UNIVERSITY OF CALIFORNIA, BERKELEY
Department of Electrical Engineering and Computer Sciences
EE251B Advanced Digital Circuits and Systems

Spring 2024, Prof. Borivoje Nikolic
Homework 1

Issued: Thursday January 25, 2024
Due: Friday February 2, 2024

---

1 SystemVerilog Assertions (50 %)

(a) Explain (in regular English) what condition each of the following assertions checks for:

i. `assert property (!(mem_read && mem_write));`
Check that the signals `mem_read` and `mem_write` are never asserted simultaneously.

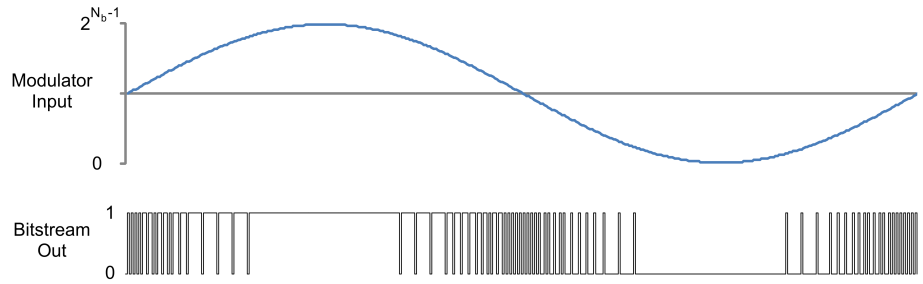
ii. `assert property (@(posedge clk) ready |-> ##1 valid);`
Every time synchronous signal `ready` is asserted, check that `valid` is asserted the next clock cycle.

iii. `assert property (@(posedge clk) ready |=> valid);`
Every time synchronous signal `ready` is asserted, check on the next cycle that `valid` is asserted.

iv. `assert property (@(posedge clk) ready |-> ##[1:2] valid);`
Every time synchronous signal `ready` is asserted, check that `valid` is asserted on the next cycle or the cycle after that.

v. `assert property (@(posedge clk) disable iff (rst) not sel[0] ##1 sel[1]);`
Check that synchronous signal `sel[1]` is never asserted on the next clock cycle after `sel[0]` is asserted, unless `rst` was asserted.

(b) A delta-sigma modulator is a common digital signal processing block that can be used to make a simple digital to analog converter using an FPGA output pin and an external RC filter. It converts a sequence of digital codes into a fast-switching bitstream, which can then be converted into a smooth analog signal by a lowpass filter. For example, here is a bitstream produced from a digitally sampled sine wave played back through a delta-sigma modulator:



One common implementation of a delta-sigma modulator is called the error feedback structure. An error feedback modulator works by taking an unsigned `input` vector of width `N` and adding it to an `N`-bit accumulator register on every clock cycle. The clock-synchronous carry-out from this addition is the bitstream. The clock runs at a much faster rate than the input update rate, allowing each digital input code to generate a long sequence of output bits with the correct average value. Write a parameterized Verilog module that implements this structure. Include a synchronous `reset` signal that sets the state of all registers to zero.

```
module efm #( parameter N = 8 ) (
   input clk,
   input reset,
   input [ N - 1 : 0 ] in,
   output reg bitstream
);

reg [ N - 1 : 0 ] acc;

always @ ( posedge clk ) begin
   if( reset ) begin
      acc <= { N{ 1'b0 } };
      bitstream <= 1'b0;
   end else begin
      { bistream, acc } <= acc + input;
   end
end

endmodule
```

Note: the parameter `N` doesn't specifically need to be set to 8, it's just good practice to set a default value.

Write a SystemVerilog assertion or assertions that check for the correct behavior of the accumulator and bitstream on each clock cycle. Make sure to include the **`reset`** signal.

```
assert property (@(posedge clk) disable iff (reset) {bistream, acc} ==
    $past(input) + $past(acc));
assert property (@(posedge clk) reset |=> (acc == 0) && (bitstream == 0);
```

Write a SystemVerilog coverage statement that detects whether the bitstream changes value during a verification simulation.

```
cover property(@(posedge clk) out !== $past(out));
```

2 System Interconnect (50 %)

You have been tasked with building a system interconnect for a system-on-chip with 1024 processing cores. You are considering several options for the on-chip system interconnect network:

i. 32-ary 2-mesh

ii. 32-ary 2-cube

iii. 16-ary 2-Cmesh4

iv. 32-ary 3-fly Clos

The application requires a packet size of 2048 bits.

(a) Calculate the required number of bits per unidirectional channel, for each network option, such that each of the networks can support the ideal throughput of 128 bits/cycle/core under uniform random traffic.

Hints: Start by calculating the total throughput of the network, then find the bisection bandwidth and number of unidirectional bisection channels for each network. Express all numerical answers in terms of integer powers of 2.

i. **32-ary 2-mesh**

First, let's calculate the total throughput (Θ_{total}) of the network:

$$\Theta_{\text{total}} = N_{\text{cores}} \cdot \Theta_{\text{core}} = 2^{10}\,\text{cores} \times 2^{7}\,\text{bits/cycle/core} = 2^{17}\,\text{bits/cycle} \tag{1}$$

Now we need to determine the bisection bandwidth (B_B) and the number of unidirectional bisection channels (B_C). A 4-ary 2-mesh network is shown in Figure 1. Let's solve for its B_B and B_C and then generalize the results.

The bisection cut goes through 4 bidirectional channels. Thus, we get $B_C = 4 \times 2 = 8$ unidirectional channels. We can also see that under uniform random traffic this topology should see a bisection bandwidth that is exactly half of the total throughput. Generalizing our results gives us:

$$B_C = 2\sqrt{N_{\text{cores}}} \tag{2}$$

$$B_B = \frac{\Theta_{\text{total}}}{2} \tag{3}$$

Now we can solve for the unidirectional bandwidth (b_C) by dividing the bisection bandwidth (B_B) by the number of bisection channels (B_C). Plugging in the numbers for a 32-ary 2-mesh gives us:

$$b_C = \frac{B_B}{B_C} = \frac{2^{17}\,\text{bits/cycle} \times 2^{-1}}{2 \times \sqrt{2^{10}}} = 2^{10}\,\text{bits/channel} \tag{4}$$

ii. **32-ary 2-cube**

A similar analysis to the previous section is performed for this topology. A smaller network, 4-ary 2-cube, is shown in Figure 2. The bisection cut goes through 8 bidirectional channels. Thus we get $B_C = 8 \times 2 = 16$ unidirectional channels. We can also see that

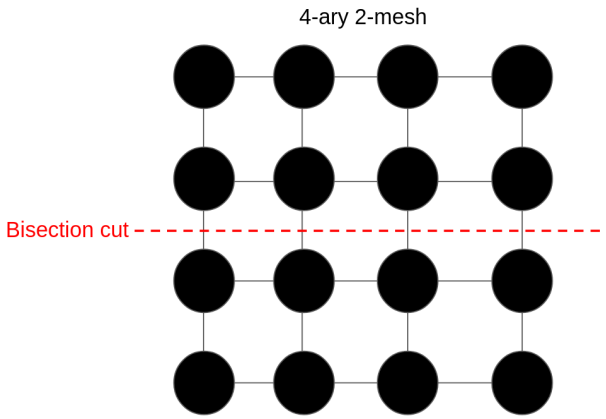


Figure 1

under uniform random traffic this topology should see a bisection bandwidth that is exactly half of the total throughput. Generalizing our results gives us:

$$B_C = 4\sqrt{N_{\text{cores}}} \tag{5}$$

$$B_B = \frac{\Theta_{\text{total}}}{2} \tag{6}$$

Now we can solve for the unidirectional bandwidth (b_C) by dividing the bisection bandwidth (B_B) by the number of bisection channels (B_C). Plugging in the numbers for a 32-ary 2-mesh gives us:

$$b_C = \frac{B_B}{B_C} = \frac{2^{17}\,\text{bits/cycle} \times 2^{-1}}{4 \times \sqrt{2^{10}}} = 2^9\,\text{bits/channel} \tag{7}$$

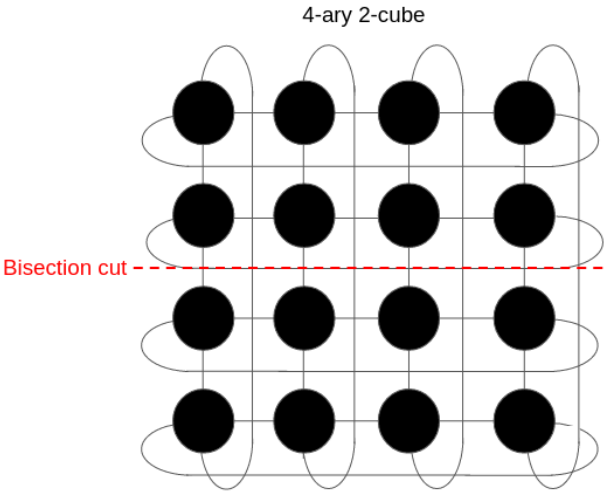


Figure 2

iii. **16-ary 2-Cmesh4**

A smaller network, 2-ary 2-Cmesh4, is shown in Figure 3. The bisection cut goes through 2 bidirectional channels. Thus we get $B_C = 2 \times 2 = 4$ unidirectional channels. We can

also see that under uniform random traffic this topology should see a bisection bandwidth that is exactly half of the total throughput. Generalizing our results gives us:

$$B_C = 2\sqrt{\frac{N_{\text{cores}}}{4}} = \sqrt{N_{\text{cores}}} \tag{8}$$

$$B_B = \frac{\Theta_{\text{total}}}{2} \tag{9}$$

Now we can solve for the unidirectional bandwidth (b_C) by dividing the bisection bandwidth (B_B) by the number of bisection channels (B_C). Plugging in the numbers for a 16-ary 2-Cmesh4 gives us:

$$b_C = \frac{B_B}{B_C} = \frac{2^{17}\,\text{bits/cycle} \times 2^{-1}}{\sqrt{2^{10}}} = 2^{11}\,\text{bits/channel} \tag{10}$$

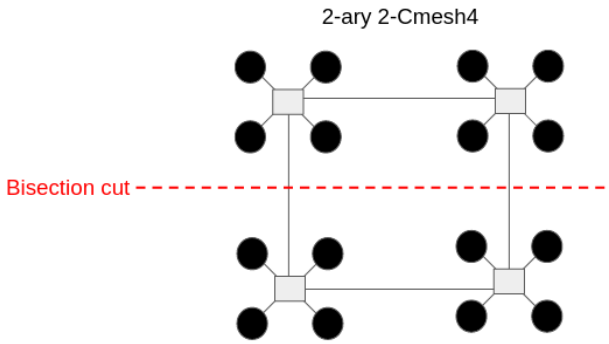


Figure 3

iv. **32-ary 3-fly Clos**

A smaller network, 2-ary 3-fly Clos, is shown in Figure 4. The bisection cut goes through 2 unidirectional channels in each direction. Thus we get $B_C = 2 \times 2 = 4$ unidirectional channels in total across the bisection. We can also see that under uniform random traffic this topology should see a bisection bandwidth that is exactly equal to the total throughput. This is different than the previous topologies.

In Figure 4, the labeled "1/4" on all the channels represents the portion of the total network throughput that any given channel is seeing under uniform random traffic. Each core inputs 1/4 of the total throughput into the first stage (because we have 4 cores total in this example). The first stage switches will split the 1/4 input throughput from one node to two 1/8 throughput streams onto the outgoing channels. This will give us 1/4 of the total network throughput on both outgoing channels (1/8 from one input and 1/8 from the other). This is repeated until we output 1/4 of the total network throughput to each of the core's receivers. Now we can draw our bisection line, shown in red, and calculate the ratio of total throughput in our bisection cut. It should be very clear that the ratio is 1:1 because we have 4 channels crossing the bisection each carrying 1/4 of the total data throughput.

Generalizing our results gives us:

$$B_C = N_{\text{cores}} \tag{11}$$

$$B_B = \Theta_{\text{total}} \tag{12}$$

Now we can solve for the unidirectional bandwidth (b_C) by dividing the bisection bandwidth (B_B) by the number of bisection channels (B_C). Plugging in the numbers for a 32-ary 3-fly gives us:

$$b_C = \frac{B_B}{B_C} = \frac{2^{17}\,\text{bits/cycle}}{2^{10}} = 2^7\,\text{bits/channel} \tag{13}$$

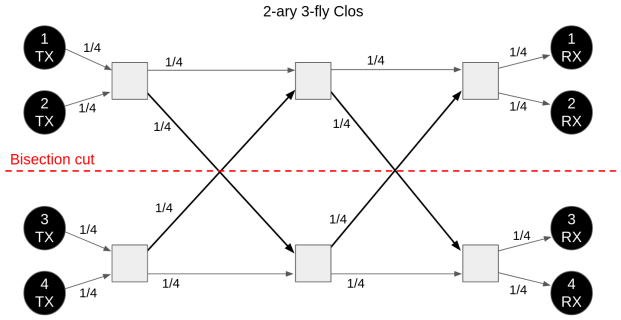


Figure 4

(b) Show the worst-case zero-load latency breakdown for each interconnect network. Independent of your calculations above, assume the following:

- The flit size is 64 bits and all channel widths are matched to that flit size.
- Assume that the latency of each channel (core-to-router or router-to-router) is 1 cycle.
- Assume the latency of each router to be 2 cycles.

We can calculate the zero-load latency as:

$$T_0 = H \cdot t_r + \frac{D}{v} + \frac{L}{b} \tag{14}$$

where H is the hop count (i.e. the number of routers on the path), t_r is the router delay, D is the total physical distance the signal must traverse, v is the speed of the signal, L is the length of the signal in bits, and b is the channel bandwidth.

We have been given a channel latency of 1 cycle for core-to-router and router-to-router. We also know that our serialization delay is the number of cycles required to serialize the packet across a channel of width b_C (this is the number we calculated in the previous question). Thus, our latency equation can be rewritten in terms of "cycles", where K is the number of channels along the path and H is the number of routers along the path:

$$T_0 = (H\,\text{routers})\cdot(2\,\text{cycles/router}) + (K\,\text{channels})\cdot(1\,\text{cycle/channel}) + (\text{flits/packet})\cdot(\text{cycles/flit}) \tag{15}$$

With a packet size of 2048 bits, this gives us $\frac{2048\text{bits/packet}}{64\,\text{bits/flit}} = 32\,\text{flits/packet}$. "cycles per flit" is 1 since the channel width (b_C) is equal to the flit size.

We want the worst-case zero-load latency and thus we have to find the diameter ($H_{\max}$) and the associated number of channels K of the network to plug into our latency equation. $H_{\max}$ is the longest minimum-hop path between two cores.

i. **32-ary 2-mesh**

For this topology, $H_{\max}$ is found from one corner node to the opposite corner node. We can now solve for the latency of the 32-ary 2-mesh:

$$H_{\max} = 2\sqrt{N} - 1 = 63 \tag{16}$$

$$K = 1 + (H_{\max} - 1) + 1 = 64 \tag{17}$$

$$T_0 = H_{\max} \times 2 + K \times 1 + 32 \times 1 = 222\,\text{cycles} \tag{18}$$

ii. **32-ary 2-cube**

$H_{\max}$ for this topology is found by going from the middle node one one edge (e.g. west edge) to the middle node on the adjacent edge (e.g. south edge). Thus,

$$H_{\max} = \sqrt{N} + 1 = 33 \tag{19}$$

$$K = 1 + (H_{\max} - 1) + 1 = 34 \tag{20}$$

$$T_0 = H_{\max} \times 2 + K \times 1 + 32 \times 1 = 132\,\text{cycles} \tag{21}$$

iii. **16-ary 2-Cmesh4**

$H_{\max}$ for this topology is found by going from one of the corner node clusters to a node in the opposite corner node cluster.

$$H_{\max} = \sqrt{N} - 1 = 31 \tag{22}$$

$$K = 1 + (H_{\max} - 1) + 1 = 32 \tag{23}$$

$$T_0 = H_{\max} \times 2 + K \times 1 + 32 \times 1 = 126\,\text{cycles} \tag{24}$$

iv. **32-ary 3-fly Clos**

$H_{\max}$ for this topology is found by going from any node to any other node.

$$H_{\max} = 3 \tag{25}$$

$$K = 1 + (H_{\max} - 1) + 1 = 4 \tag{26}$$

$$T_0 = H_{\max} \times 2 + K \times 1 + 32 \times 1 = 42\,\text{cycles} \tag{27}$$