Programowanie Układów FPGA

Piotr Onyszczuk, Michał Milewski

Maj 2020

1 Wstęp

Przedmiotem projektu realizowanego na przedmiot Programowanie Układów FPGA jest stworzenie prostego układu UART RX i TX działający jako kalkulator z działaniami: +,-. Do układu dane wprowadzane są jako wartości ASCII, każda cyfra liczby jest oddzielnym znakiem. Jako wartość zwrotna przekazywany jest std_logic_vector.

2 Akceptowane wyrażenia

Jako wejście akceptowane są ciągi znakowe stanowiące wyrażenia matematyczne na przykład:

- liczba 521 jest akceptowana jako ciąg '5' '2' '1'
- wyrażenie $521 + 3$ jest akceptowane jako ciąg '5' '2' '1' '+' '3'
- wyrażenie $521 + 3 =$ jest akceptowane jako ciąg '5' '2' '1' '+' '3' '=' – **jest już ono akceptowane jako poprawne wyrażenie dla którego zostanie zwrócony wynik dodawania**

Podczas podawania argumetów dozwolone są znaki:

- '0' '1' '2' '3' '4' '5' '6' '7' '8' '9' - znaki numeryczne tworzące liczbę
- '+' '-' - znaki identyfikujące działanie między kolejnymi liczbami lub opcjonalnie znak liczby,
- '=' - znak zakańczający wyrażenie rozpoczynający ewaluację,
- '(' ')' - głównym i jedynym celem istnienia i akceptacji znaków nawiasów jest zapewnienie możliwości podania liczby ujemnej jako jednego z argumentów działania. Jedynym dozwolonym użyciem tych znaków jest: $(42) + (-4) + 3 - (+4) =$

Zabronione są następujące wyrażenia (skutkują błędem):

- '+' lub '-' pod nawiasem, w innym celu niż oznaczenie znaku liczby, np. $(11 + 4) =$,
- zagnieżdżone nawiasy, np. $((1) + 4)$ lub $((-5))$,
- niepoprawne nawiasy, np. $)14 + 6 =$ lub $51 + (62 =$,
- znaki identyfikujące działanie jeden bezpośrednio po drugim (nie rozdzielone liczbą), za wyjątkiem oznaczenia znaku liczby np. $14 + -4 =$ lub $941 + +41 - 1 =$

3 Opis systemu

Zaimplementowany system działa w następujący sposób:

1. Układ TX otrzymuje żądane wyrażenie w postaci kolejnych znaków
2. Układ TX każdy otrzymany znak szeregowo przesyła do układu odbierającego RX
3. Układ RX szeregowo odbiera znak i porzekazuje go do układu liczącego(dalej zwany Kalkulator)
4. Kalkulator z odebranych znaków składa liczby i wyrażenia
5. Kalkulator po otrzymaniu znaku zakończenia wyrażenia('=') rozpoczyna proces ewaluacji(liczenia) wyrażenia
6. Kalkulator zwraca w obliczony wynik jako std_logic_vector
7. Wynik z powrotem przekazywany jest przez TX i RX.

Typy, z których korzystają poszczególne komponenty umieszczone są w pakiecie package_types ('package_types.vhd')

3.1 Układ TX ('sender.vhd')

Układ nadający ma za zadanie szeregowo nadać sygnał, który otrzymuje na wejściu.

Jest on parametryzowany długością słowa wejściowego (WORD_LEN), częstotliwością zegara (CLOCK_SPEED), częstotliwością nadawania (BOD) i liczbą bitów stopu (STOP_LEN).

Przyjmuje daną wejściową (D), sygnał zegarowy (clock (C) i reset (R)) i flagę nakazującą nadawanie (START).

Jego wyjściem jest sygnał szeregowy (TX), informacja o nadawaniu (TRANSMITTING), informacja o zakonczeniu nadawania (DONE) i sygnały pomocnicze do obserwowania: TIMER_OUT - licznik zegara, STATUS_OUT - aktualny status, BIT_NUMBER - pozycja w słowie.

Cykl życia układu nadającego składa się z następujących stanów:

- 'czekaj': Układ czeka na pojawienie się flagi START, przechodzi do stanu 'data',
- 'data': Układ nadaje kolejne bity słowa odczekując wymagane przerwy, przechodzi do stanu 'parzystosc'
- 'parzystosc': Układ nadaje 'bit parzystosci', odczekuje, przechodzi do stanu 'stop',
- 'stop': Układ ustawia flagę DONE, nadaje zdefiniowanę liczbę bitów stopu - '0', odczekując wymagane przerwy, przechodzi do stanu 'czekaj'

3.2 Układ RX ('receiver.vhd')

Układ odbierający ma za zadanie wypisać równolegle odebrany sygnał szeregowy.

Jest on parametryzowany długością słowa wejściowego (WORD_LEN), częstotliwością zegara (CLOCK_SPEED), częstotliwością nadawania (BOD) i liczbą bitów stopu (STOP_LEN).

Przyjmuje sygnał szeregowy (RX), sygnał zegarowy (clock (C) i reset (R)) i flagę informującą o rozpoczęciu nadawania (START).

Jego wyjściem jest odebrana dana (D), informacja o błędzie (ERROR), informacja o zakonczeniu odbierania (DONE) i sygnały pomocnicze do obserwowania: TIMER_OUT - licznik zegara, STATUS_OUT - aktualny status, WRITING - informacja czy moduł pisze do danej D.

Cykl życia układu nadającego składa się z następujących stanów:

- 'czekaj': Układ czeka na pojawienie się flagi START, przechodzi do stanu 'data',
- 'data': Układ odczytuje kolejne bity słowa odczekując wymagane przerwy, przechodzi do stanu 'parzystosc'
- 'parzystosc': Układ odczytuje 'bit parzystosci', sprawdza jego poprawnosc, odczekuje, przechodzi do stanu 'stop',
- 'stop': Układ odczytuje zdefiniowanę liczbę bitów stopu - '0', odczekując wymagane przerwy, ustawia flagę DONE, przechodzi do stanu 'czekaj'

3.3 Kalkulator ('kalkulator.vhd')

Układ kalkulatora parsuje kolejno odebrane znaki do wyrażenia arytmetycznego i oblicza jego wartość.

Jest on parametryzowany długością słowa wejściowego - pojedynczego znaku (WORD_LEN), maksymalną liczbą argumentów zadania (MAX_ARGS) i długością wyniku (WORD_LEN_RES).

Przyjmuje daną wejściową (CALC_D_IN), sygnał zegarowy (clock (C) i reset (R)), flagę potwierdzenia odebrania rezultatu (RECEIVED) i flagę nadawania (PASS).

Jego wyjściem jest rezultat obliczeń (RESULT), flaga zakończenia obliczń (DONE) oraz sygnały pomocnicze do obserowania: STATUS_OUT - aktualny status, ARGS_OUT - tablica argumentów, OPERATIONS_OUT - tablica operacji oraz ERR_OUT - flaga wykrycia błędu w składni

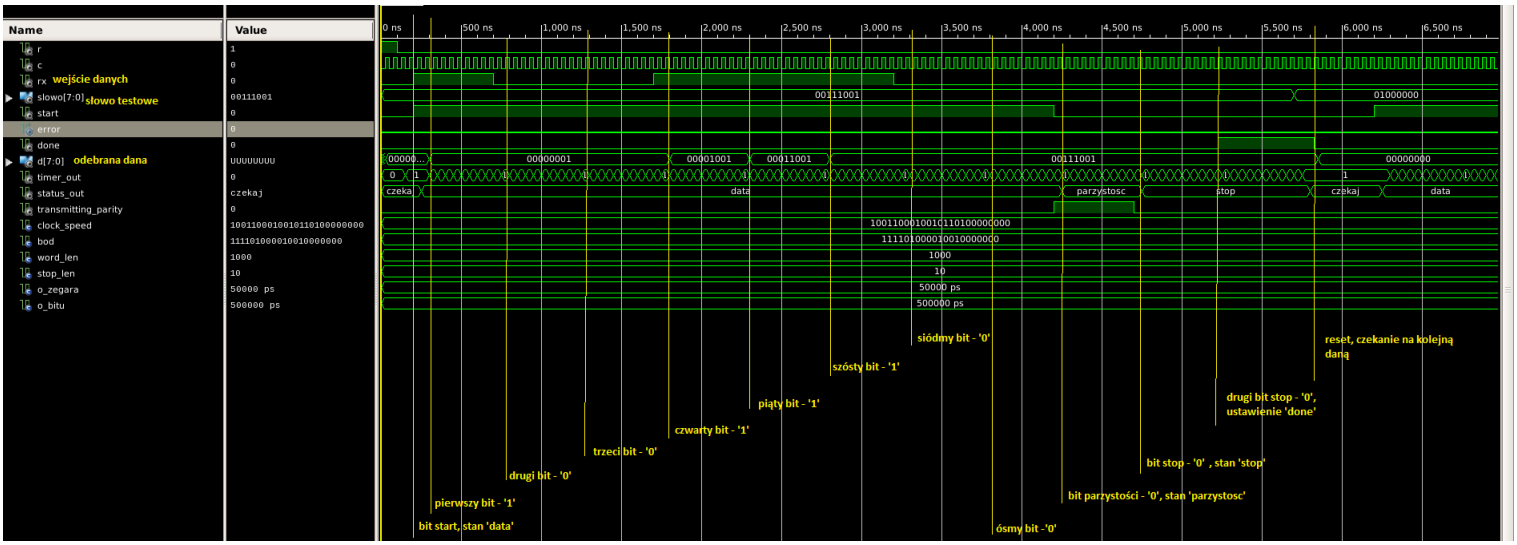
Cykl życia kalkulatora składa się z następujących stanów:

- 'ARGUMENTY': Układ oczekuje na kolejne argumenty aż do otrzymania znaku '=', wtedy przechodzi do stanu 'WYNIK'. Po drodze zapisuje argumenty i operatory do tablic oraz waliduje poprawność składni.
- 'WYNIK': Układ, jeśli wynik nie został już obliczony, bazując na otrzymanych argumentach oblicza wartość przekazanego wyrażenia. Przed obliczeniem sprawdzana jest zgodność ilości liczb i operatorów. Wartość obliczana jest za pomocą pętli, gdzie w zależności od kolejnych operatorów kolejne liczby są dodawane lub odejmowane od wyniku. Po przejściu przez wszystkie argumenty, wynik('RESULT') oraz flaga sygnalizująca obliczenie wyniku('DONE') zostają ustawione.

3.4 Symulacje

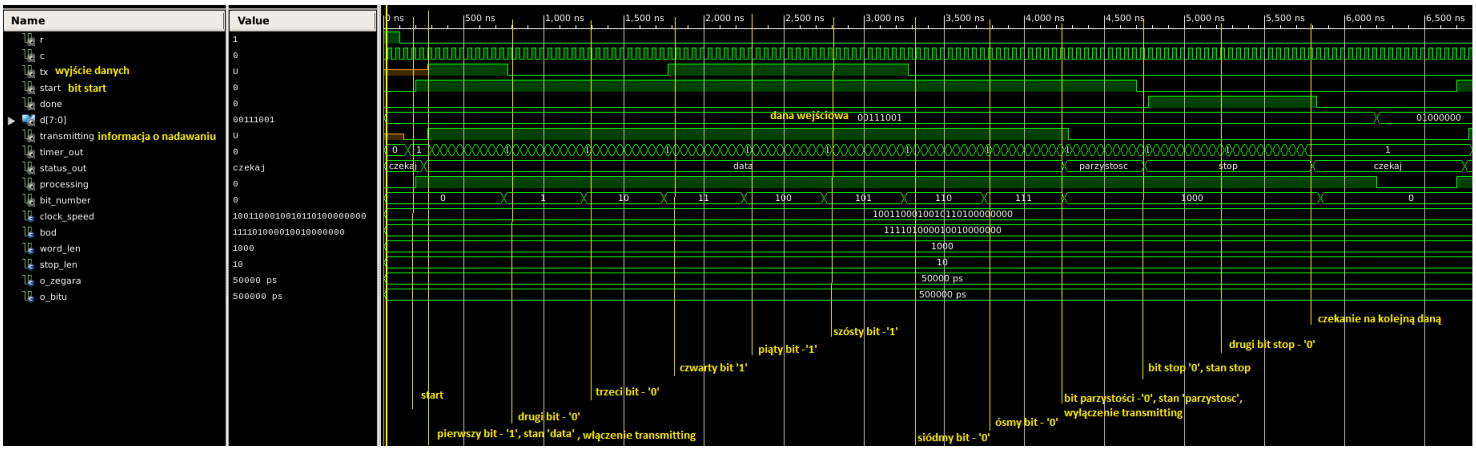
Przygotowaliśmy następujące pliki symulacji:

- 'receiver_tb.vhd' - prezentujący działanie modułu RX,



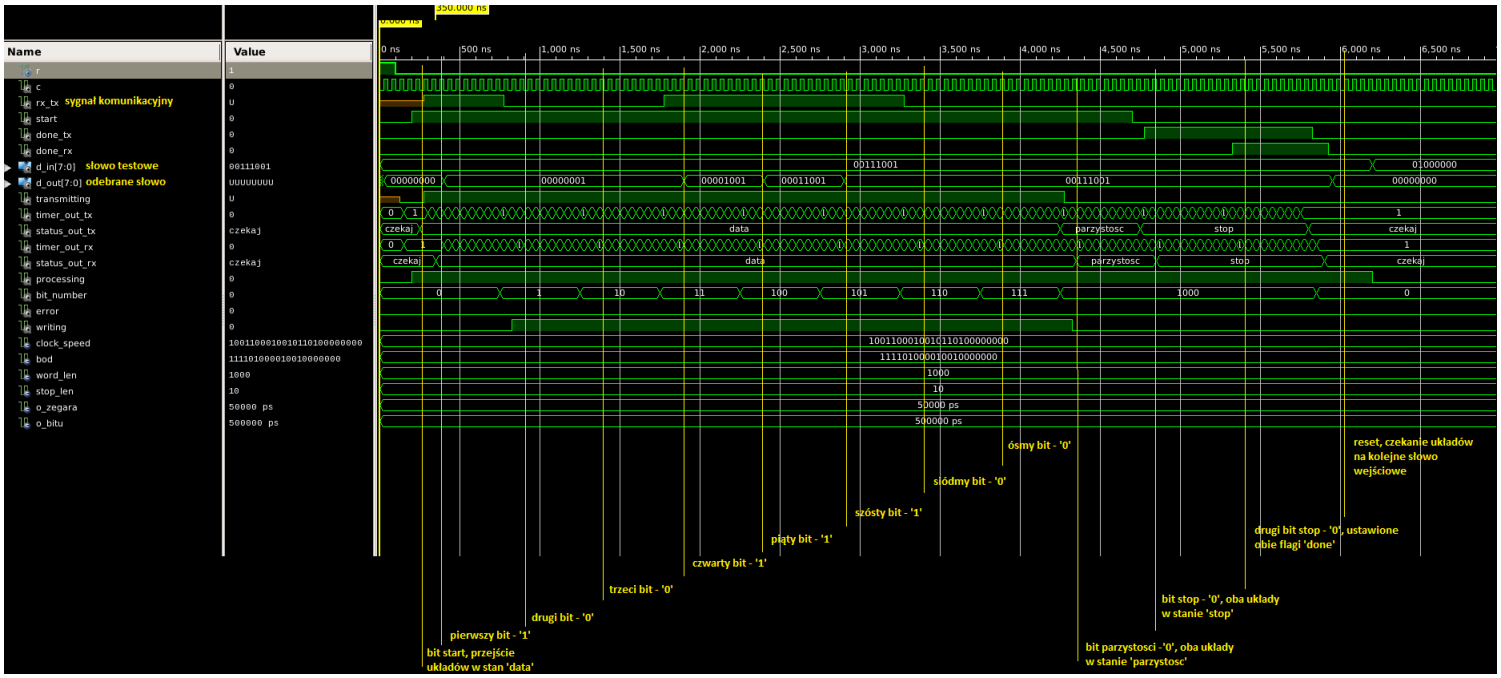
Rysunek 1: Receiver

- 'sender_tb.vhd' - prezentujący działanie modułu TX,



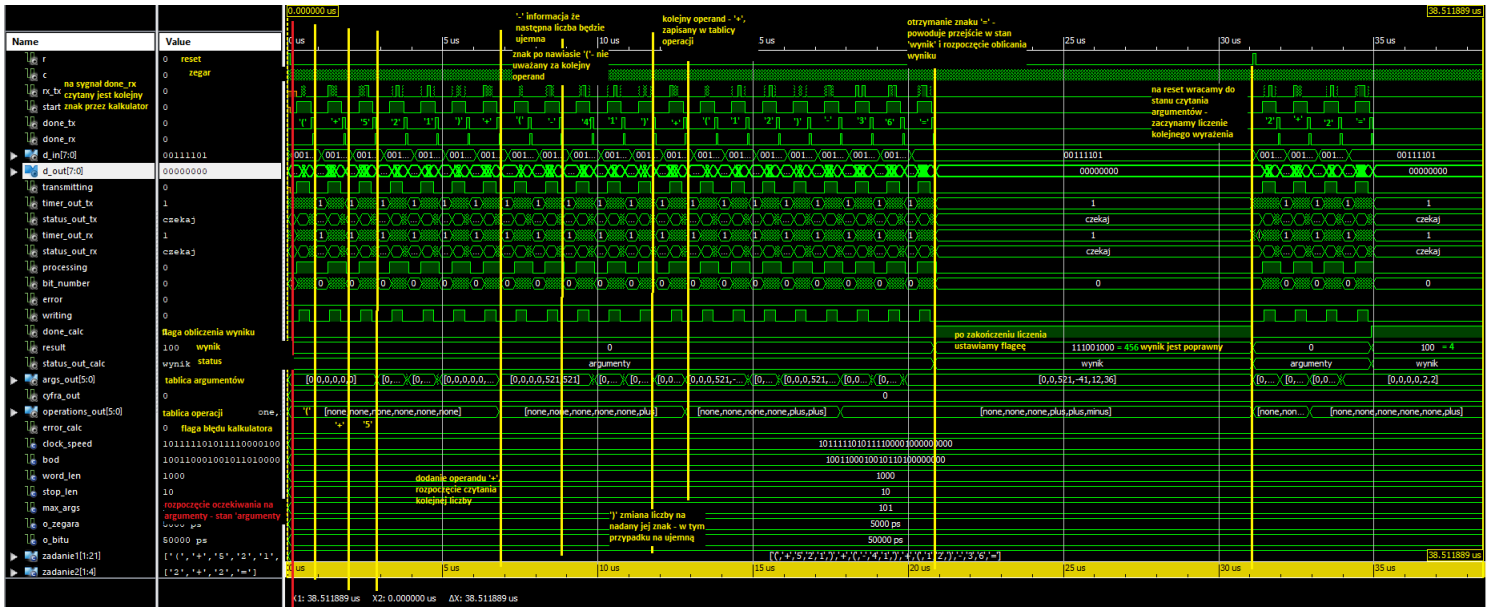
Rysunek 2: Sender

- 'receiver_and_sender.vhd' - prezentujący współpracę modułów RX i TX,

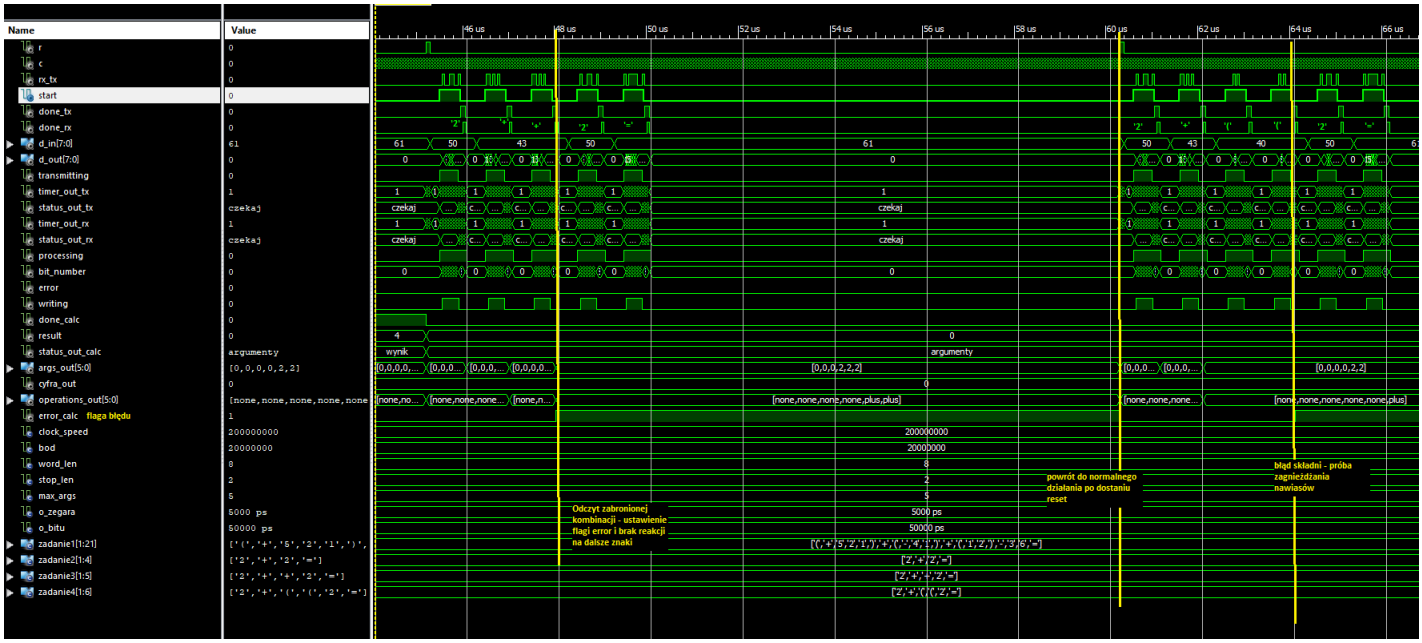


Rysunek 3: Receiver i sender

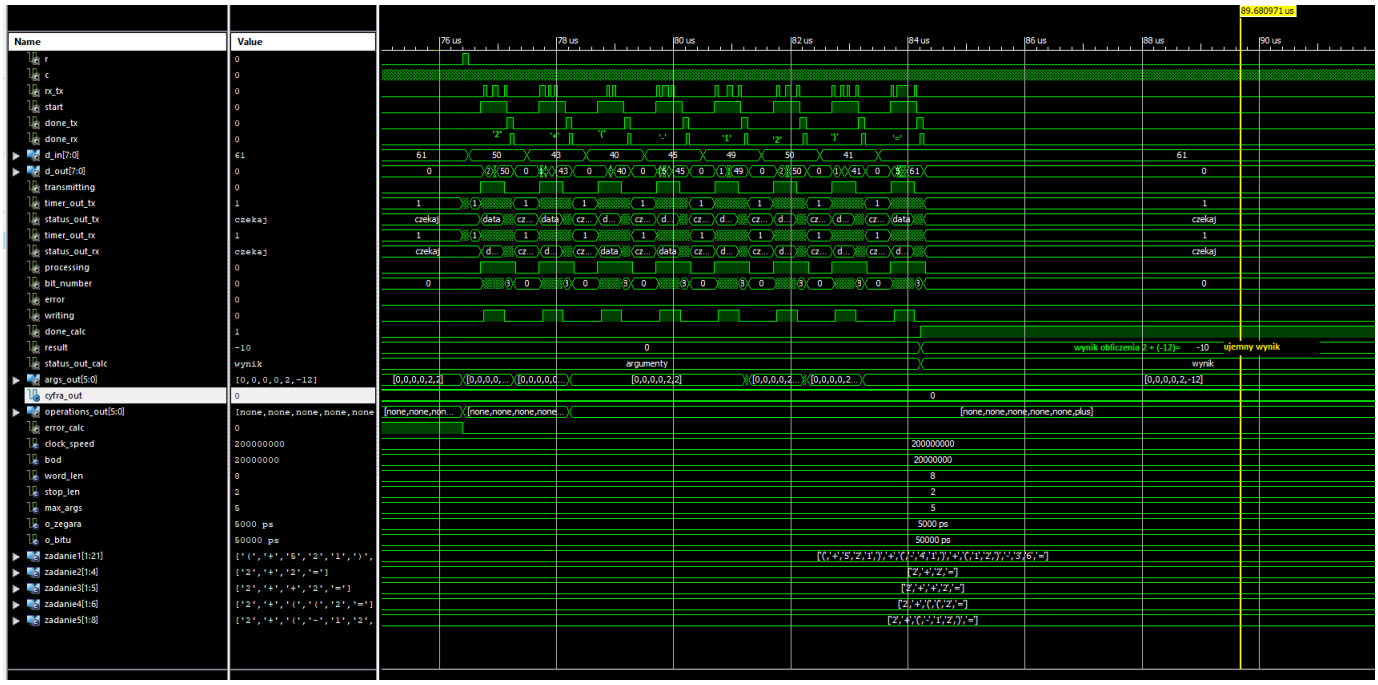
- 'calc_tb.vhd' - prezentujący przekazywanie danych do kalkulatora za pomocą RX i TX oraz obliczenia kalkulatora.



Rysunek 4: Kalkulator

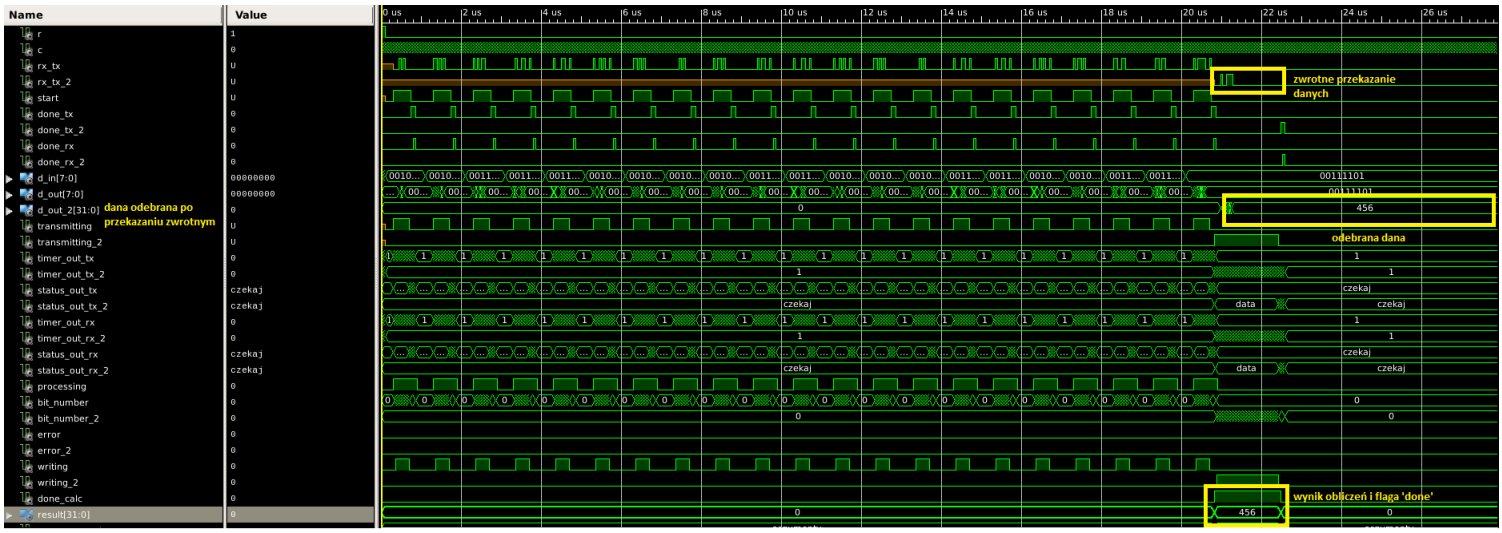


Rysunek 5: Kalkulator - obsługa błędów



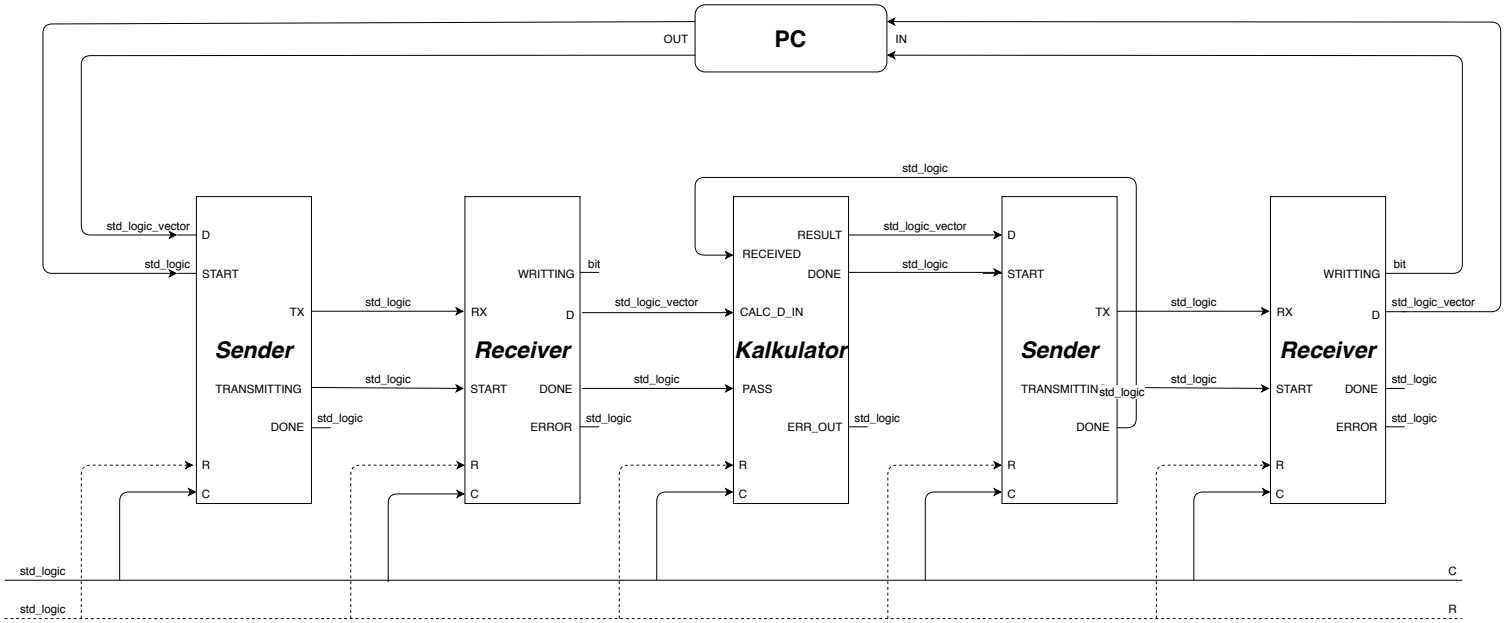
Rysunek 6: Kalkulator - negatywny wynik

- 'calc_tb_2.vhd' - prezentujący końcowy efekt, tj. przekazywanie danych do kalkulatora za pomocą RX i TX, obliczenia kalkulatora oraz przekazanie ich z powrotem przez RX i TX.



Rysunek 7: Kalkulator - końcowy efekt

3.5 Diagram Połączeń Modułów



Rysunek 8: Diagram wejść/wyjść modułów

4 Podział pracy

- Michał Milewski : Sender i obliczenia Kalkulatora
- Piotr Onyszczuk: Receiver i parsowanie wejścia Kalkulatora

5 Podsumowanie

Udało się nam zrealizować projekt zgodny z wymaganiami. Przyjmuje on zadanie wejściowe i zwraca wynik przeprowadzonych obliczeń. Działanie naszego układu zostało przetestowane na kilku przykładach. Wszytskie przykłady zobrazowane zostały na wyżej pokazanych wycinkach symulacji. Wyniki wszytskich przykładów były zgodne z oczekwianiami.

5.1 Przykłady

- $(+521) + (-41) + (12) - 36 =$, wynik oczekiwany: 456, wynik otrzymany: 456,
- $2 + 2 =$ wynik oczekiwany: 4 wynik otrzymany: 4,
- $2 + +2 =$ wynik oczekiwany: błąd, wynik otrzymany: błąd,

- $2 + ((2 =$ wynik oczekiwany: błąd, wynik otrzymany: błąd,
- $2 + (-12) =$ wynik oczekiwany: -10 wynik otrzymany: -10.