Advanced Port Usage

1	Advanced port usage	2
	Zipping and Unzipping data onto ports	3

1 Advanced port usage

When running out of 1-bit ports, it may be possible to use a 4-bit port instead to emulate four 1-bit ports, or to use a 32-bit port to emulate 32 1-bit ports.

1.1 module zip

This module can ZIP and UNZIP data onto 4-bit ports:

Functionality provided	Resources required		Status
	4-bit port		Memory
quad zip and out	1	1250 bytes	Implemented
quad zip and in	1	600 bytes	Implemented

2 Zipping and Unzipping data onto ports

The zip module can input words from a 4-bit port, and output words to a 4-bit port as if these were four 1-bit ports.

2.1 API

void outputWordsZipped(buffered out port:32 p, int b0, int b1, int b2,
int b3)

Function that outputs 4 words to a single 4-bit buffered port.

Each word is output to one of the 4 bits serialised.

- p port to output to
- **b0** word to output to bit 0 of port p
- **b1** word to output to bit 1 of port p
- **b2** word to output to bit 2 of port p
- **b3** word to output to bit 3 of port p

{int,int,int,int} i inputWordsZipped(buffered in port:32 p)
Function that inputs 4 words from a single 4-bit buffered port.

Each word is input from one of the 4 bits serialised.

• p - port to input from

Returns the four words, the first word is bit 0, the 4th return value is bit 3

2.2 Example

An example program is shown below. An input and/or output ports should be declared as buffered ports:

A function is called to output data, or to input data:

If you want to do both simultaneously, you will need to manually interleave the source code of the two functions.

The document is provided to you "AS IS" with no warranty of any kind, express or implied and shall have no liability in relation to its use. The author makes no representation that the information, or any particular implementation thereof, is or will be free from any claims of infringement and again, shall have no liability in relation to any such claims.