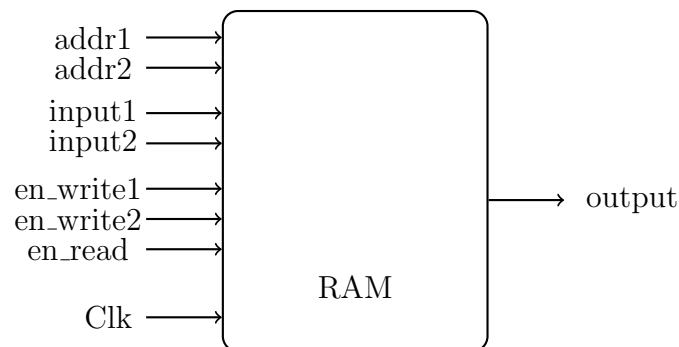


RAM (Datenspeicher)

Ihre Aufgabe ist es, das Verhalten einer Entity namens „RAM“ zu programmieren. Die Entity ist in der angehängten Datei „RAM.vhdl“ deklariert und hat folgende Eigenschaften:

- Eingang: Clk vom Typ std_logic
- Eingang: en_read vom Typ std_logic
- Eingang: en_write1 vom Typ std_logic
- Eingang: en_write2 vom Typ std_logic
- Eingang: input1 vom Typ std_logic_vector
- Eingang: input2 vom Typ std_logic_vector
- Eingang: addr1 vom Typ std_logic_vector
- Eingang: addr2 vom Typ std_logic_vector
- Ausgang: output vom Typ std_logic_vector



Verändern sie die Datei „RAM.vhdl“ nicht!

Die Entity „RAM“ soll sich wie folgt verhalten:

Die Ausgänge der Entity und der Inhalt des RAMs können sich nur bei einer steigenden Flanke des Taktsignals verändern. Der anfängliche Inhalt des Speichers ist Null.

Die Adresse einer Speicherposition, an die Daten geschrieben oder von der Daten ausgelesen werden, wird durch die Adresseingänge festgelegt. Wenn die Steuereingänge für einen Lesebefehl („read enable“) oder einen Schreibbefehl („write enable“) aktiv sind ('1'), dann soll der jeweilige Befehl bei steigender Flanke des Taktsignals ausgeführt werden.

- Die erste Adresse ist die Adresse für den ersten Schreibbefehl (aktiviert durch `en_write1`) und den Lesebefehl (aktiviert durch `en_read`). Die zweite Adresse ist die Adresse für den zweiten Schreibbefehl (aktiviert durch `en_write2`) und hat keine zusätzliche Funktion.
- Wenn nur `en_read '1'` ist, dann wird der Inhalt von `addr1` gelesen und am Ausgang ausgegeben.
- Wenn nur `en_write1 '1'` ist, dann wird der Eingang `input1` an die Adresse `addr1` geschrieben.
- Wenn nur `en_write2 '1'` ist, dann wird der Eingang `input2` an die Adresse `addr2` geschrieben.
- Lesen von `addr1` und Schreiben auf `addr2` kann gleichzeitig ausgeführt werden, wenn `addr1` sich von `addr2` unterscheidet, `addr2` sich von der nächsthöheren Adresse von `addr1` unterscheidet und `addr1` sich von der nächsthöheren Adresse von `addr2` unterscheidet.
- Zwei Schreibbefehle können gleichzeitig ausgeführt werden, wenn sich `addr1` von `addr2` unterscheidet, `addr2` sich von der nächsthöheren Adresse von `addr1` unterscheidet und `addr1` sich von der nächsthöheren Adresse von `addr2` unterscheidet.
- Beachten Sie, dass sich in allen anderen Fällen der Inhalt des RAMs nicht verändern darf und der Ausgang hochohmig (high impedance, 'Z') sein soll.
- Wenn der Speicher nicht ausgelesen wird, soll der zugehörige Ausgang hochohmig (high impedance, 'Z') sein.

Die Länge der Adressen beträgt 10 Bit, die Länge der Eingangsdaten 16 Bit, die Länge der Ausgangsdaten 16 Bit und die Länge der einzelnen Speicherzellen 8 Bit.

- Die Länge der Eingangsdaten ist doppelt so groß wie die Länge der Speicherzellen. Daher soll die untere Hälfte der Eingangsdaten an die angegebene Adresse und die obere Hälfte an die nächsthöhere Adresse geschrieben werden. Wir nehmen dabei an, dass die letztmögliche Speicheradresse nie für einen Schreibbefehl verwendet wird.
- Die Länge der Ausgangsdaten ist doppelt so groß wie die Länge der Speicherzellen. Daher sollen die Daten von der angegebenen Adresse gelesen und in die untere Hälfte der Ausgangsdaten gesetzt werden. In die obere Hälfte der Ausgangsdaten sollen die Daten von der nächsthöheren Adresse gesetzt werden. Wir nehmen dabei an, dass die letztmögliche Speicheradresse nie für einen Lesebefehl verwendet wird.

Programmieren Sie dieses Verhalten in der angehängten Datei „RAM_beh.vhdl“.

Um Ihre Lösung abzugeben, senden Sie ein E-Mail mit dem Betreff „Result Task 5“ und Ihrer Datei „RAM_beh.vhdl“ an vhdl-mc+e384@tuwien.ac.at.

Viel Erfolg und möge die Macht mit Ihnen sein.