# Chips-2.0 Demo for SP605 Development Card


**Author:** Jonathan P Dawson
**Date:** 2013-10-17
**email:** chips@jondawson.org.uk


This project implements a TCP/IP stack. The TCP/IP stack acts as a server, and can accept a single connection to a TCP port. The connection is provided as a bidirectional stream of data to the application. The following protocols are supported:

- ARP request/response (with 16 level cache)
- ICMP echo request/response (ping)
- TCP/IP socket

# Dependencies

The stack is implemented in C, and needs Chips-2.0 to compile it into a Verilog module.

# Source Files

The TCP/IP stack is provided by two source files:

- source/server.h
- source/server.c

# Configuration

The following parameters can be configured at compile time within source/server.h:

- Local Ethernet MAC address
- Local IP Address
- Local TCP Port number

# Compile

Compile into a Verilog module (server.v) using the following command:

```
$ chip2/c2verilog source/server.v
```

# Interface

## Ethernet Interface

The ethernet interface consists of two streams of data:

- An input, input_eth_rx.
- An output, output_eth_tx.

Both streams are 16 bits wide, and use the following protocol:

word, designation 0, length in bytes n, data

## Socket Interface

The socket interface consists of two streams of data:

- An input, input_socket.
- An output, output_socket.

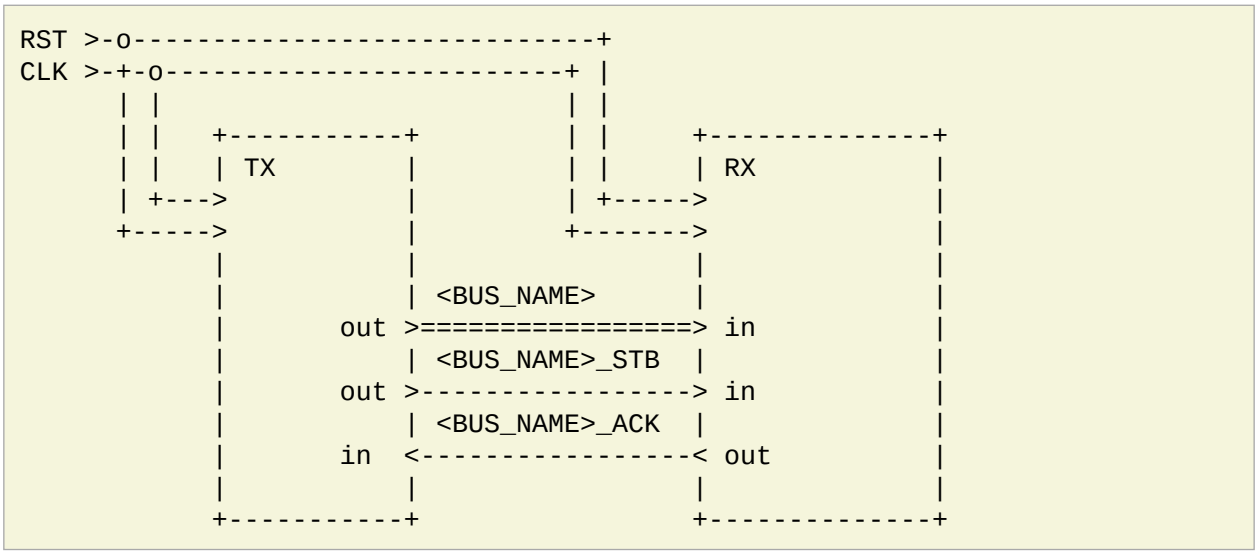
Both streams are 16 bits wide, and use the following protocol:

word, designation 0, length in bytes n, data

# Stream Interconnect Conventions

The main aims of the interface are:

- To be simple to implement.
- Add little performance/logic overhead.
- Allow designs to grow without adding extra levels of asynchronous logic.
- Easy to interface with standard interconnects.




## Global Signals

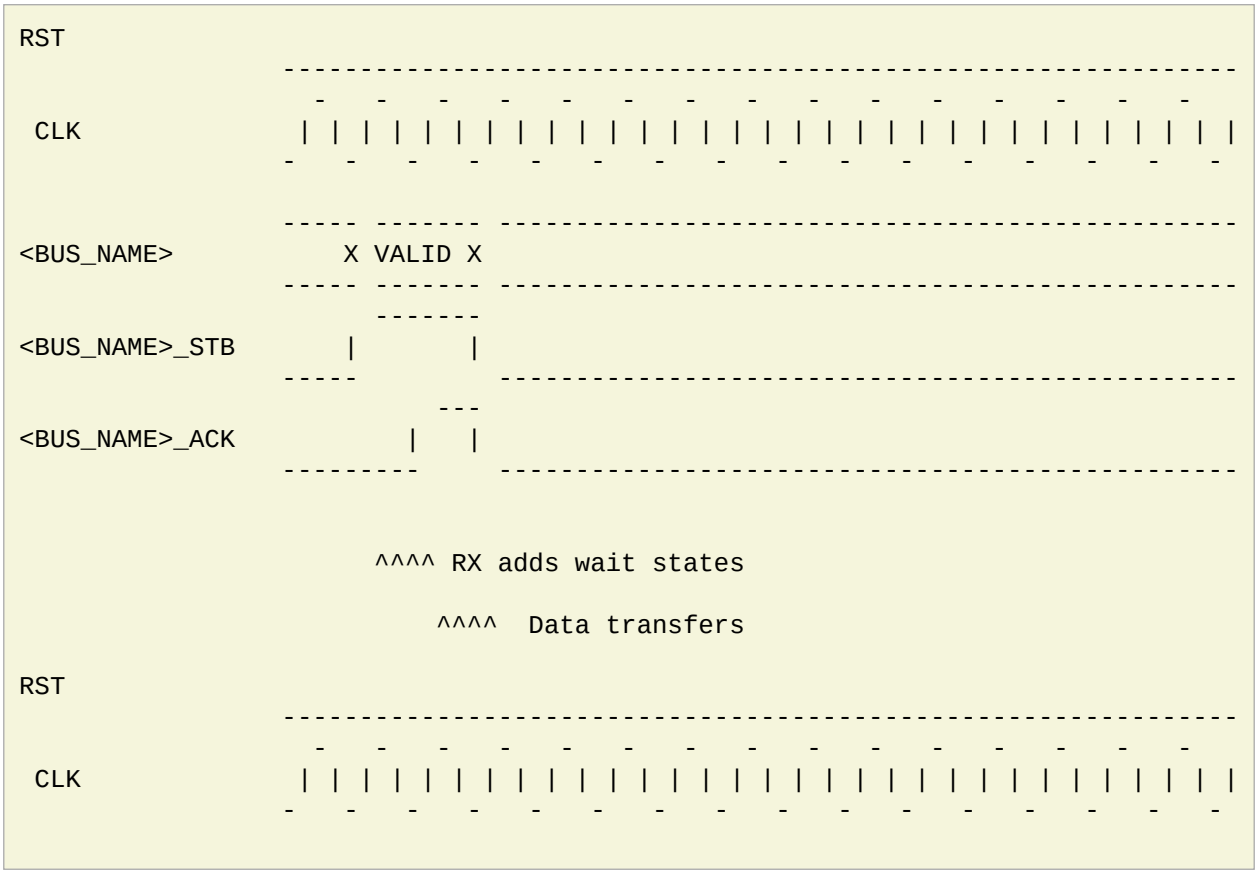
| Name | Direction | Type | Description |
|---|---|---|---|
| CLK | input | bit | Clock |
| RST | input | bit | Reset |

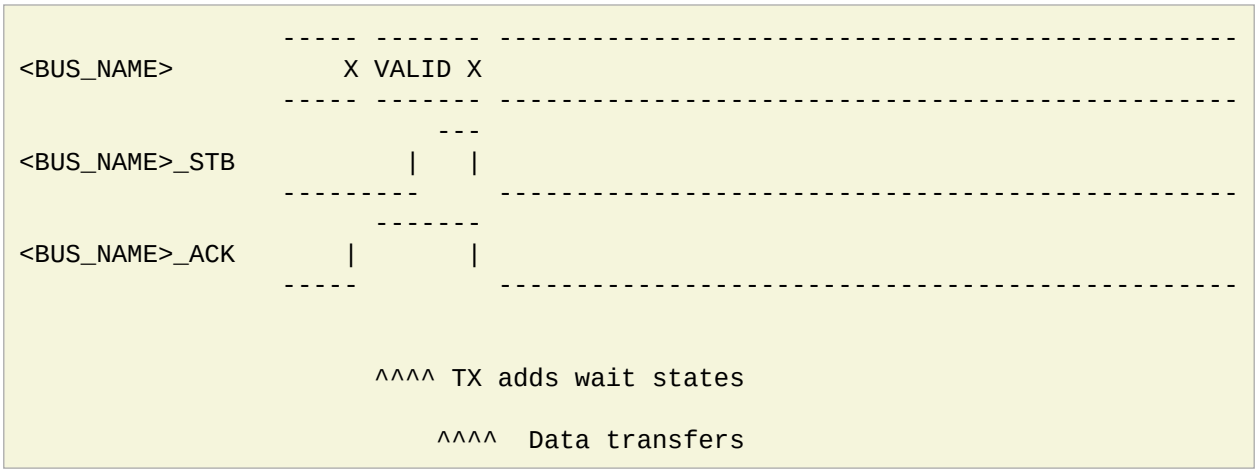
## Interconnect Signals

| Name | Direction | Type | Description |
|---|---|---|---|
| <BUS_NAME> | TX to RX | bus | Payload Data |
| <BUS_NAME>_STB | TX to RX | bit | '1' indicates that payload data is valid and TX is ready. |
| <BUS_NAME>_ACK | RX to TX | bit | '1' indicates that RX is ready. |

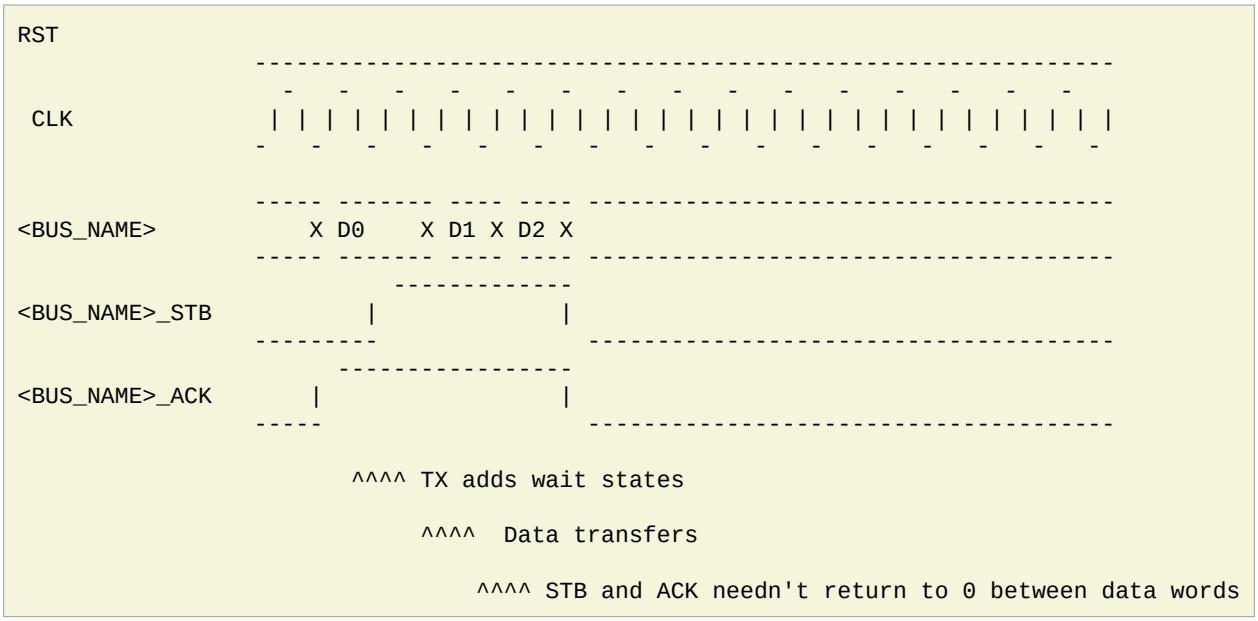
# Interconnect Bus Transaction

- Both transmitter and receiver shall be synchronised to the '0' -> '1' transition of CLK.
- If RST is set to '1' upon the '0' -> '1' transition of clock the transmitter shall terminate any active bus transaction and set <BUS_NAME>_STB to '0'.
- If RST is set to '1' upon the '0' -> '1' transition of clock the receiver shall terminate any active bus transaction and set <BUS_NAME>_ACK to '0'.
- If RST is set to '0', normal operation shall commence as follows:
- The transmitter may insert wait states on the bus by setting <BUS_NAME>_STB '0'.
- The transmitter shall set <BUS_NAME>_STB to '1' to signify that data is valid.
- Once <BUS_NAME>_STB has been set to '1', it shall remain at '1' until the transaction completes.
- The transmitter shall ensure that <BUS_NAME> contains valid data for the entire period that <BUS_NAME>_STB is '1'.
- The transmitter may set <BUS_NAME> to any value when <BUS_NAME>_STB is '0'.
- The receiver may insert wait states on the bus by setting <BUS_NAME>_ACK to '0'.
- The receiver shall set <BUS_NAME>_ACK to '1' to signify that it is ready to receive data.
- Once <BUS_NAME>_ACK has been set to '1', it shall remain at '1' until the transaction completes.
- Whenever <BUS_NAME>_STB is '1' and <BUS_NAME>_ACK are '1', a bus transaction shall complete on the following '0' -> '1' transition of CLK.






- Both the transmitter and receiver may commence a new transaction without inserting any wait states.




- The receiver may delay a transaction by inserting wait states until the transmitter indicates that data is available.
- The transmitter shall not delay a transaction by inserting wait states until the receiver is ready to accept data.
- Deadlock would occur if both the transmitter and receiver delayed a transaction until the other was ready.