Datenblatt

Kommunikationsmatrix

Nick Meier

2. Juni 2008

Inhaltsverzeichnis

1 Communication Matrix Details

Die Nutzung der dynamischen partiellen Rekonfiguration (DPR) für Hardwareprojekte bedingt einen speziellen Aufbau der Anwendungskomponenten und der Kommunikation zwischen diesen. Daher wurde bisher für jede neue Anwendung auch die Aufteilung der Chipressourcen und der Kommunikationsaufbau neu und vor allem individuell erstellt. Damit können die einzelnen Anwendungen aber nicht beliebig ausgetauscht werden und der Entwicklungsaufwand ist entsprechen hoch. Mit dem neu entwickelten Programmierparadigma POL (Parallel Object Language) sollen aber beliebige Anwendungen in einer Hochsprache spezifiziert werden und anschließend möglichst automatisch in funktionierende Hardware übersetzt werden. Dazu ist aber eine allgemeingültige Struktur für die Verteilung der Chipressourcen und für die Kommunikation nötig. Eine Realisierung dieser Struktur für POL ist im folgenden beschrieben.

1.1 Introduction

Für die maximale Größe der Anwendungen gibt es im Prinzip nur einen freien Parameter, nämlich die verfügbaren Ressourcen des verwendeten FPGAs. Diese bestimmen sowohl die Anzahl der dynamisch rekonfigurierbaren Bereiche des Chips als auch die Größe und Anzahl der Puffermodule. Die Anzahl der realisierbaren Objekte ist hauptsächlich durch die in der Kommunikationsmatrix reservierten Adressbereiche beschränkt. Die Tiefe der Puffermodule ist ebenfalls von dem auf dem Chip verfügbaren Speicher (Bram-Blöcke) abhängig.

Diese erste Spezifikation setzt die Breite der Datenwörter und den Adressbereich auf 16 bit fest. In diesen 16 bit Adressen ist Raum für 16 verschiedene Klassen mit jeweils 64 Instanzen. Außerdem wird der Eingang der einzelnen Klassen auf maximal 16 beschränkt.

Tabelle 1: Communication Word

data	id			
	class_id	instance_id	register_id	reserved
[31:16]	[15:12]	[11:6]	[5:2]	[1:0]
xxxxxxxxxxxxxxxx	xxxx	xxxxxx	xxxx	00

Die eigentliche Kommunikationsmatrix setzt sich aus den Klassenpuffern, den Taskareas und einer Kontrolleinheit zusammen. Die Puffer und die Areas bestehen allgemein aus einer Ausgangsfifo und einem Eingangsmultiplexer. Verbunden sind jeweils jeder Ausgang einer Komponente mit allen Eingängen

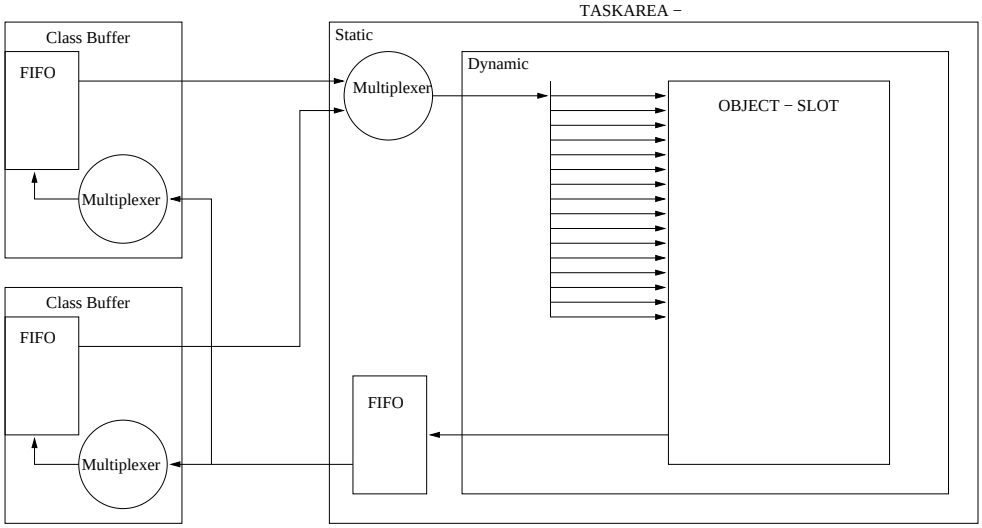
der jeweils anderen. Also ist z. B. die Ausgangsfifo einer Taskarea mit jedem Eingangsmultiplexer der Klassenpuffer verbunden und umgekehrt.

In den Klassenpuffern werden die Daten der einzelnen Klassen unabhängig von den Instanzen gespeichert. Die Multiplexer beobachten die Ausgänge und sammeln die entsprechend adressierten Nachrichten ein.

Die Kontrolleinheit ist für Systemnachrichten an die Klassenobjekte und das dynamische Instantieren von Klassen zuständig. Dieser Dynamic Controll Bus (DCB) ist in beiden Richtungen jeweils 16 bit breit.

Die Taskarea enthält außer dem Multiplexer und dem Ausgangspuffer die für die Rekonfiguration wichtigsten Komponenten. Hier findet das Rekonfiguration Domain Crossing statt. Das bedeutet, in der Taskarea findet der Übergang von den statischen zu den dynamischen Komponenten über spezielle Busmacros statt. Erst nach diesem Übergang werden die Nachrichten auf die einzelnen Eingänge der Objekte übersetzt. Dies entspricht der VHDL-Komponente der Taskarea. Das eigentliche Objekt ist dieser dann untergeordnet.

Abbildung 1: Communication Matrix

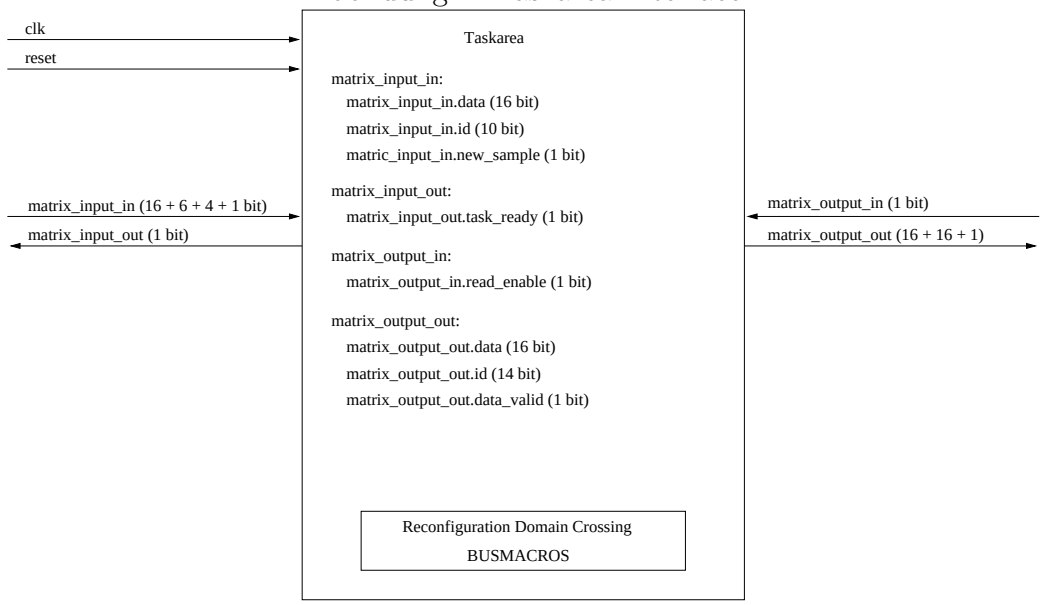


Die genaue Zusammensetzung der Komponenten und der Interfaces wird in den einzelnen Kapiteln beschrieben.

1.2 Taskarea Interface

Wie oben beschrieben, ist die Taskarea für die Funktionalität der Rekonfiguration entscheidend. Natürlich muss auch das Interface zu den Klassenpuffern und dem Controllbus passen.

Abbildung 2: Taskarea Interface



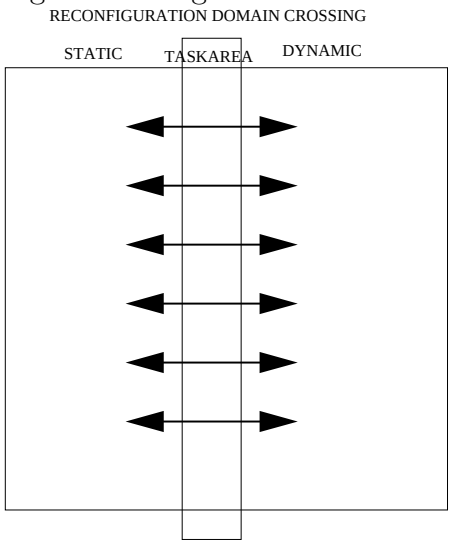
Die eigentliche Aufgabe der Taskarea ist aber das Reconfiguration Domain Crossing (RDC). An dieser Stelle werden die Komponenten aus dem System herausgeschnitten und sind durch andere Module austauschbar. Damit dies funktioniert, müssen alle Drähte immer an der selben Stelle aufeinandertreffen. Dies sollte logisch natürlich vor der Taskarea geschehen, aber technisch ist das RDC vor der Aufteilung der Daten auf die Eingänge der Klassen nötig, da sonst der gesamte Chip mit Interfacelogic (Busmacros) überlaufen ist. Daher ist eine deutliche Unterscheidung zwischen der logischen Taskarea (mit Multiplexer und Fifo) und der VHDL-Komponente (ohne Multiplexer und Fifo) zu treffen.

1.3 Class Buffer Interface

Der Klassenpuffer ist eindeutig einer Klasse zugeordnet und enthält alle Datenpakete der Instanzen dieser Klasse. Außerdem übernimmt er das Abholen der Daten aus dem Ausgangspuffer der verschiedenen Taskareas. Dazu enthält er einen Multiplexer, der die Taskareas auf Daten für die zugeordnete Klasse überprüft, und eine eigene Fifo. Mit dieser Fifo werden Rekonfigurationszeiten sowohl bei der langen Konfiguration (Klassenwechsel) als auch der kurzen Konfiguration (Instanz- bzw. Kontextwechsel) abgefangen.

Der Klassenpuffer erhält auch den Systemtakt und den globalen Reset. Die Anzahl der weiteren Eingänge hängt von der Anzahl der Taskareas ab.

Abbildung 3: Reconfiguration Domain Crossing



Ein einzelner Eingang besteht aus einem 33 bit breiten Eingangspaket und einer read_enable Ausgangsmarkierung, um die Daten aus dem Ausgangspuffer der entsprechenden Taskarea zu erhalten. Der Ausgang besteht ebenfalls aus einem 33 bit Ausgangspaket und einer write_enable Eingangsmarkierung, die die Bereitschaft der Instanz signalisiert.

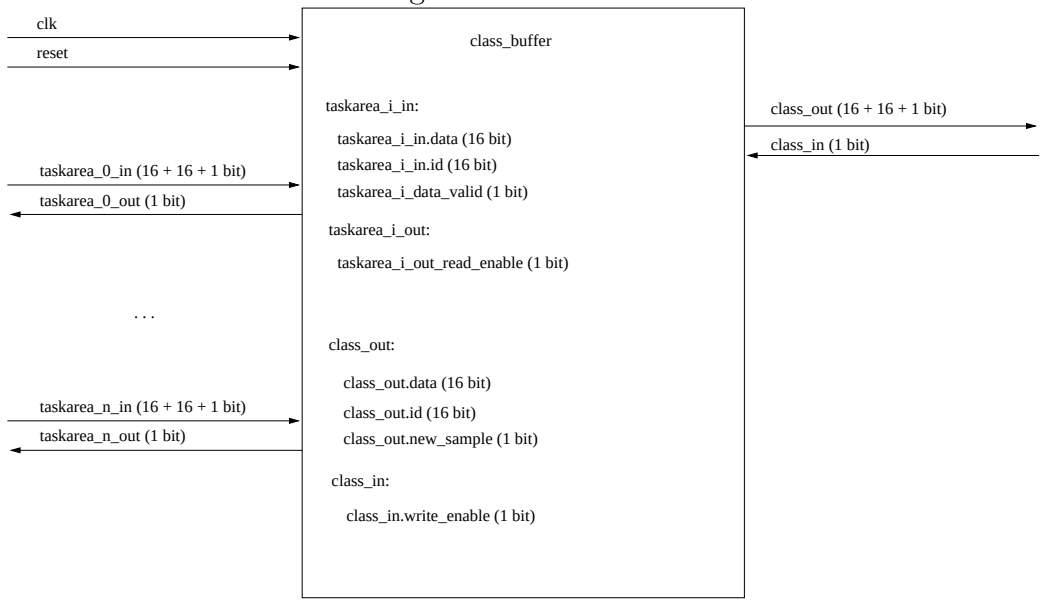
Tabelle 2: Data Package

data	id	data_valid
[32:17]	[16:1]	[0]
xxxxxxxxxxxxxxxx	xxxxxxxxxxxxxxxx	1

1.4 Object Interface

Ein Objekt erhält allgemein den Systemtakt und einen globalen Reset. Als Eingang stehen einem Objekt 16 Eingangskanäle zur Verfügung. Diese bestehen aus einem 16 bit breiten Datenwort und einer new_data Eingangsmarkierung sowie einer data_acknowledge Ausgangsmarkierung. Der Ausgang des Objektes setzt sich aus einem 16 bit Adresswort sowie einem 16 bit Datenwort und einer write_enable Ausgangsmarkierung zusammen. Zusätzlich gibt es einen Befehlsbus (dynamic).

Abbildung 4: Class Buffer Interface



1.5 Dynamic Controll Bus

Der Befehlsbus (DCB), über den sowohl das dynamische Anlegen und Zerstören von Objekten als auch die Steuerbefehle laufen, setzt sich im Ein- und Ausgang aus einem 5 bit Befehlswort, einem 10 bit Datenwort und einer Bestätigungsmarkierung zusammen.

Tabelle 3: Dynamic Output Word

command	data	write_enable
[15:11]	[10:1]	[0]
xxxxx	xxxxxxxxxx	1

Tabelle 4: Dynamic Input Word

command	data	write_enable
[15:11]	[10:1]	[0]
xxxxx	xxxxxxxxxx	1

Auf diesem Bus können System und Objekt direkt miteinander kommunizieren. Die bisherigen Befehle sind in den Tabellen 5 und 6 beschrieben.

Abbildung 5: Object Interface

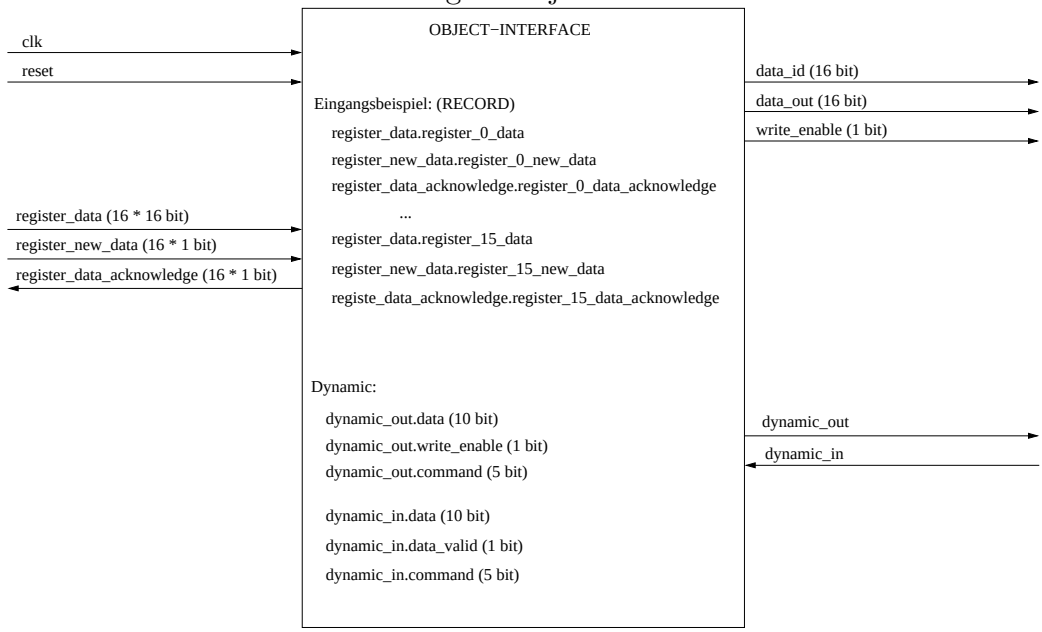


Tabelle 5: Dynamic Output Command Code

Command	Code	Description
DELETE	00000	Indicates the destruction of an class instance.
NEW	00001	Indicates the instantiation of a class.
RECONF	00010	Confirms a reconfiguration.

Tabelle 6: Dynamic Input Command Code

Command	Code	Desciption
HOLD	00001	Initiates the assurance of the instance content for reconfiguration.
INSTID	00010	Write Instance ID: return of a reference after generating a new class instance.