Technische Universität München
Fakultät für Informatik
Rechnerarchitektur-Praktikum
SS 2015

SPEICHERTECHNOLOGIE – DRAM-INTERFACE

SPEZIFIKATION

Bearbeitet von:

Mahdi Sellami
Niklas Rosenstein
Christoph Pflüger

07.06.2015

INHALT

1. Aufgabenkurzbeschreibung

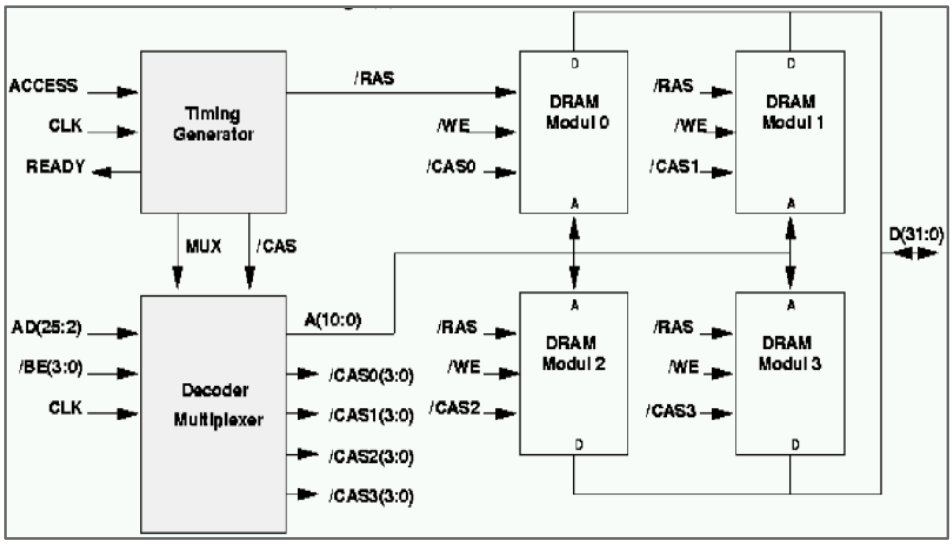

Abb. 1: Gegebenes Modul inkl. Ein- und Ausgänge

Gegeben ist ein DRAM-Speicherbaustein (siehe Abb. 1) bestehend aus den folgenden Teilbausteinen:

- Timing Generator
- 4x DRAM-Modul
- Decoder Multiplexer

Im Rahmen dieses Projekts soll eine Simulation des Decoder Multiplexers mithilfe der Hardwarebeschreibungssprache VHDL entwickelt werden. Dabei muss eine korrekte und zulässige Abfolge der Signale an den Eingängen /RAS, /MUX und /CAS durch den Timing Generator sichergestellt werden.
Dem Decoder Multiplexer stehen die Eingänge AD(25:2), /BE(3:0) und CLK zur Verfügung. Er hat die Aufgabe, das in AD(25:2) anliegende Signal in Zeilen- und Spaltenadresse aufzuteilen und an den A(10:0) Ausgang zu übermitteln. Weiter soll dieser Baustein mithilfe von /BE(3:0), /CAS und MUX das entsprechende DRAM-Modul durch /CASx(3:0) ansprechen.

2. Lösungsansätze

Die Lösung für das Auswählen des angesprochenen DRAM Moduls kann bis auf ein paar stilistische Änderungen (z.B. Switch statt if/else) nur auf einem Weg umgesetzt werden. Für das Aufsplitten der Spalten- und Zeilenadressen hingegen wurden zwei Lösungsansätze ausgearbeitet. **Lösungsansatz A** verwendet die ledigliche Zuweisung der höher- und niederwertigen Bits aus A(25:2). **Lösungsansatz B** hingegen verwendet Components, um eine Design Hierarchie zu erstellen.

2.1. Lösungsansatz A

Der Lösungsansatz ergibt sich aus der nachfolgenden Beschreibung des Decoder Multiplexers in VHDL:

Entity

Name	Modus	Type
MUX	IN	std_logic
/CAS	IN	std_logic
CLK	IN	std_logic
AD	IN	std_logic_vector (25 downto 2)
/BE	IN	std_logic_vector (3 downto 0)
A	OUT	std_logic_vector (10 downto 0)
/CAS0	OUT	std_logic_vector (3 downto 0)
/CAS1	OUT	std_logic_vector (3 downto 0)
/CAS2	OUT	std_logic_vector (3 downto 0)
/CAS3	OUT	std_logic_vector (3 downto 0)

Process

Wenn falling_edge(/RAS)
    Initialisiere A(10:0) **mit** 11 höherwertigen Bits **von** AD(25:2)
Wenn falling_edge(/CAS)
    Wenn falling_edge(/BE)
        Initialisiere /CASx(3:0) **mithilfe von** /A(25:2) **mit** /BE(3:0)
    Ansonten
        Initialisiere A(10:0) **mit** 11 niederwertigen Bits **von** AD(25:2)

2.2. Lösungsansatz B

Im Unterschied zu **Lösungsansatz A**, in welchem die höher- und niederwertigen Bits aus A(25:2) lediglich dem Ausgang A(10:0) zugewiesen werden, schlägt Lösungsansatz B das Gestalten dieses Prozesses in einer Component vor. Die Lösung zum Ansprechen des benötigten DRAM Moduls entspricht der Lösung aus **Lösungsansatz A.**

3. Bewertung der Lösungsansätze

Wobei **Lösungsansatz A** eine sehr einfache Zuweisung der benötigten Daten darstellt, stellt **Lösungsansatz B** eine technisch hochwertigere Struktur dar. Durch den etwas höheren Aufwand, der mit diesem Lösungsansatz verbunden ist, wird bessere Wartbarkeit garantiert und nebenbei wird eine Design Hierarchie gebildet, die die Struktur des VHDL Programms fordert.

4. Entscheidungswahl

Im Rahmen dieses Projektes haben wir uns entschieden in den ersten Schritten aufgrund der Einfachheit und basierend auf unserem Wissensstand im Umgang mit der Hardwarebeschreibungssprache VHDL **Lösungsansatz A** zu verfolgen.