```python
from pynq import Overlay, allocate

# Load the bitstream
overlay = Overlay("adder.bit")
dma = overlay.axi_dma_0  # AXI DMA instance from the block design

# Allocate memory buffers
input_buffer = allocate(shape=(2,), dtype='int32')
output_buffer = allocate(shape=(1,), dtype='int32')

# Assign two integers to the input buffer
input_buffer[0] = 10
input_buffer[1] = 50

# Send the data using DMA (to your custom IP)
dma.sendchannel.transfer(input_buffer)
dma.recvchannel.transfer(output_buffer)

# Wait for the transfer to complete
dma.sendchannel.wait()
dma.recvchannel.wait()

# Check the output
print(f"Sent integers: {input_buffer[:]}")
print(f"Received sum: {output_buffer[0]}")
```


```python
from pynq import Overlay, allocate

overlay = Overlay("adder.bit")
dma = overlay.axi_dma_0

input_buffer = allocate(shape=(2,), dtype='int32')
output_buffer = allocate(shape=(1,), dtype='int32')

input_buffer[0] = 10
input_buffer[1] = 50

dma.sendchannel.transfer(input_buffer)
dma.recvchannel.transfer(output_buffer)

dma.sendchannel.wait()
dma.recvchannel.wait()

print(f"Sent integers: {input_buffer[:]}")
print(f"Received sum: {output_buffer[0]}")
```


Bitstream loading and DMA initialization

Allocate memory buffers and initialize with the values to be sent

Send the data to PL and receive the output from PL

Wait for the DMA transfers to complete