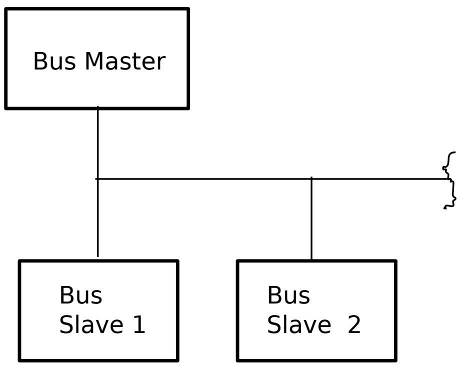
# Specification

*simple pipelined bus* IP-Core (VHDL)

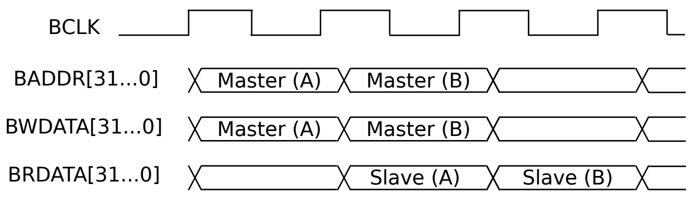
Dossmatik GmbH
Karlstraße 41
04420 Markranstädt
info@dossmatik.de

http://www.dossmatik.de

## block diagram



## timing diagram



## contents

## Features

- all signals have positive logic
- similar handshaking
- easy to understand
- no overhead
- high-bandwidth
- data wide 32bit
- bidirectional /data duplex
- slave can hold bus with waitstages
- synthesizable
- byte mask for active bytes

## Applications

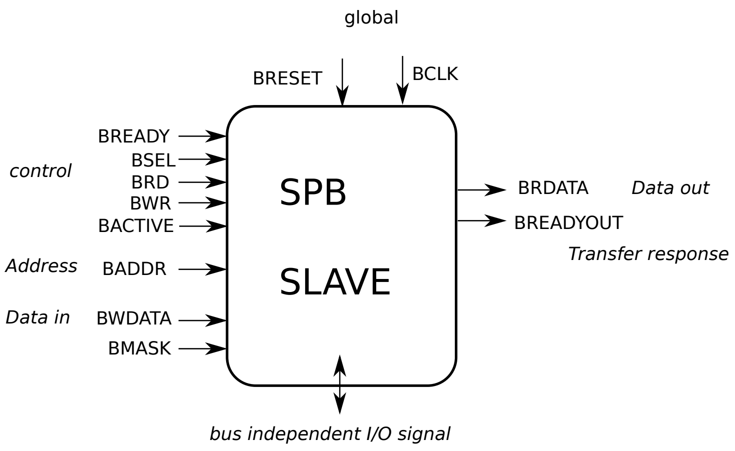
- system on chip
- single bus master
- data flow between moduls

Version: V1.0
Date: 1.10.2013

# 1 Introduction

This specification describes a digital bus system. It is small and very practial for use in hardware design. All signals names and valid of signals are written in this document. The bus is designed for single bus master and high-bandwidth communication with one or more slaves. The master has to decode which slave is active. All bus control signal are generated by the master.

# 2 Slave-Interface



## 2.1 Global Signals

| signal | description |
|---|---|
| **BCLK** | All signals are related to rising edge of **BCLK**. |
| **BRESET** | The **BRESET** is high active. Used for inital setting in slaves. |

## 2.2 Signals to Slave

| signal | description |
|---|---|
| **BADDR [31:0]** | 32-bit address |
| **BSEL [x:0]** | The bus master decodes addtional to the address a slave select signal. Each slave has an own individual bit in the array. The slave must also monitor the status of **HREADY** to ensure that the previous bus transfer has completed, If **HREADY** not high the current select is not valid. |
| **BRD** | Read signal |
| **BWR** | Write signal |
| **BACTIVE** | new Master phase, read or write sequence |
| **BMASK [3:0]** | byte mask, show transfer width and witch byte is tranfered |
| **BWDATA [31:0]** | 32bit data from master to slave |
| **BREADY** | all signals to slave are valid and the previous transfer is finished |

## 2.3 Signals from Slave

| signal | description |
|---|---|
| **BRDATA [31:0]** | 32bit data from slave to master |
| **BREADYOUT** | Slave can indicate with zero level to get additional clock cycle. |

# 3 Transfer

## 3.1 typical transfer

The SPB-Bus use two phases. A master and a slave phase. The transfer is started with a master phase and this phase will change to slave phase if **BREADY** is 1. When the slave phase is on the Bus the data from slave is transfered to master. The bus is duplex. Then a slave phase is on the bus also the next master phase can bealso on the bus. This technique is call pipeline. It is an optimised data troughput.
In the master phase all signals are broadcasted to slave. The resonse from slave is in the slave phase. The slave generates an acknowledge with **BREADYOUT**.

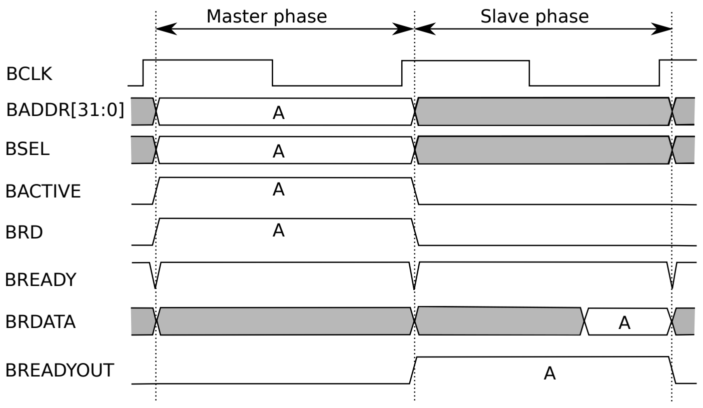


Figure 1: read transfer

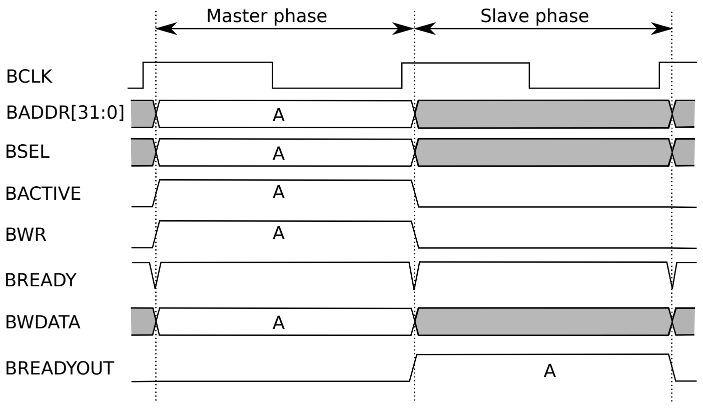


Figure 2: write transfer

Each transfer has valid address and also valid control signals. Burst writes and reads are possible but each transfer is a single transfer.

## 3.2 Transfers with waitstages

Some slaves can not receive too much datas or response in the next BCLK cycle like in section typical transfer. A slave indicates with **BREADYOUT**=0 the slave phase is not finsh. The master holds on bus the next master phase valid until **BREADYOUT**=1. Wait stages are allowed in read and write transfers. But they blocking the bus and can also blocking the master. This technic should not used for a handshake. This addtional clock cycles are for garante the data valid or prevent overflows in data stream.

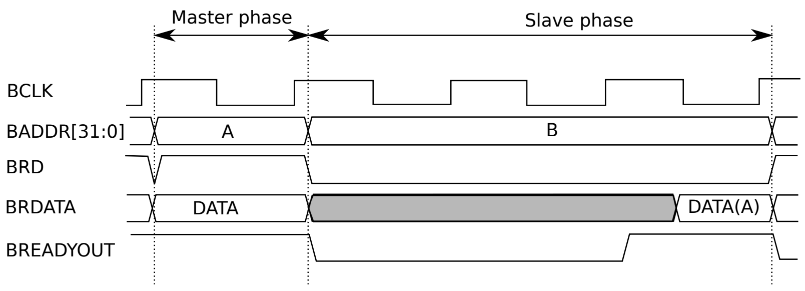


Figure 3: read transfer with waitstages

## 4 VHDL Helpers

```
type SPB_in is
record
  BADDR     : std_logic_vector (31 downto 0);
  BSEL      : std_logic_vector (sel_size-1 downto 0);
  BRD       : std_logic;                --READ
  BWR       : std_logic;                --WRITE
  BACTIVE   : std_logic;
  BMASK     : std_logic_vector (3 downto 0);
  BWDATA    : std_logic_vector (31 downto 0);
  BREADY    : std_logic;
end record;

type SPB_out is
record
  BRDATA    : std_logic_vector (31 downto 0);
  BREADYOUT : std_logic;
end record;
```

## 5 licence