Program nauczania sponsorowany przez firmę INTEL

Advanced ASIC Design

Instrukcja Laboratoryjna

Ćwiczenie nr 1:
SPYGLASS LINT

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. CDC

I. Zapoznanie z oprogramowaniem

Do uruchomienia narzędzi Synopsys (a więc również oprogramowania VCS) niezbędne jest ustawienie odpowiednich zmiennych i ścieżek systemowych. Zmienne te można ustawić poprzez uruchomienie przygotowanego wcześniej skryptu wywołując komendę:

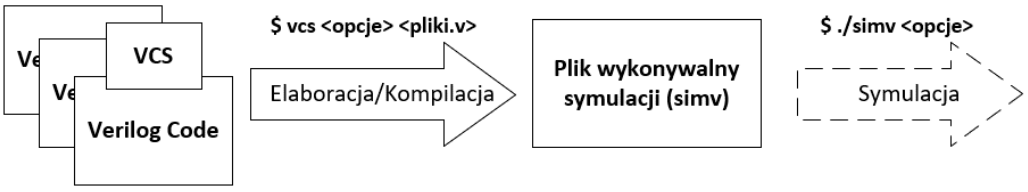
add-synopsys-FEV-all

Kompilator symulatora VCS nie posiada interfejsu graficznego i wywoływany jest z wiersza poleceń za pomocą komendy

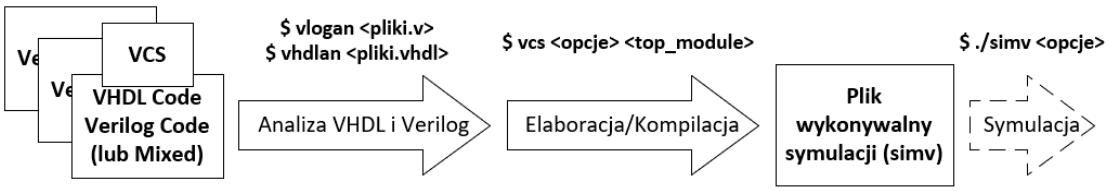
vcs

1. Sposób działania symulatora VCS

Komenda „*vcs*" służy do kompilacji projektu oraz do generacji plików wykonywalnych symulacji. Dopiero w kolejnym kroku taki plik wykonywalny (a więc symulację) można uruchomić. Plik wykonywalny symulacji można wygenerować na 2 sposoby, jednoetapowego flow oraz dwuetapowego flow:



Rys. 1 Kompilacja symulacji jednoetapowy flow



Rys. 2 Kompilacja symulacji dwuetapowy flow

Jednoetapowy flow jest możliwy, gdy wszystkie pliki projektowe to pliki Verliog.

Domyślna nazwa pliku wykonywalnego symulacji to „*simv*" który można uruchomić komendą:

./simv

Po użyciu tej komendy symulacja zostanie uruchomiona w konsoli tekstowej.

Komenda „*vcs*" posiada wiele przełączników, ich listę wraz z opisem można wyświetlić używając komendy „*vcs -help*". Większość tych przełączników jest również dostępne dla komend „*vlogan*" oraz „*vhdlan*". Najważniejsze z nich to:

*-f <scieżka_do_pliku *.f> - wczytanie pliku zawierającego listę plików projektowych*

-o <nazwa_pliku> - wybranie nazwy pliku wykonywalnego symulacji

-l <nazwa_pliku> - zapis logów kompilacji do wybranego pliku

-kdb – generowanie bazy danych Verdi (niezbędne do uruchomienia Verdi w trybie interaktywnym)

-debug_access+all – włączenie wszystkich opcji debugowania (niezbędne do uruchomienia Verdi w trybie interaktywnym)

Symulacje w trybie graficznym interaktywnym można uruchomić uruchamiając plik wykonywalny symulacji wraz z przełącznikiem *„./<nazwa_pliku_wkonywalnego> -gui (lub - verdi)”*. Należy jednak pamiętać, że do uruchomienia nakładki graficznej Verdi niezbędne jest skompilowanie symulacji z odpowiednimi przełącznikami.

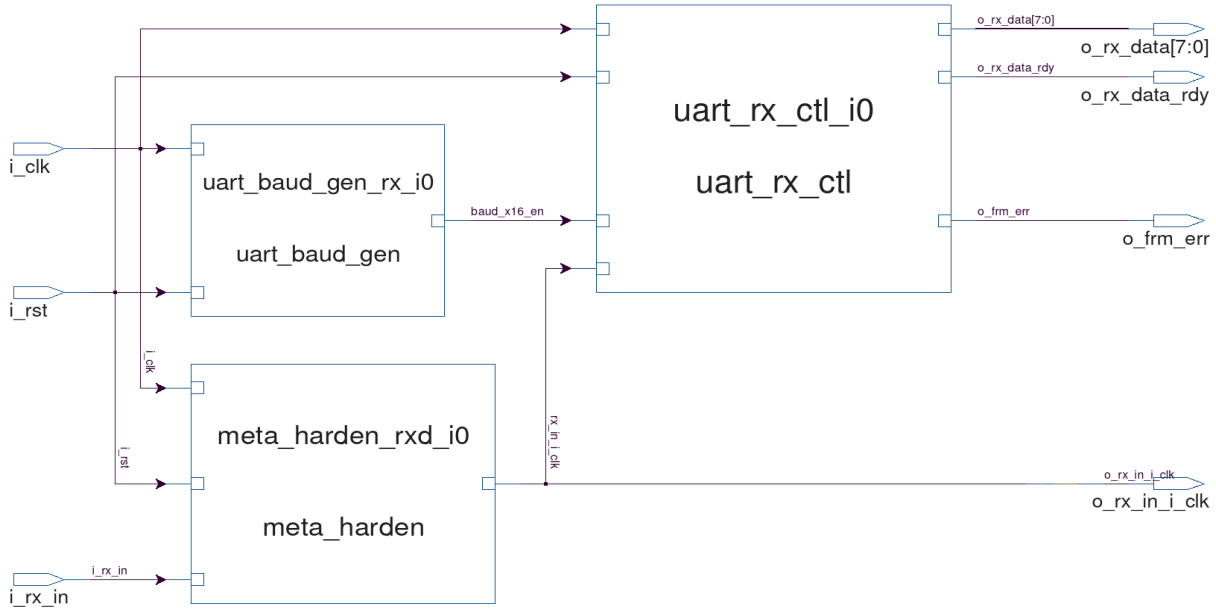
Wskazane jest stworzenie skryptu uruchamiającego symulacje, który przyśpieszy proces uruchamiania projektu. Skrypt taki powinien czyścić ewentualny poprzedni plik wykonywalny symulacji oraz poprzednie logi, uruchomić kompilacje symulacji z odpowiednimi przełącznikami oraz uruchomić symulację (standardowa nazwa takiego skryptu to „runsym.sh”).

II. Pliki dołączone do laboratorium

Do instrukcji laboratoryjne dołączono 5 plików:

uart_tb.v, uart_rx.v, uart_rx_ctl.v, uart_baund_gen.v oraz meta_harden.v

Pliki te zawierają model parametrycznego odbiornika magistrali szeregowej UART wraz z testbench’em pozwalającym zaprezentować jej działanie. **Proszę wstępnie zapoznać się z modelem zawartym w powyższych plikach.**



Rys. 3 Schemat modelu odbiornika UART

W sprawozdaniu proszę wyjaśnić jaka jest funkcja modułu „*meta_harden*” w projekcie odbiornika UART.

III. Symulacja w trybie tekstowym

Przy użyciu informacji zawartych w rozdziale I należy napisać skrypt uruchamiający symulacje w trybie tekstowym. W pliku *„uart_tb.v”* w liniach 94 -98 umieszczono monitor wyświetlający odebrane dane w konsoli:

```
94      always @ (posedge data_rdy) //data monitor
95       begin
96           $display("Data received on UART interface:");
97           $display("Data received (hex, ASCII) = %h at time: %t", data_out, $time);
98       end
```

Rys. 4 UART data received monitor

Po poprawnym uruchomieniu symulacji, w konsoli tekstowej zostanie wyświetlony ciąg odebranych danych.

W sprawozdaniu zaprezentować napisany skrypt wraz z opisem uruchamiający symulacje w trybie tekstowym oraz screen zawierający dane wyświetlone podczas symulacji.

Następnie w pliku *„uart_tb.v”* należy dopisać podobny monitor, wyświeltający dane wysyłane w przez testbench w formie szeregowej (sygnał *„rx_in”*). Oznaczyć w symulacji bit startu i Stopu.

```
Data received (hex, ASCII) = 44 at time:           342560000
Stop Bit, at time:           347200000
Start Bit, at time:           355880000
Data send = 0, at time:           364560000
Data send = 0, at time:           373240000
Data send = 0, at time:           381920000
Data send = 0, at time:           390600000
Data send = 0, at time:           399280000
Data send = 1, at time:           407960000
Data send = 0, at time:           416640000
Data send = 0, at time:           425320000
Data received on UART interface:
Data received (hex, ASCII) = 20 at time:           429500000
Stop Bit, at time:           434000000
Start Bit, at time:           442680000
```

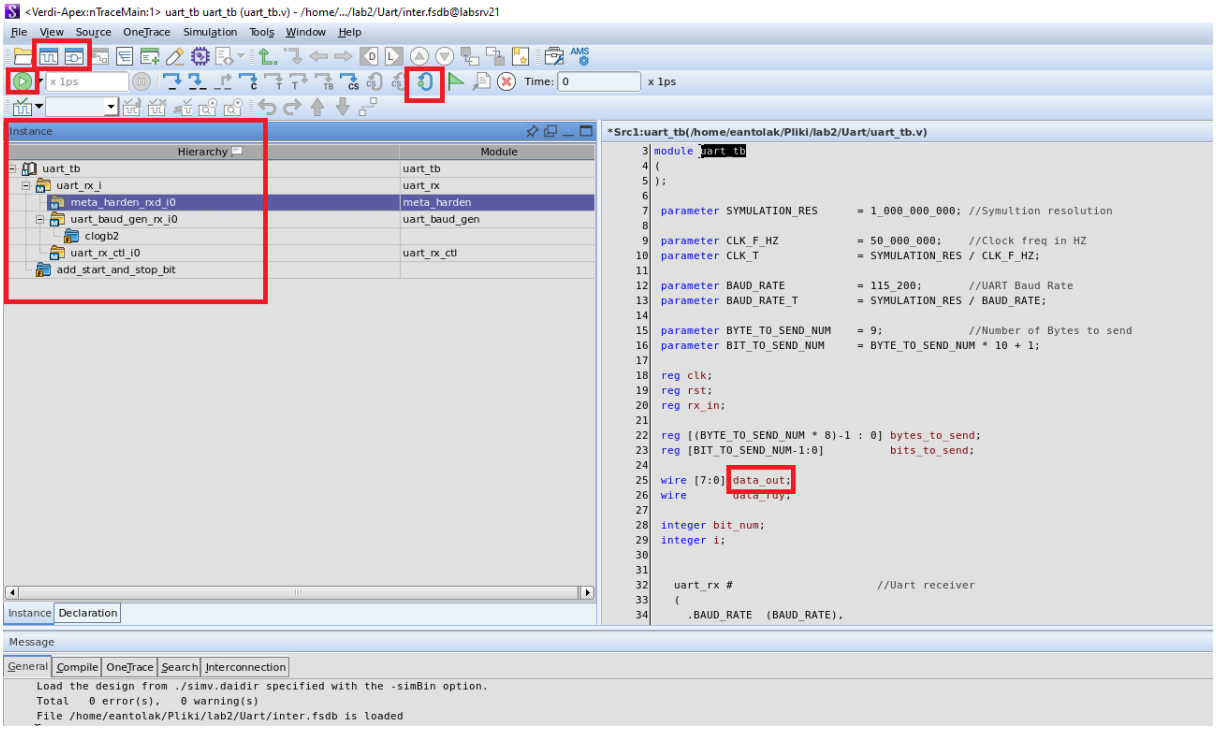
Rys. 5 Przykładowy log symulacji uwzględniający dodany monitor "rx_in"

Porównać kilka przesłanych bajtów z bajtami odebranymi. **W sprawozdaniu umieścić kod tego monitora wraz z informacjami wyświetlanymi podczas symulacji. Czy odbiornik UART działa poprawnie?**

IV. Symulacja w interaktywnym trybie graficznym

Symulacje VCS można otworzyć w trybie graficzny, dzięki nakładce graficznej Verdi. Przy użyciu informacji zawartych w rozdziale I należy napisać skrypt uruchamiający symulacje w trybie graficznym. **W sprawozdaniu należy zaprezentować napisany skrypt wraz z opisem uruchamiający symulację w trybie graficznym.**

Po uruchomieniu prawidłowego skryptu, zostanie otwarty graficzny interfejs Verdi:

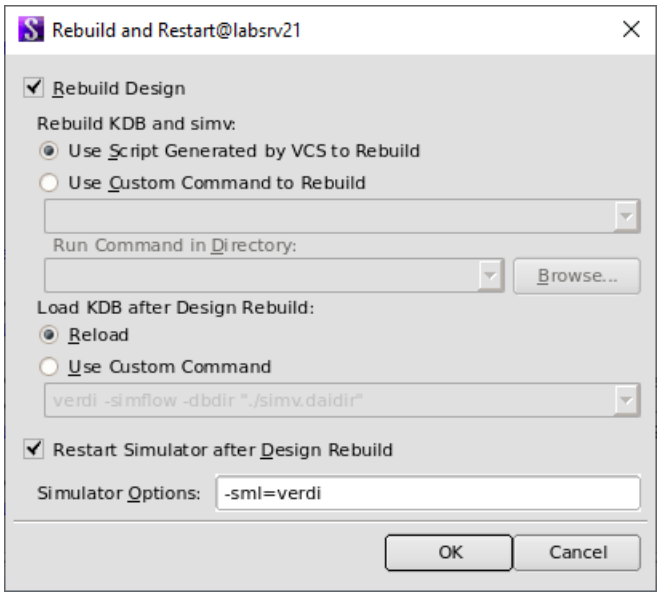


Rys. 6 Interfejs graficzny Verdi

Obsługa Verdi jest bardzo intuicyjna. Sygnały na przebieg czasowy możemy dodać klikając prawym klawiszem myszki na daną instancje i wybierając opcje *„Add to Waveform”*. Opcja ta działa również, gdy klikniemy na poszczególne sygnały w okienku podglądu kodu (wystarczy kliknąć na nazwę sygnału w kodzie) oraz oknie schematu. Jeżeli chcemy otworzyć inny plik tekstowy do podglądu, wystarczy kliknąć dwa razy na jego instancje w okienku *„Hierarchy”*.

Symulację uruchamiamy ikoną trójkąta na zielonym tle (lewy górny róg okna) a restartujemy drugą zaznaczoną ikoną w tej linii. Po użyciu zaznaczonych ikon w rzędzie wyżej można uruchomić nowy przebieg czasowy i widok schematu.

Główne okno podglądu plików tekstowych jest oknem tylko do odczytu, dlatego edytując pliki najlepiej skorzystać z zewnętrznego edytora. Zmiana kodu plików projektowych wymaga przebudowania symulacji. Nie trzeba robić tego ręcznie z poziomu konsoli, a można wykorzystać interfejs graficzny i zakładkę *„Simulation” -> „Rebuild and restart”* (pasek zakładek na samej górze ekranu).

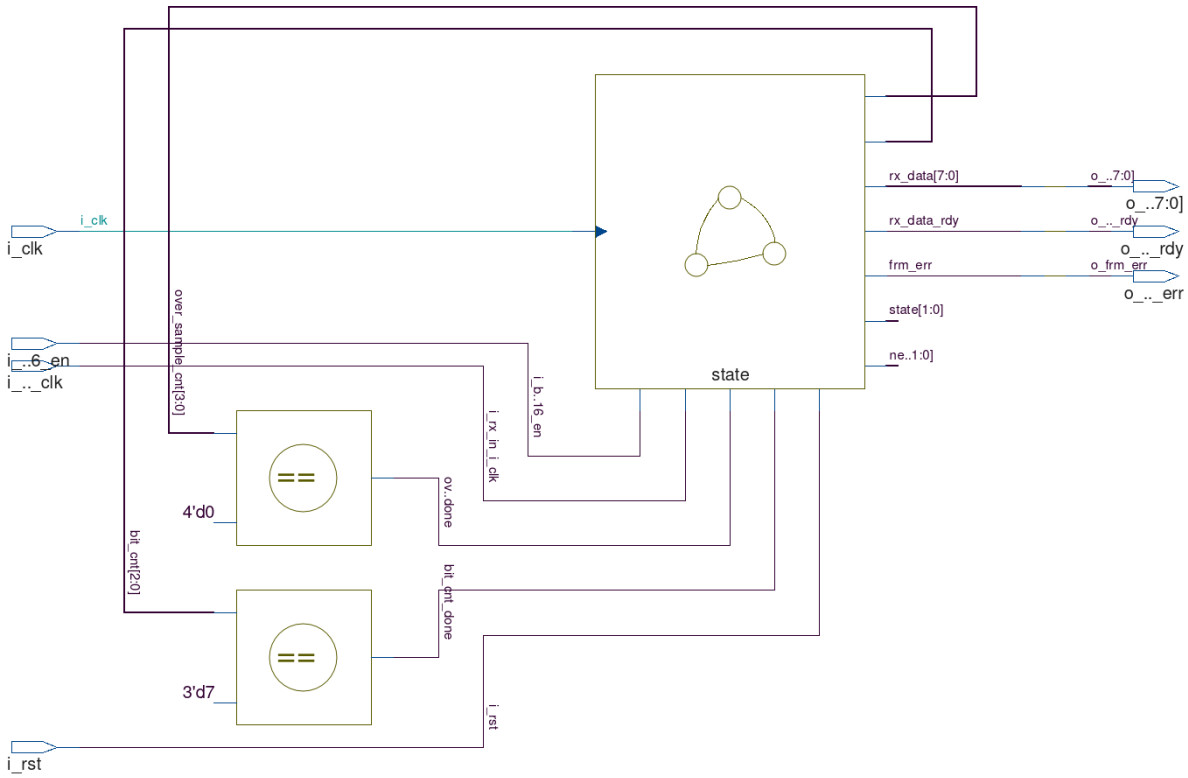


Rys. 7 Okno Rebuild and restart

Jeżeli nie używamy żadnych dodatkowych skryptów (sytuacja standardowa) otwarte okno należy ustawić tak jak na Rys. 7.

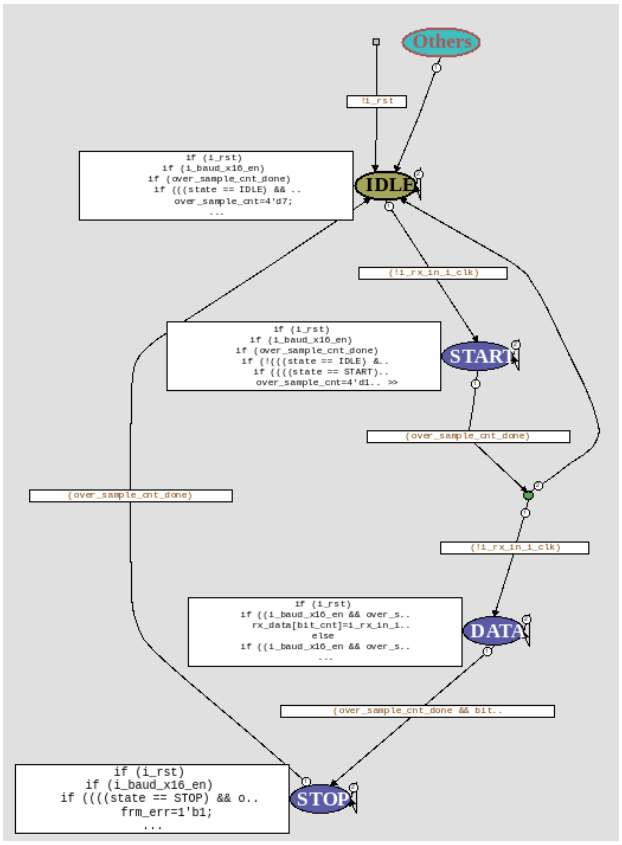
Następnie należy utworzyć przebiegi czasowe zawierające sygnały *„rx_in”*, *„data_out”* oraz *„data_rdy”*. Porównać kilka przesłanych bajtów z bajtami odebranymi. **W sprawozdaniu zamieścić screen zawierający powyższe przebiegi czasowe. Czy odbiornik UART działa poprawnie?**

W module *„uart_rx_ctl”* znajduje się automat skończony (finite state machine - FSM) kontrolujący etapy odbierania bitów transmisji szeregowej. Narzędzie Verdi pozwala na prostą analizę takich automatów. Po otarciu widoku schematu („New Schematic” klikając prawym klawiszem myszy na moduł na Rys. 6) tego modułu (*„uart_rx_ctl”*) zostanie na nim wyświetlony blok FSM.



Rys. 8 Blok FSM w module uart_rx_ctl

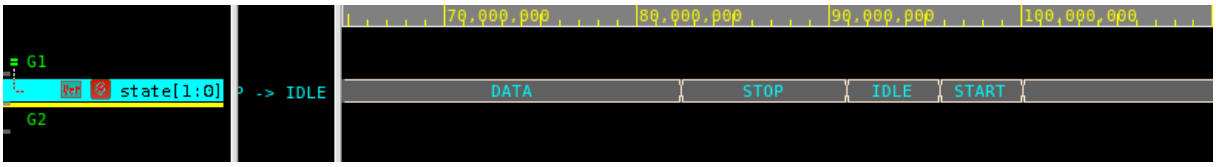
Klikając dwukrotnie na ten blok, zostanie otwarty graf badanego automatu:



Rys. 9 Graf automatu

Widoczność dodatkowych informacji w widoku schematu oraz FSM, zawsze można włączyć poprzez zakładkę *„View”*.

Dodatkowo na przebiegach czasowych zawierających sygnał „state” (z modułu *„uart_rx_ctl”)* automatycznie zostaną wczytane etykiety identyfikujące poszczególne stany automatu:

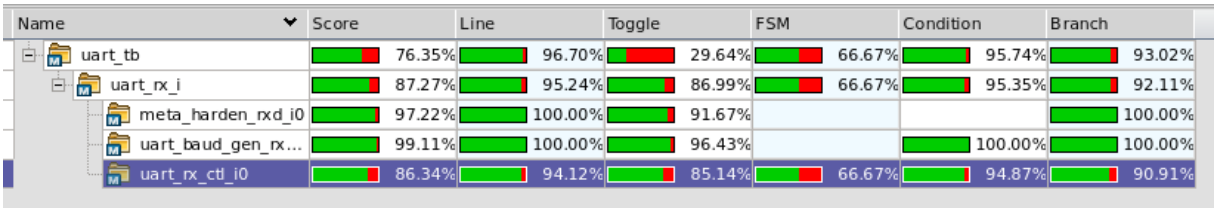


Rys. 10 Stany automatu

Proszę teraz poświęcić trochę czasu i zapoznać się z interfejsem Verdi.

V. Pokrycie symulacji

W środowisku Verdi oraz VCS możliwe jest również sprawdzenie, w jakim stopniu uruchamiana symulacja pokrywa sprawdzane w niej moduły. Możliwe jest np. sprawdzenie czy automaty skończone przechodzą przez wszystkie zaprojektowane stany.



Rys. 11 Pokrycie symulacji dla przypadku rozważanego na zajęciach.

Aby uruchomić pokrycie symulacji, niezbędne jest stworzenie nowego skryptu. W pierwszym kroku należy skompilować pliki wykonywalne symulacji uwzględniające pokrycie, można to zrobić następującym przełącznikiem komendy *„vsc”*:

-cm line+tgl+cond+fsm+branch

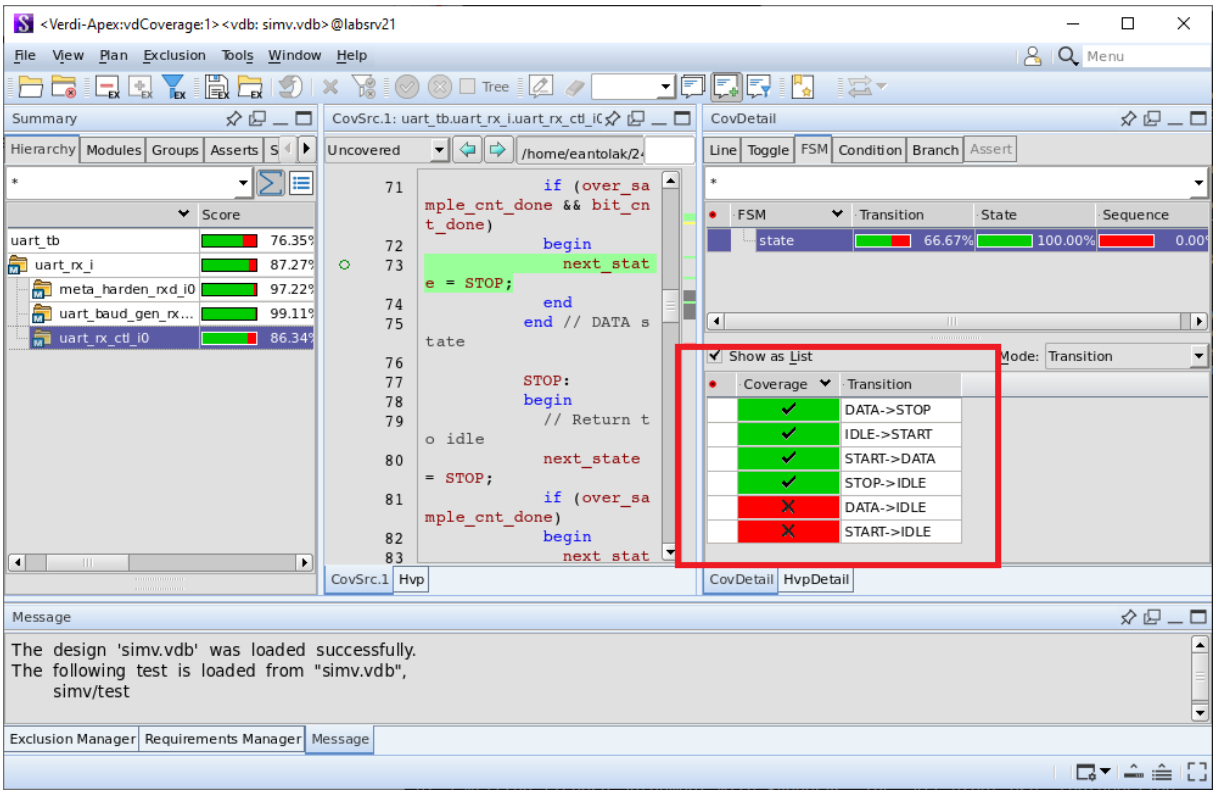
Symulacje należ uruchomić w trybie tekstowym (standardowo *„./simv”)* również używając powyższego przełącznika.

Kolejnym krokiem jest wygenerowanie raportu z symulacji. Można to zrobić poniższym poleceniem

urg -dir simv.vdb (gdzie simv.vdb to plik wygenerowany podczas symulacji)

W ostatnim kroku uruchamiamy raport w narzędziu Verdi:

verdi -cov -covdir simv.vdb



Rys. 12 Pokrycie FSM w module "uart_rx_ctl"

Z powyższego raportu można przykładowo odczytać, że automat stanów skończonych w module „uart_rx_ctl" nigdy nie przechodzi ze stanu DATA do stanu IDLE.

W sprawozdaniu zamieścić skrypt uruchamiający pokrycie symulacji oraz screen dokumentujące uruchomienie tego narzędzia. Wyjaśnić, dlaczego wyżej wymieniony automat nigdy nie przechodzi ze stanu DATA do stanu IDLE. Jakie jeszcze informacje można wyczytać z tego oprogramowania? Do czego te informacje mogą być przydatne?

VI. Projekt

Korzystając z powyższych informacji, zaprojektuj nadajnik UART współpracujący z powyższym modelem odbiornika. **Nadajnik powinien zawierać FSM (aby uzyskać ocenę 5).**

Nadajnik powinien znajdować się w zaprojektowanym module *„uart_tx"* (top_module nadajnika). Moduł ten powinien mieć przynajmniej następujące wejścia i wyjścia:

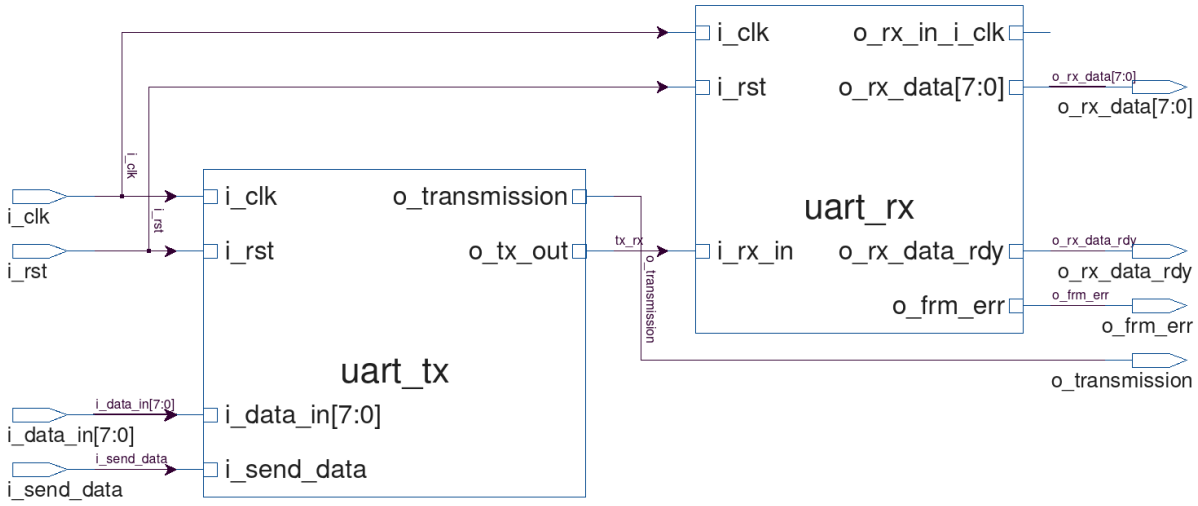
i_data_in [7:0] – port wejściowy, na który podawane są informacje do wysłania

i_send_data - port wejściowy, stan wysoki ma spowodować rozpoczęcie transmisji, następnie stan portu ma być ignorowany aż do zakończenia transmisji

o_transmission – port wyjściowy, stan wysoki zwraca informacje, że trwa transmisja danych. Stan niski zwraca informację, że nadajnik jest gotowy do wysyłania danych

o_tx_out – port wyjściowy, transmisja szeregowa informacji z i_data_in

Nadajnik i odbiornik powinien być połączony w module uart_top:



Rys. 13 Moduł uart_top

<u>Napisany kod należy sprawdzić narzędziem SpyGlass (*rtl_handoff*).</u> Do uruchomienia symulacji niezbędna będzie jeszcze mała zmiana testbench'u, pozwalająca na wysłanie przez projektowany moduł przynajmniej jednego bajtu danych.

<u>W sprawozdaniu należy załączyć kod projektowanego nadajnika, wraz ze screen'em pokazującym jego prawidłową prace (*sygnały i_send_data, o_transmission, o_tx_out, i_rx_in, o_rx_data oraz o_rx_data_rdy*) oraz pokrycie symulacji.</u>