Weryfikacja modułów ALU i ACU mikroprocesora osoby z roku

Wymagania

ALU.sv

1. **Operacje arytmetyczne**:
    - ALU musi wykonywać operację dodawania (OP_CODE == 000) na dwóch operandach (left_operand i right_operand) z uwzględnieniem przeniesienia wejściowego (carry_in).
    - ALU musi wykonywać operację odejmowania (OP_CODE == 001) na dwóch operandach (left_operand i right_operand) z uwzględnieniem przeniesienia wejściowego (carry_in).
2. **Operacje logiczne**:
    - ALU musi wykonywać operację bitowego AND (OP_CODE == 002) na dwóch operandach (left_operand i right_operand).
    - ALU musi wykonywać operację bitowego OR (OP_CODE == 003) na dwóch operandach (left_operand i right_operand).
    - ALU musi wykonywać operację bitowego XOR (OP_CODE == 004) na dwóch operandach (left_operand i right_operand).
    - ALU musi wykonywać operację bitowego NOT (OP_CODE == 005) na jednym operandzie (left_operand).
3. **Operacje ładowania i zapisywania**:
    - ALU musi wykonywać operację ładowania (OP_CODE == 006), gdzie wynikiem jest right_operand.
    - ALU musi wykonywać operację zapisywania (OP_CODE == 007), gdzie wynikiem jest left_operand.
4. **Obsługa przeniesienia**:
    - ALU musi generować przeniesienie wyjściowe (carry_out) dla operacji dodawania (OP_ADD) i odejmowania (OP_SUB).

- Dla pozostałych operacji
  (OP_AND, OP_OR, OP_XOR, OP_NOT, OP_LD, OP_ST) przeniesienie wyjściowe (carry_out) musi być równe 0.

5. **Obsługa sygnału CE (Chip Enable)**:
   - Gdy CE = 1, ALU musi wykonywać operacje zgodnie z podanym kodem operacji (OP_CODE).
   - Gdy CE = 0, ALU musi zostawić wyjście op_out, a carry_out musi być równe 0.
6. **Obsługa nieprawidłowych kodów operacji**:
   - ALU musi ustawić wyjście op_out w stan wysokiej impedancji ('z), gdy podany kod operacji (OP_CODE) jest nieprawidłowy (np. 3'bXXX).

ACU.sv

1. **Resetowanie rejestru**:
   - Gdy sygnał resetowania (rstn) jest nieaktywny (niski), rejestr wewnętrzny (int_val_r) musi zostać wyzerowany
2. **Zapisywanie wartości**:
   - Gdy sygnał CE (Control Enable) jest aktywny (wysoki) oraz sygnał zegara (clk) osiąga zbocze narastające, wartość wejściowa (in_val) musi zostać zapisana do rejestru wewnętrznego (int_val_r).
3. **Przekazywanie wartości na wyjście**:
   - Wartość przechowywana w rejestrze wewnętrznym (int_val_r) musi być stale dostępna na wyjściu (out_val).
4. **Obsługa sygnału CE**:
   - Gdy sygnał CE jest nieaktywny (niski), rejestr wewnętrzny (int_val_r) nie powinien być aktualizowany, a jego wartość powinna pozostać niezmieniona.

Plan testów

Test 1

REQ: ALU.sv 1 – Operacje arytmetyczne

Testplan:

1. Wyzwól CE
2. Wprowadź wartości dla left_operand i right_operand, oraz OP_CODE == 000 (dla dodawania) lub 001 (dla odejmowania), a także wartość na wejście carry_in
3. Zweryfikuj poprawność operacji zgodnie z wymaganiami

Test 2

REQ: ALU.sv 2 – Operacje logiczne

Testplan:

1. Wyzwól CE
2. Wprowadź wartości dla left_operand i right_operand, oraz odpowiedni OP_CODE (002-005)
3. Zweryfikuj poprawność operacji zgodnie z wymaganiami

Test 3

REQ: ALU.sv 3 – Operacje ładowania i zapisu

Testplan:

1. Wyzwól CE
2. Wprowadź wartości dla left_operand i right_operand, oraz odpowiedni OP_CODE (006 dla ładowania, 007 dla zapisu)
3. Zweryfikuj poprawność operacji zgodnie z wymaganiami

Test 4

REQ: ALU.sv 4 – Obsługa przeniesienia

Testplan:

1. Wyzwól CE
2. Wprowadź wartości dla left_operand i right_operand oraz carry_in tak aby wygenerować przeniesienie (np. left_operand = 0000_0010, right_operand = 1000_000, carry_in = 1) , oraz odpowiedni OP_CODE (000 dla dodawania, 001 dla odejmowania)
3. Zweryfikuj poprawność operacji zgodnie z wymaganiami

Test 5

REQ: ALU.sv 5 – Obsługa sygnału CE

Testplan:

1. Wyzwól CE
2. Wprowadź wartość na wejście (in_val)
3. Wprowadź wartości dla left_operand, right_operand, carry_in oraz OP_CODE

Test 6

REQ: ALU.sv 6 – Obsługa nieprawidłowych kodów operacji

Testplan:

1. Wyzwól CE
2. Wprowadź wartości dla left_operand, right_operand, carry_in, oraz OP_CODE spoza listy
3. Zweryfikuj stan wyjścia czy jest zgodny z wymaganiami

Test 7

REQ: ACU.sv 1 – Resetowanie rejestru

Testplan:

1. Wyzwól CE oraz rstn
2. Wprowadź wartość na wejście (in_val)
3. Odczekaj takt
4. Zwolnij rstn
5. Sprawdź, czy wyjście zostało wyzerowane

Test 8

REQ: ACU.sv 2 – Zapisywanie wartości

Testplan:

1. Wyzwól CE oraz rstn
2. Wprowadź wartość na wejście (in_val)
3. Odczekaj takt
4. Sprawdź, czy wartość na wyjściu jest zgodna z wymaganiami

Test 9

REQ: ACU.sv 3 – Przekazywanie wartości na wyjście

Testplan:

1. Wyzwól CE oraz rstn
2. Wprowadź wartość na wejście (in_val)
3. Odczekaj takt
4. Sprawdź, czy wartość na wyjściu jest zgodna z wymaganiami

Test 10

REQ: ACU.sv 4 – Obsługa sygnału CE

Testplan:

1. Wyzwól CE oraz rstn
2. Wprowadź wartość na wejście (in_val)
3. Odczekaj takt
4. Zwolnij CE,
5. Odczekaj takt
6. Sprawdź, czy wartość na wyjściu jest zgodna z wymaganiami

Matryca pokrycia wymagań

Wymaganie	Moduł	Test
REQ1	ALU.sv	1, 4, 5
REQ2	ALU.sv	2, 4, 5
REQ3	ALU.sv	3, 4, 5
REQ4	ALU.sv	4, 1, 5
REQ5	ALU.sv	5
REQ6	ALU.sv	6
REQ1	ACU.sv	7
REQ2	ACU.sv	8, 9
REQ3	ACU.sv	8, 9
REQ4	ACU.sv	10, 8, 9

Layered TB

Oba moduły zostały przetestowane za pomocą warstwowego testbencha. Wyjście ACU zostało podłączone do wejścia ALU left_operand. Wyjście op_out zostało podłączone do wejścia in_val. Sygnał CE został podłączony do wejść CE obu modułów. Na wejścia interfejsu zostały podane dane losowe, które zawierają przypadki z testplanu.