیله سیگنال **state**مدل‌سازی می‌شود.

نکته‌های مهم در این کد عبارتند از:

۱. تعریف متغیر **state**با نوع **state_type**که شامل وضعیت‌های **s0**، **s1**، **s2**، **s3**و **s4**است.

۲. فرایندی که با تغییرات سیگنال‌های **clk**و **reset**فعال می‌شود. در صورتی که **reset**برابر با '۱' باشد، ماشین به وضعیت اولیه **s0**بازمی‌گردد.

۳. با فعال شدن لبه صعودی سیگنال **clk**، یک **case**برای وضعیت‌های ماشین تعریف شده است. در هر وضعیت، براساس ورودی **input**، ماشین به وضعیت بعدی منتقل می‌شود.

۴. در وضعیت **s3**، اگر ورودی **input**برابر با '۰' باشد، ماشین به وضعیت نهایی **s4**منتقل می‌شود که در آن خروجی **output**برابر با '۱' قرار دارد. در غیر این صورت، ماشین در وضعیت **s3**باقی می‌ماند تا تاخیری با ورودی '۰' رخ دهد.

۵. در وضعیت **s4**، ماشین براساس ورودی **input**به وضعیت جدید منتقل می‌شود. اگر **input**برابر با '۰' باشد، ماشین به وضعیت اولیه **s0**منتقل می‌شود. در غیر این صورت، ماشین در وضعیت **s1**باقی می‌ماند.

۶. در نهایت، خروجی **output**بر اساس وضعیت کنونی ماشین تعیین می‌شود. در وضعیت **s4**، خروجی **output**برابر با '۱' است و در سایر وضعیت‌ها خروجی **output**برابر با '۰' است.

این کد ماشین حالت Moore را توصیف می‌کند که ورودی‌ها را بررسی کرده و خروجی را بر اساس وضعیت کنونی ماشین تعیین می‌کند.

توضیح کد تست:

کد تست برای ماشین حالت Moore را توضیح می‌دهد. در این کد، ماشین حالت Moore با پورت‌های **clk**، **reset**، **input**، و **output**به عنوان ورودی‌ها و خروجی‌ها تعریف شده است.

نکته‌های مهم در این کد عبارتند از:

۱. تعریف کامپوننت **moore_fsm**که پورت‌های ماشین حالت را شبیه‌سازی می‌کند.

۲. تعریف سیگنال‌های ورودی و خروجی برای ماشین حالت، که به پورت‌های ماشین حالت متصل می‌شوند.

۳. فرایند اصلی کد تست که دو فرآیند داخلی دارد. اولین فرآیند با تغییر سیگنال **clk**فعال می‌شود و مقادیر '۰' و '۱' را به تناوب سیگنال **clk**اعمال می‌کند.

۴. دومین فرآیند مربوط به تست ورودی‌ها و خروجی‌های ماشین حالت است. در این فرآیند، ابتدا پس از انقضای زمان ۱۰۰ns، سیگنال **reset**را به '۱' تنظیم می‌کند تا ماشین حالت ریست شود. سپس ورودی **input**را تنظیم می‌کند و به ازای هر مقدار ورودی، زمان‌های مشخصی را منتظر می‌ماند. این فرآیند شامل چندین تغییر ورودی **input**است تا تست دنباله ورودی صورت بگیرد.

۵. در نهایت، فرآیند تمام شده را با استفاده از **wait**بی‌نهایت متوقف می‌کند.

۶. تنظیمات **configuration**نیز انجام می‌شود تا کامپوننت **moore_fsm**به عنوان واحد تست استفاده شود و تغییرات مورد نیاز را اعمال کند.

کد تست ابتدا سیگنال **clk**را تنظیم کرده و سپس پس از گذشت زمان‌های مشخص، ورودی‌ها را تغییر می‌دهد و نتایج را بررسی می‌کند. با توجه به دنباله ورودی‌های تست شده، خروجی ماشین حالت را بررسی کرده و نتایج را ثبت می‌کند.

سوال چهارم

توضیح کد سوال:

این کد، طراحی یک مبدل binary به unary را نشان می‌دهد. در این کد، ما یک واحد **BINARY_TO_UNARY**با دو پورت ورودی و خروجی تعریف کرده‌ایم. ورودی **binary**یک بردار از سه بیت است و خروجی **unary**یک بردار از هشت بیت است.

در معماری **transform**، یک فرآیند با ورودی **binary**تعریف شده است. در این فرآیند، با استفاده از ساختار **case**، مقادیر ممکن برای **binary**بررسی می‌شوند و خروجی **unary**متناسب با مقدار **binary**تنظیم می‌شود.

مقادیر **binary**از "۰۰۰" تا "۱۱۱" بررسی می‌شوند و مقادیر متناظر آن‌ها در **unary**تنظیم می‌شوند. به عنوان مثال، وقتی **binary**برابر با "۰۰۰" است، خروجی **unary**باید "۰۰۰۰۰۰۰۰" باشد.

در صورتی که مقدار **binary**نامعتبر باشد، که به معنی هیچ یک از مقادیر ممکن نیست، خروجی **unary**به صورت "XXXXXXX")هفت عدد (X تنظیم می‌شود. این به معنی وضعیت نامعتبر یا نامشخص است.

به طور خلاصه، این کد یک مبدل سه بیتی به یک بیتی است که با استفاده از ساختار **case**، مقادیر ممکن برای **binary**را بررسی کرده و مقادیر متناظر در **unary**را تنظیم می‌کند.

توضیح کد تست:

کد تست سوال ۴، برای تست و اعتبارسنجی واحد `BINARY_TO_UNARY` طراحی شده است. این کد تست، ورودی‌های مختلف را به واحد `BINARY_TO_UNARY` ارسال می‌کند و خروجی متناظر را در `unary` ذخیره می‌کند. در ادامه، توضیحی از اجزای این کد تست آمده است:

- در قسمت تعریف `binary_to_unary_tb`، یک تست بنچ با نام `binary_to_unary_tb` تعریف شده است.

- در معماری `TB_ARCHITECTURE`، یک کامپوننت با نام `binary_to_unary` تعریف شده است که پورت‌های `binary` و `unary` را دارد. این کامپوننت تست شده را نمایش می‌دهد.

- سیگنال `binary` به عنوان ورودی و سیگنال `unary` به عنوان خروجی تعریف شده است.

- در بلوک `UUT`، واحد تست شده `binary_to_unary` با پورت‌های متناظر به `binary` و `unary` مپ می‌شود.

- در فرآیند `process`، مقادیر مختلف برای `binary` تنظیم می‌شوند و پس از گذشت زمان مشخص (۱۰۰ns)، مقدار `binary` تغییر می‌کند. سپس منتظر می‌ماند تا پردازش تمام شود.

- در قسمت `configuration`، تنظیمات برای تست بنچ انجام می‌شود. این کد تست `binary_to_unary_tb` را برای واحد `binary_to_unary` با استفاده از معماری `transform` فراخوانی می‌کند.

به طور خلاصه، این کد تست برای تست واحد `BINARY_TO_UNARY` طراحی شده است. با تغییر مقادیر `binary` و انتظار برای تکمیل فرآیند، خروجی متناظر در `unary` بررسی می‌شود و صحت عملکرد واحد تست شده بررسی می‌شود.

پایان