RIFFA: A Reusable Integration Framework For FPGA Accelerators

C/C++ Interface

The software interface is provided by bindings for C/C++. After installation all bindings are available in their respective runtime environments. The API is based on the notion of channels. RIFFA 2 can be configured to support between 1 - 12 independent channels. Each channel connects to an IP core and can be addressed by specifying the channel number from the user application. The channels are independent and thread safe. At most one thread should be used to access a single channel.

The C/C++ bindings are used by including the `<riffa.h>` header file and linking with the `-lriffa` library. Below is a complete example and an API listing.

```
1.  #include <stdio.h>;
2.  #include <stdlib.h>;
3.  #include <riffa.h>;
4.
5.  #define BUF_SIZE (1*1024*1024)
6.  unsigned int buf[BUF_SIZE];
7.
8.  int main(int argc, char* argv[]) {
9.    fpga_t * fpga;
10.   int fid = 0; // FPGA id
11.   int channel = 0; // FPGA channel
12.
13.   fpga = fpga_open(fid);
14.   fpga_send(fpga, channel, (void *)buf, BUF_SIZE, 0, 1, 0);
15.   fpga_recv(fpga, channel, (void *)buf, BUF_SIZE, 0);
16.   fpga_close(fpga);
17.   return 0;
18. }</riffa.h></stdlib.h></stdio.h>
```

This example first opens up FPGA with id 0. It then sends 4 MB of data (1 mega-words) to channel 0 with 0 destination offset, 0 timeout, and marks the transfer as last. The IP core on channel 0 is designed to send back some data. So the next call is to receive data from the same channel, up to 4 MB, with 0 timeout.

Note a few things:

1. We're using the same buffer to send data and receive data. This is not required, it just makes for a simpler example.
2. The timeout is set to 0, which may hang the application if there's a problem with the IP core logic.
3. In practice, you'd want to check the return values to see how much data was sent and received. You'd also probably want some error handling.

Using gcc, this example can be compiled as:

```
gcc tester.c -lriffa
```

Using Microsoft Visual Studio, you'll need to set your project configuration properties to find the `riffa.h` header file and add the import library as a dependency. Under C/C++ -> General -> Additional Include Directories, specify the path to the `riffa.h` header file under. Under Linker -> Input -> Additional Dependencies, specify the path to `riffa.lib`. Visual Studio should be able to find the `riffa.lib` without adding an additional dependency path.

API

```
int fpga_list(fpga_info_list * list);
```

Populates the `fpga_info_list` pointer with all FPGAs registered in the system. See `riffa_driver.h` for the `fpga_info_list` definition. Returns 0 on success, a negative value on error.

`list` - Pointer to a `fpga_info_list` struct to populate.

Returns:

0 on success, a negative value on error.

```
fpga_t * fpga_open(int id);
```

Initializes the FPGA specified by id. On success, returns a pointer to a `fpga_t` struct. On error, returns `NULL`. Each FPGA must be opened before any channels can be accessed. Once opened, any number of threads can use the `fpga_t` struct pointer.

id - Identifier for the FPGA (in single FPGA installations, this is always 0).

Returns:

A `fpga_t` struct pointer or `NULL`.

```
void fpga_close(fpga_t * fpga);
```

Cleans up memory/resources for the FPGA specified by the descriptor (pointer).

fpga - Pointer to `fpga_t` struct.

Returns:

Nothing.

```
int fpga_send(fpga_t * fpga, int chnl, void * data, int len, int destoff, int last, long long timeout);
```

Sends `len` words (4 byte words) from `data` to FPGA channel `chnl` using the `fpga_t` struct. The FPGA channel will be sent `len`, `destoff`, and `last`. The value of `destoff` is used to support sending data across multiple send transactions. Note that only the low 31 bits of this unsigned int are sent. If `last` is 1, the channel should interpret the end of this send as the end of a transaction. If `last` is 0, the channel should wait for additional sends before the end of the transaction. If `timeout` is non-zero, this call will send data and wait up to `timeout` ms for the FPGA to respond (between packets) before timing out.
If `timeout` is zero, this call may block indefinitely. Multiple threads sending on the same channel may result in corrupt data or error. This function is thread safe across channels. Returns the number of words sent.

fpga - Pointer to `fpga_t` structure.
chnl - Channel number over which to communicate.
data - Pointer to array of data to send. Note that the data transfer unit is a 32 bit word.
len - Length of data to send, in (32 bit) words. Thus a value of 4 means send 16 bytes.
destoff - Value sent to FPGA core to indicate where to start writing this data. Only the least significant 31 bits are sent (not all 32).
last - If 1, this transfer is the last in a sequence of transfers. If 0, this transfer is not the last in a sequence of transfers (more transfers to come).
timeout - Timeout value in ms. If 0, no timeout is specified. Otherwise, the PC will wait up to `timeout` ms in between PC/FPGA communications.

Returns:

The number of words sent.

```
int fpga_recv(fpga_t * fpga, int chnl, void * data, int len, long long timeout);
```

Receives data from the FPGA channel `chnl` to the `data` pointer, using the `fpga_t` struct. The FPGA channel can send any amount of data, so the data array should be large enough to accommodate. The `len` parameter specifies the actual size of the `data` buffer in words (4 byte words). The FPGA will specify an offset value which will determine where received data will start being written. If the amount of data plus the offset exceed the size of the `data` array, then the additional data will be discarded. If `timeout` is non-zero, this call will wait up to `timeout` ms for the FPGA to respond (between packets) before timing out. If `timeout` is zero, this call may block indefinitely. Multiple threads receiving on the same channel may result in corrupt data or error. This function is thread safe across channels. Returns the number of words received to the `data` array.

fpga - Pointer to `fpga_t` structure.
chnl - Channel number over which to communicate.
data - Pointer to buffer array where received data will be written.
len - Length of buffer array, in (32 bit) words. Thus a value of 4 means send 16 bytes.
timeout - Timeout value in ms. If 0, no timeout is specified. Otherwise, the PC will wait up to `timeout` ms in between PC/FPGA communications.

Returns:

The number of words received to the `data` array.

```
void fpga_reset(fpga_t * fpga);
```

Resets the state of the FPGA and all transfers across all channels. This is meant to be used as an alternative to rebooting if an error occurs while sending/receiving. Calling this function while other threads are sending or receiving will result in unexpected behavior.

fpga - Pointer to `fpga_t` structure.

Returns:

Nothing.