Program nauczania sponsorowany przez firmę INTEL

Advanced ASIC Design

Instrukcja Laboratoryjna

Ćwiczenie nr 5:
SPYGLASS CDC & RDC

1. SG Lint
2. VCS
3. FC Synteza i implementacja – Część 1
4. FC Synteza i implementacja – Część 2
5. FC Synteza i implementacja – Część 3
6. **SG CDC & RDC**

I. Synchronizator dwuetapowy (Two-stage synchronizer/FF synchronizer)

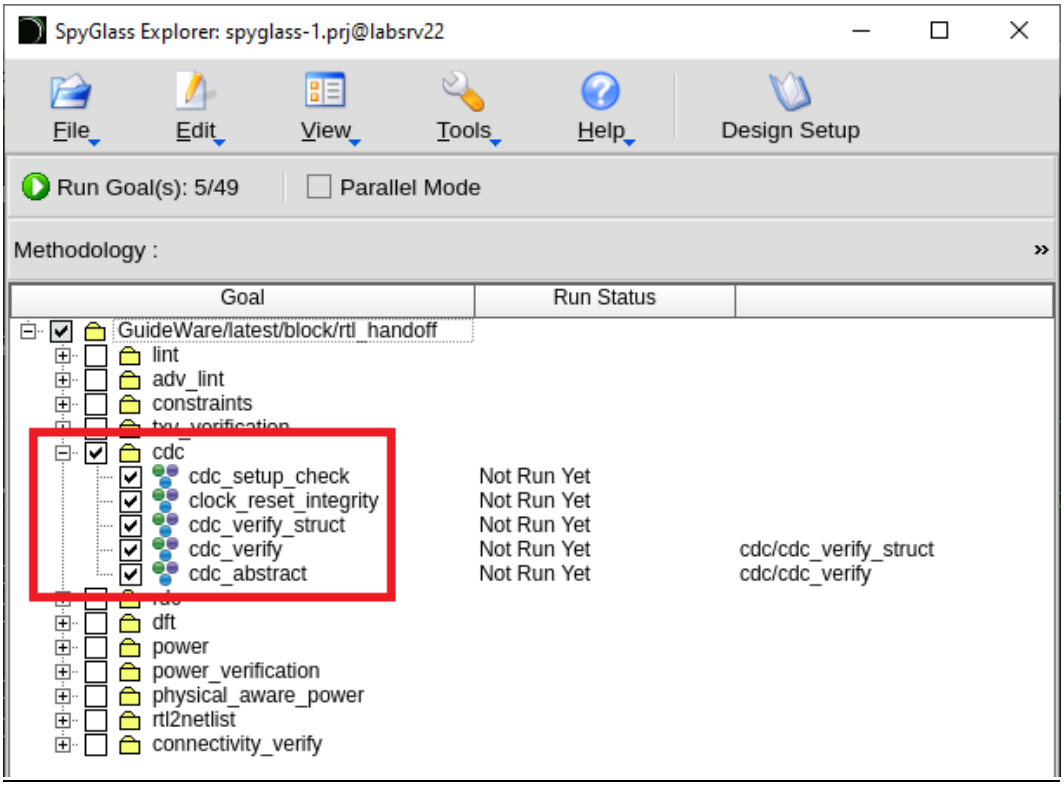
1. Zapoznanie się z przykładem synchronizatora

Pobrać z platformy zdalnej edukacji plik *„ff_synch_CDC.zip”*. Zawiera on model synchronizatora wraz z testbenchem pozwalającym na obserwacje jego działania oraz skrypt uruchamiający symulację (*„my_synch_sym_CDC.sh”*). Testbench pozwala na prostą zmianę częstotliwości zegara źródłowego (clk_a) oraz docelowego (clk_b) poprzez edycje parametrów **CLK_a_F_HZ**, **CLK_b_F_HZ**. Zapoznać się z plikiem *„my_synchornizer_CDC.v”*.

Uruchomić symulacje przy ustawianiach domyślnych (przejście z domeny zegarowej 50 MHz do 80MHz) i zaobserwować zachowanie układu. **W sprawozdaniu (Punkt 1) należy odpowiedzieć na pytanie: Czy na symulacji dane z domeny zegarowej A, są poprawnie przenoszone do domeny zegarowej B? Zamieść przebiegi czasowe potwierdzające tą odpowiedź.**

2. SpyGlass jako narzędzie do weryfikacji CDC

Narzędzie SpyGlass posiada reguły pozwalające na weryfikację projektowanych synchronizatorów CDC. Po dodaniu plików projektowych do tego narzędzia oraz ustawienia modułu nadrzędnego (w tym wypadku *„my_synchornizer_CDC”*), w zakładce „Goal setup” można wybrać reguły wspomagające projektowanie synchronizatorów CDC.



Rys. 1 Reguły CDC

Do poprawnej interpretacji problemów CDC, do narzędzia SpyGlass należy dostarczyć informację na temat sygnałów wejściowych sprawdzanego modułu. Dane te zapisuje się w pliku „*.sgdc” który następnie załącza się razem z plikami źródłowymi.

Minimalna zawartość tego pliku to informacje na temat sygnałów zegarowych oraz sygnałów resetujących. Informacje o dodatkowych ustawieniach możliwych do zawarcia w pliku „*.*sgdc”* można znaleźć w pomocy SpyGlassa (Przycisk „Help”). W rozważanym przykładzie plik ten będzie wyglądał następująco:

```
1  current_design my_synchronizer_CDC
2  clock -name my_synchronizer_CDC.i_clk_a -period 20 -edge {0 10}
3  clock -name my_synchronizer_CDC.i_clk_b -period 12.5 -edge {0 6.25}
4  reset -name my_synchronizer_CDC.i_rst_a -value 0
5  reset -name my_synchronizer_CDC.i_rst_b -value 0
```

*Rys. 2 Plik *.sgdc*

current_design “nazwa modułu” – to nazwa badanego modułu

clock – parametry zegara

-name ”scieżka” – lokalizacja portu zegara

-period „ns” – okres częstotliwości zegara w ns

-edge „{ns ns}” – wypełnienie przebiegu, pierwsza wartość to czas w wystąpienia w nanosekundach zbocza narastającego, a druga to czas wystąpienia zbocza opadającego. Przy okresie 10ns, {0 5} oznacza przebieg o wypełnieniu 50%.

reset – parametry resetu

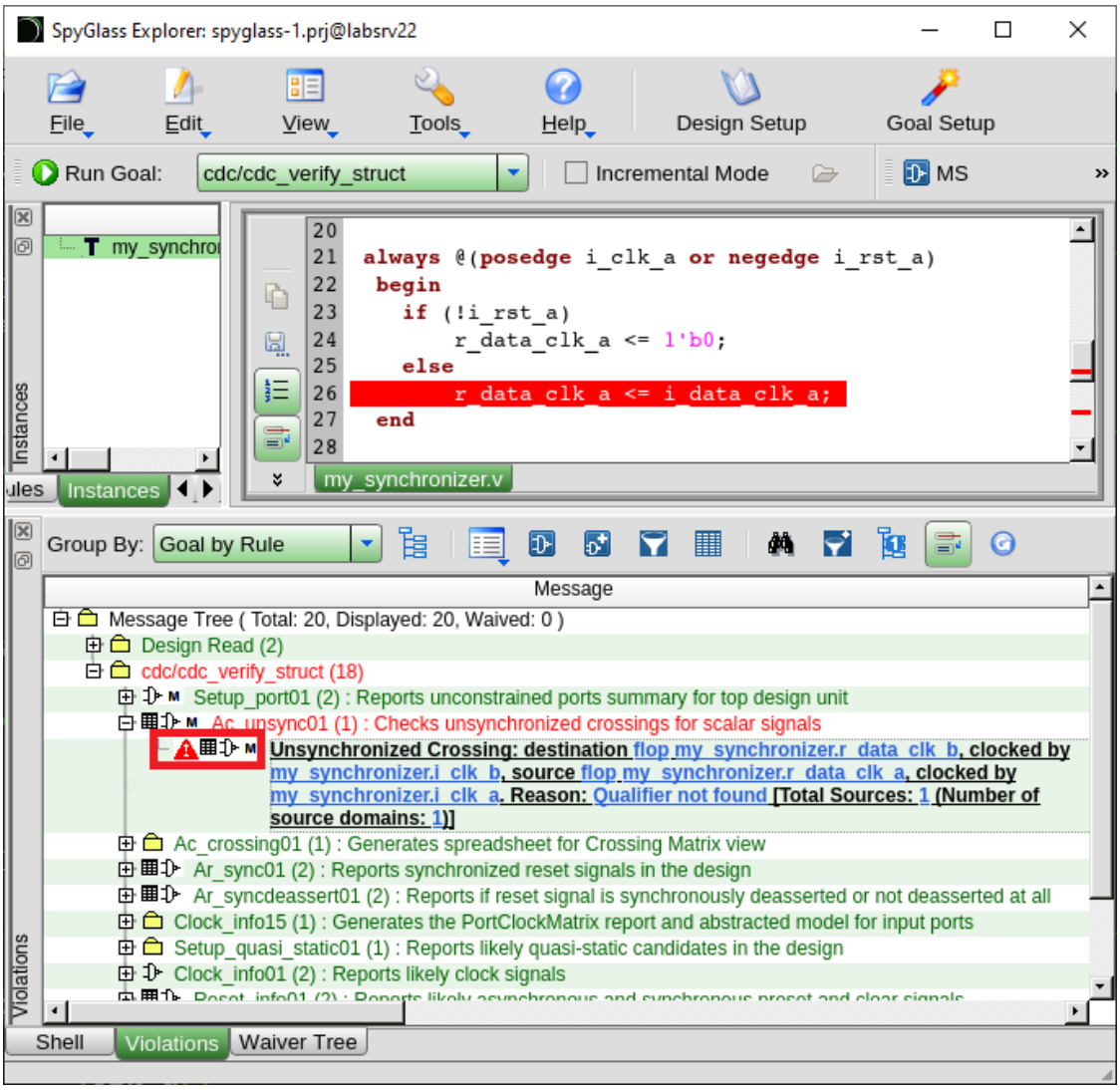
-name ”scieżka” – lokalizacja portu resetu

- value – określa czy reset jest aktywny przy zerze czy przy jedynce

Dodaj plik *„*.sgdc”* do projektu oraz uruchom sprawdzanie reguł CDC.

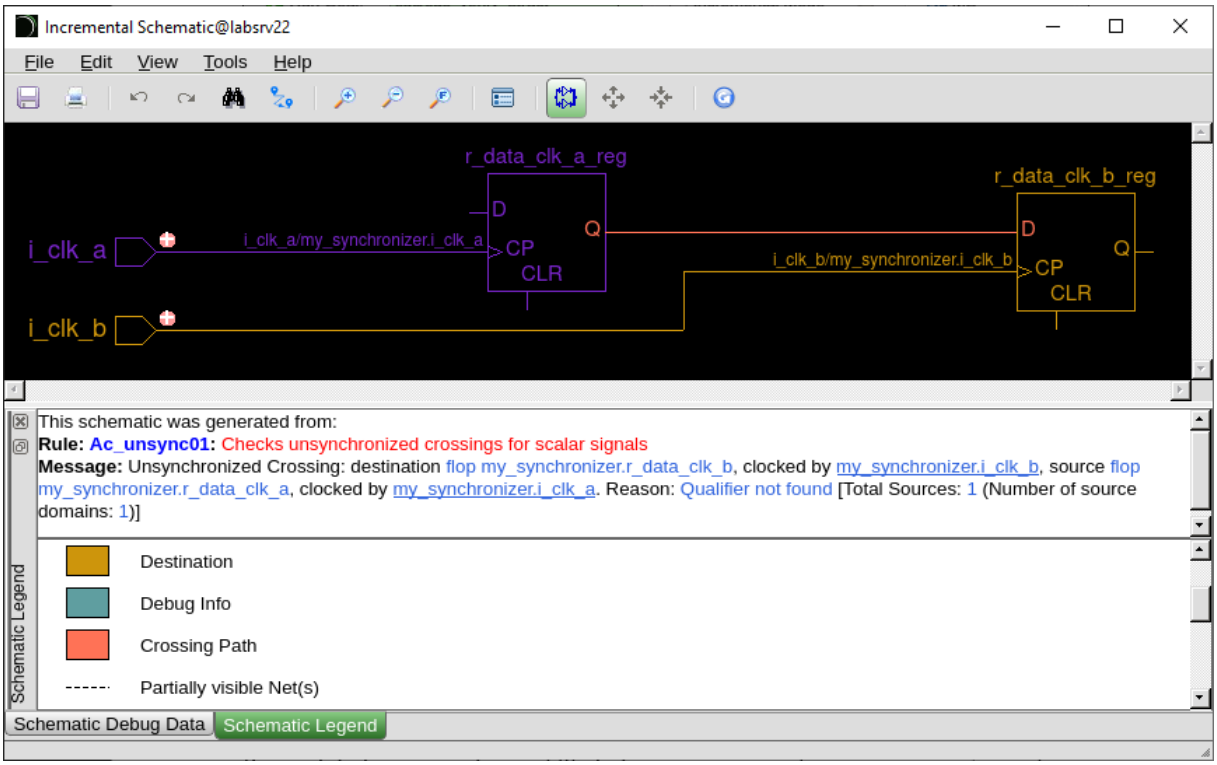
3. Analiza wyników

Po sprawdzeniu reguł w kategorii *„cdc_verify_struct”* narzędzie SpyGlass wyszuka naruszenie: *„Ac_unsync01”*. Proszę pamiętać o włączeniu okna pomocy (jeżeli nie jest włączone) *„View -> Windows -> Help Viewer”*, pozwoli ono wyświetlić pomoc dotyczącą aktualnego (zaznaczonego) błędu.



Rys. 3 Naruszenie Ac_unsync01

Po rozwinięciu naruszenia oraz kliknięciu na zaznaczony obszar można np. otworzyć schemat (po wybraniu *„Incremental Schematic”)* pokazujący miejsce braku synchronizacji:



Rys. 4 Schemat pokazujący miejsce braku synchronizacji

W sprawozdaniu (Punkt 2) należy opisać i wyjaśnić, dlaczego ten błąd występuje mimo tego, że na symulacji układ zachowuje się prawidłowo (na symulacji nie występują stany „X"). Następnie należy zmodyfikować układ modułu „*my_synchornizer_CDC*" tak, aby był on poprawnie zsynchronizowany. Następnie poprawiony kod umieścić w sprawozdaniu. Układ przetestować również na symulacji i udokumentować to poprzez dodanie przebiegów czasowych do sprawozdania.

4. Przejście z domeny zegarowej szybszej do wolniejszej

Zmodyfikować plik testbenchu *„my_synchornizer_CDC_tb.v"* tak, aby nastąpiło przejście z szybszej domeny zegarowej do wolniejszej (np. z 80MHz do 50MHz). Uruchomić symulację. Uruchomić narzędzie SpyGlass z zmienionym plikiem *„*.sgdc"*.

W sprawozdaniu (Punkt 3) Odpowiedzieć na pytania, dokumentując odpowiedzi screen'ami: Czy dane podczas symulacji zostały prawidłowo przekazane z domeny zegarowej A to domeny zegarowej B? Czy narzędzie Spyglass zgłosiło jakieś naruszenia? Na czym polega ten problem?

Projekt:

Na projekt proszę poświęcić max 2h. Jeżeli ten czas nie wystarczy proszę przejść dalej, a w sprawozdaniu opisać tylko to co udało się wykonać.

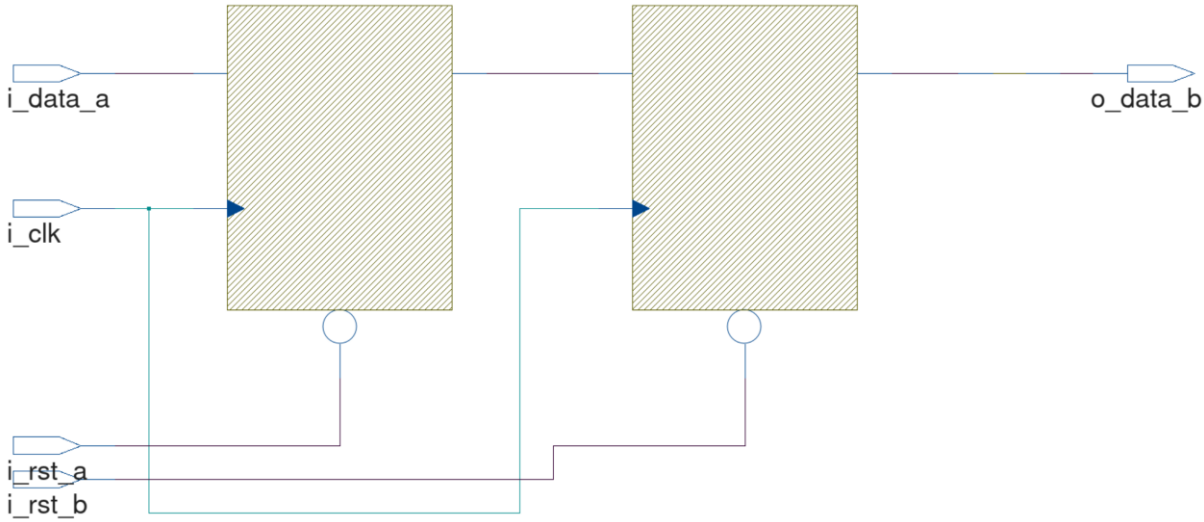
Zmodyfikować synchronizator (*„my_synchronizer_CDC"*) tak aby pozwalał na bezbłędną transmisję jednego bitu z szybszej domeny zegarowej do wolniejszej. W testbenchu znajduje się sygnał „next_data_clk_a". Pozwala on opóźnić wysyłanie nowych danych z domeny zegarowej A. Nowa dana zostanie zaktualizowana („data_a") tylko wtedy, gdy

sygnał „next_data_clk_a”, będzie miał wartość jeden podczas zbocza narastającego zegara A. Należy więc tak wysterować ten sygnał, żeby nie dochodziło do utraty danych wynikającej z różnej częstotliwości taktowania zegarów A i B.

W sprawozdaniu (Punkt 4) Umieścić kod zmodyfikowanego synchronizatora oraz przebiegi czasowe pokazujące prawidłową prace układu. Zamieścić zawartość utworzonego pliku *.sgdc. Następnie umieścić screen z narzędzia Spyglass pokazujący brak znalezionych naruszeń. Dodać krótki opis zaprojektowanego układu. Opisać wnioski i przemyślenia pochodzące z procesu projektowania układu.

5. Synchronizacja dwóch sygnałów reset w jednej domenie zegarowej

Pobrać z platformy zdalnej edukacji plik „ff_synch_RDC.zip. Zawartość oraz nazwa plików są bardzo podobna do pierwszego, powyższego przykładu. Przykład ten zawiera jednak jedną tylko jedną domenę zegarową, a problem dotyczy synchronizacji resetów przerzutników. Schemat pierwotnego rozwiązania umieszczono na Rys. 5.



Rys. 5 Synchronizacja resetu

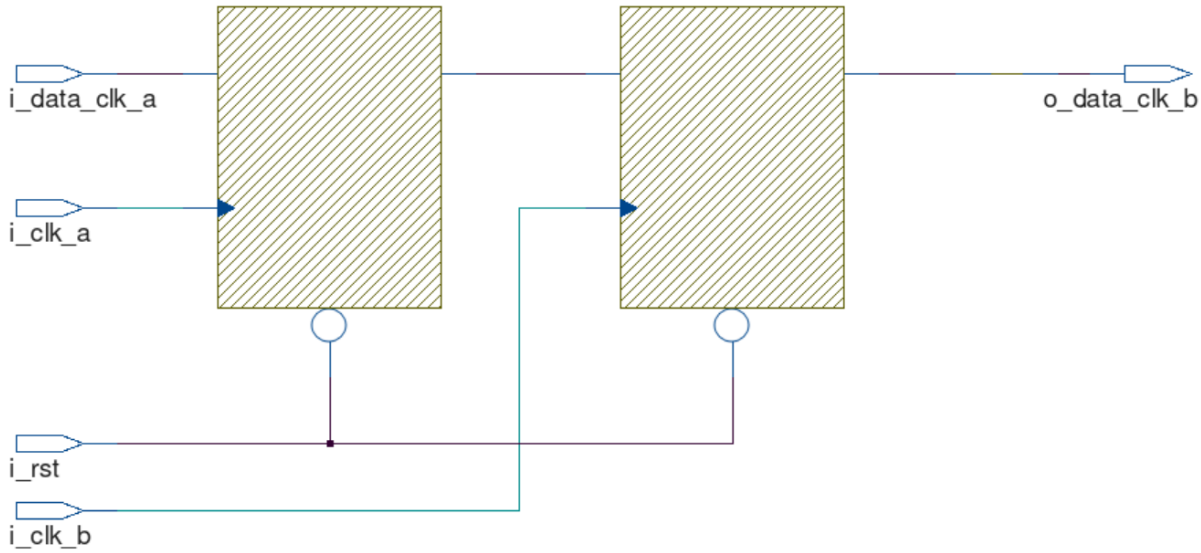
Pobrany przykład należy sprawdzić narzędziem Spyglass, uruchamiając wszystkie reguły CDC oraz RDC. Występujące naruszenia należy usunąć, dodając do projektu synchronizator dwuetapowy.

Rys. 6 - Reguły RDC

W sprawozdaniu (Punkt 5) Opisać, dlaczego wymagana jest synchronizacja resetów przerzutników, skoro resety w omawianych przykładach są asynchroniczne? Do sprawozdania powinny zostać dołączone zrzuty ekranu z narzędzia Spyglass, przedstawiające wykryte naruszenia RDC. Następnie należy zamieścić kod zaprojektowanego synchronizatora sygnału resetu oraz zrzut ekranu potwierdzający eliminację wcześniej zgłaszanych naruszeń po wprowadzeniu synchronizacji.

6. Dwie domeny zegarowe, jeden globalny asynchroniczny reset

Z platformy zdalnej edukacji należy pobrać plik „ff_synch_RDC2.zip". W dołączonym przykładzie przedstawiono przypadek synchronizacji sygnału pochodzącego z domeny zegarowej A do domeny zegarowej B. Cały układ powinien być resetowany jednym globalnym, asynchronicznym sygnałem resetującym. Do układu należy dodać synchronizator danych oraz układy generujące lokalne sygnały resetujące dla domeny zegarowej A i domeny zegarowej B. Schemat pierwotnego rozwiązania umieszczono na Rys. 7.

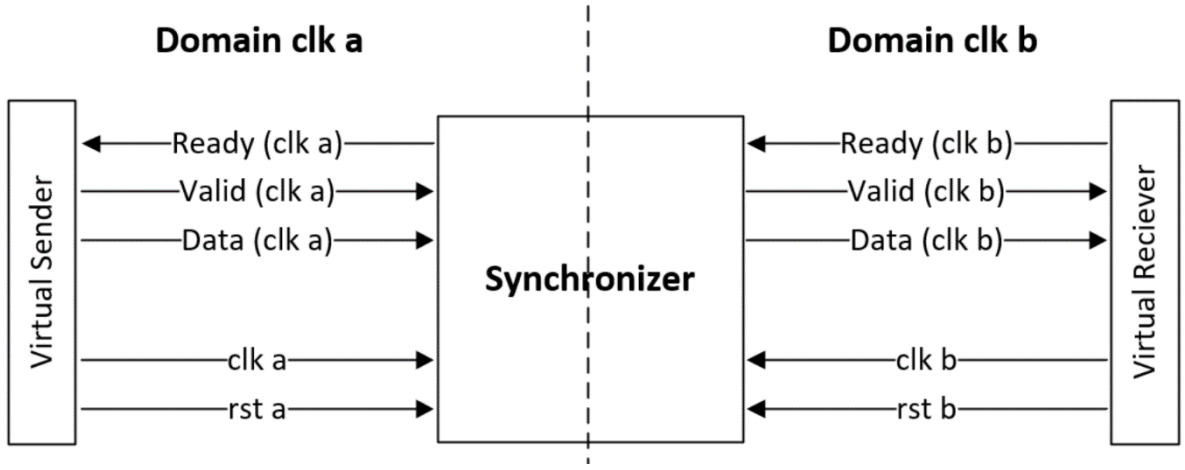


Rys. 7 Globalny sygnał reset

W sprawozdaniu (Punkt 6) należy wypisać błędy wygenerowane przez reguły CDC i RDC w oprogramowaniu SpyGlass, wraz z krótkim opisem sposobu ich usunięcia.

II. Protokół Valid/Ready

W kolejnej części laboratoriów będą omawiane mechanizmy przesyłania wielobitowych danych z jednej domeny zegarowej do drugiej. Aby ujednolicić sposób odbierania i nadawania danych, w tych przykładach będzie wykorzystywany protokół Valid/Ready. Badany synchronizator będzie posiadał interfejs wejściowy oraz wyjściowy V/R, a środowisko testowe wirtualny nadajnik i odbiornik V/R.



Rys. 8 Schemat synchronizatora wykorzystującego protokół Valid/Ready

Protokół ten dział na bardzo prostej zasadzie: Jeżeli odbiornik jest gotowy na odebranie danych ustawia sygnał „Ready" na wartość 1. Jeżeli nadajnik jest gotowy na wysłanie danych ustawia sygnał „Valid" oraz wysyłane dane na linię „Data". Transmisja następuje tylko i wyłącznie wtedy, gdy sygnał Valid i Ready są w stanie wysokim.

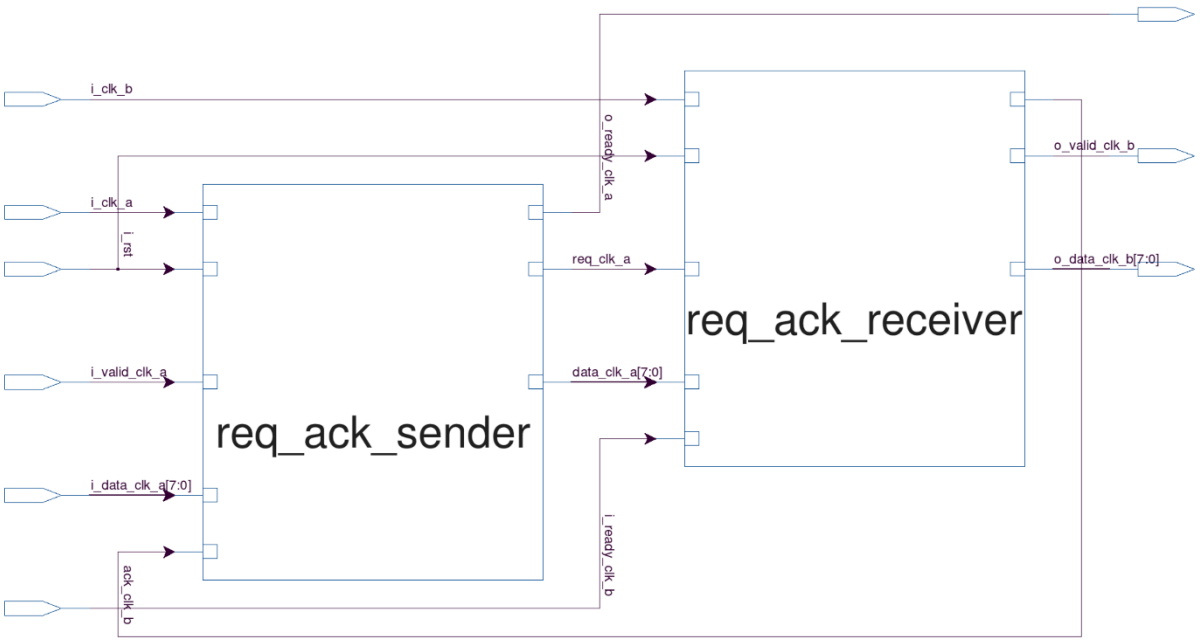
1. Valid/Ready testbench

Do zaprezentowania działania synchronizatorów zaprojektowane środowisko testowe zawierające wirtualny nadajnik i odbiornik V/R (Rys. 8). Zachowanie wirtualnych odbiorników i nadajników można zmieniać poprzez parametry opisane na początku testbenchu. Przy użyciu parametrów: MIN_DELAY_VALID_A oraz MAX_DELAY_VALID_A można ustawić opóźnienie przesłania danych przez wirtualny nadajnik. Jest to czas ustawienia flagi „Valid" oraz wystawienia nowych danych (w cyklach zegara) po prawidłowym przesłaniu danych. Czas ten losowany jest z przedziału tych parametrów (MAX i MIN). Jeżeli wartości te są takie same, parametr ten jest stały (nie jest on losowany). Przykładowo: jeżeli obie te wartości są ustawione na wartość „0" po każdym sygnale zwrotnym „Ready" (w następnym cyklu zegara) będzie ustawiana flaga „Valid" wraz z nowymi danymi. Nadajnik nie będzie miał więc żadnych opóźnień. W taki sam sposób można regulować opóźnienie wirtualnego odbiornika poprzez parametry MIN_DELAY_READY_B oraz MAX_DELAY_READY_B.

Podczas symulacji w konsoli wypisywane są informacji o tym jakie dane są wysyłane i odbierane oraz kiedy. Pozwoli to na proste porównanie szybkości i zachowania sprawdzanych synchronizatorów.

III. Synchronizator REQ/ACK

Pobrać z platformy zdalnej edukacji plik „req_ack.zip". Zawiera on model wielobitowego synchronizatora req/ack wraz z testbenchem pozwalającym na obserwacje jego działania oraz skrypt uruchamiający symulację („req_ack_sym.sh").



Rys. 9 Schemat pierwotnego rozwiązania REQ/ACK

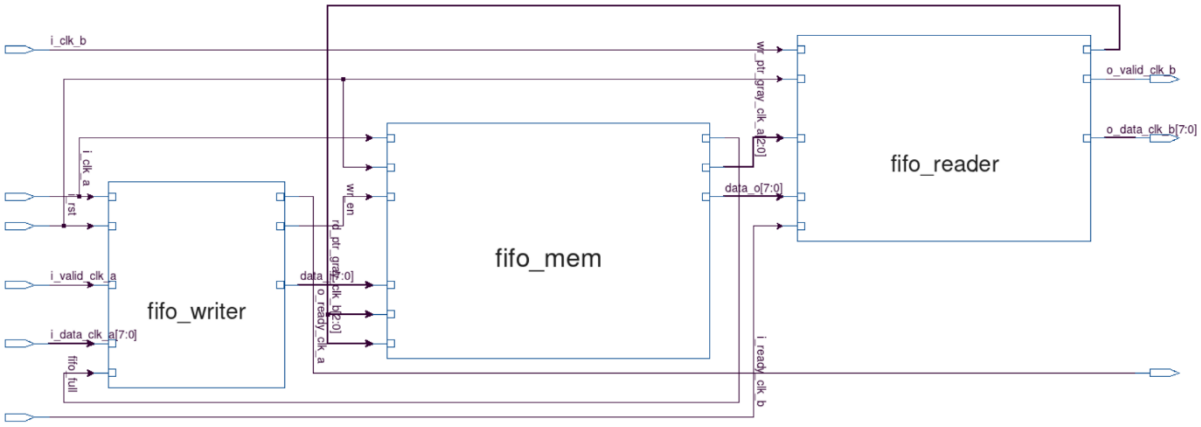
Uruchomić symulacje i zapoznać się z działaniem synchronizatora. Narzędziem SpyGlass wyszukać naruszenia CDC. Skorygować wykryte błędy.

Zapoznać się z działaniem skorygowanego synchronizatora w symulacji w różnych warunkach (poprzez zmianę parametrów częstotliwości zegarów oraz protokołu V/R).

W sprawozdaniu (Punkt 7) opisać działanie synchronizatora. Na czym polegał błąd wykryty przez narzędzie Spyglass? Jak go naprawić? Dlaczego synchronizatory dwuetapowe wymagane są tylko na linii ack i req? W sprawozdaniu udokumentować poprawne działanie synchronizatora poprzez: screeny symulacji oraz screeny oprogramowania SpyGlass.

IV. Synchronizator FIFO

Pobrać z platformy zdalnej edukacji plik „fifo.zip". Zawiera on model wielobitowego synchronizatora fifo wraz z testbenchem pozwalającym na obserwacje jego działania oraz skrypt uruchamiający symulację („fifo_sym.sh").



Rys. 10 Schemat pierwotnego rozwiązania FIFO

Uruchomić symulacje i zapoznać się z działaniem synchronizatora. Narzędziem SpyGlass wyszukać naruszenia CDC. Skorygować wykryte błędy.

Zapoznać się z działaniem skorygowanego synchronizatora w symulacji w różnych warunkach (poprzez zmianę parametrów częstotliwości zegarów oraz protokołu V/R). Proszę również sprawdzić wpływ głębokości pamięci FIFO na synchronizator.

W sprawozdaniu (Punkt 8) opisać działanie synchronizatora. Na czym polegał błąd wykryty przez narzędzie Spyglass? Jak go naprawić? Dlaczego wskaźniki FIFO są zapisywane w kodzie Greya? Jak dobierać głębokość FIFO do obciążenia synchronizatora? W sprawozdaniu udokumentować poprawne działanie synchronizatora poprzez: screeny symulacji oraz screeny oprogramowania SpyGlass.

V. Porównanie Synchronizatora REQ/ACK oraz FIFO

Proszę zaproponować kilka scenariuszy testowych (parametry testbencha) i uruchomić w nich obydwa synchronizatory.

W sprawozdaniu (Punkt 9) wypisać zaproponowane scenariusze (listy parametrów). Wynotować w postaci tabeli całkowity czas transmisji danych w każdym ze scenariuszy. Który z synchronizatorów jest szybszy? Porównać wady i zalety obydwóch synchronizatorów. Jakie ulepszenia można dodać do dostarczonych implementacji synchronizatorów, aby działały one szybciej?