ECE 5745 Complex Digital ASIC Design

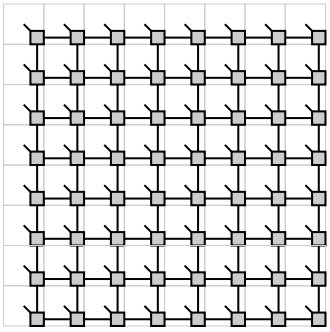
Practice Problem

http://www.csl.cornell.edu/courses/ece5745
School of Electrical and Computer Engineering
Cornell University

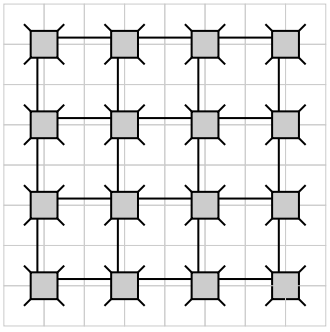
revision: 2021-03-28-22-07

Problem 1. Initial Mesh Network Analysis

In this problem, we will be analyzing two different mesh network topologies suitable for use in an on-chip processor-to-memory network. The figure below illustrates how the two topologies are integrated into a tiled chip-multiprocessor which contains a total of 64 tiles. Each tile contains a processor, private L1 caches, and a bank of the L2 unified cache which is shared across all processors. The private L1 caches send memory read/write requests (i.e., for refills, evictions, and cache coherence) to a specific L2 cache bank somewhere on the chip, and the L2 cache banks send back memory read/write responses. We can assume the global address space is cache-line interleaved across the L2 cache banks. *To simplify our analysis we will only consider the memory request network, and we will ignore the memory response network.* For the memory request network, the L1 instruction and data caches share an input terminal into the network; each L2 cache banks serves as an output terminal from the network.



mesh8x8



cmesh4x4

Each tile is 2×2 mm, so the overall chip size is about 16×16 mm or 256 mm^2. We will assume that the processors and caches can run at 500 MHz, and that we will pipeline the routers and the channels so that neither creates a critical path and limits the cycle time. For all problems, assume that the packet length is also 128 bits. For all problems, you can ignore the latency of the channels from the input terminals (i.e., the L1 caches) to the first router, and you can also ignore the channels from the last router to the output terminal (i.e, the L2 cache banks). For this problem, assume that in both topologies it takes one cycle for a phit to traverse the edge of one tile (i.e., 2 mm). For both topologies assume dimension-ordered routing (which happens to also achieve the ideal throughput).

The *mesh8x8* topology is a 2-dimensional 8-ary mesh topology where 64 routers are arranged in an eight-by-eight grid. There is one input/output terminal pair per router, and thus each router has five input ports and five output ports. Assume that the five-port routers have a two-cycle latency: one cycle for route computation and switch arbitration, and one cycle for switch traversal. For this topology the bandwidth of each channel is 64 bits, and the latency of each channel is one cycle. The *cmesh4x4* topology is a concentrated 2-dimensional 4-ary mesh topology where 16 routers are arranged in a four-by-four grid. There are four input/output terminal pairs per router, and thus

each router has eight input ports and eight output ports. Assume that the eight-port routers have a three-cycle latency: one cycle for route computation and switch arbitration, and two cycles for switch traversal due to the larger number of ports. For this topology the bandwidth of each channel is 128 bits, and the latency of each channel is two cycles.

Part 1.A Mesh Network Ideal Throughput

Calculate the ideal terminal throughput for both mesh topologies assuming uniform random traffic. As always, you should assume perfect routing and perfect flow control. **Qualitatively explain why your results make intuitive sense.**

Recall that under uniform random traffic half the traffic from each input terminal will cross the bisection assuming ideal routing. In other words, the max channel load is $\gamma_{max} = N/2B_C$ where N is the number of terminals and B_C is the number of bisection channels. Also recall that the ideal terminal throughput assuming uniform random traffic is just $\Theta_{ideal} = b/\gamma_{max}$ where b is the channel bandwidth.

For the *mesh8x8* topology, N is 64 and B_C is 16 so the max channel load is 2. The channel bandwidth (b) is 64 bits/cycle, so the ideal terminal throughput is $b/\gamma_{max} = 64/2 = 32$ bits/cycle.

For the *cmesh4x4* topology, N is 64 and B_C is 8 so the max channel load is 4. The channel bandwidth (b) is 128 bits/cycle, so the ideal terminal throughput is $b/\gamma_{max} = 128/4 = 32$ bits/cycle.

So the *cmesh4x4* topology has half the number of bisection channels but twice the channel bandwidth compared to the *mesh8x8* topology. This means both topologies have the same bisection bandwidth and the same ideal terminal throughput under uniform random traffic. One might argue that this makes comparing these two topologies a fair comparison.

Part 1.B Mesh Network Zero-Load Latency

Calculate the average zero-load latency in cycles for both mesh topologies assuming uniform random traffic. Remember that we are completely ignoring the channels from the input/output terminals to the first/last routers. **Qualitatively explain why your results make intuitive sense.**

Recall that the zero-load latency is defined as follows:

$$t_0 = H_r \times t_r + H_c \times t_c + (L/b)$$

With the following definitions:

- H_r : Average number of router hops assuming uniform random traffic
- t_r : Average per-hop router latency in cycles assuming uniform random traffic
- H_c : Average number of channel hops assuming uniform random traffic
- t_c : Average per-hop channel latency in cycles assuming uniform random traffic
- L/b : Serialization latency in cycles, L = msg length, b = channel bandwidth in bits/cycle

The per-hop router and channel latencies are given in the problem. We need to calculate H_r and H_c. Since we know $H_c = H_r - 1$, we can just focus on calculating H_r. Here is a simple Python script that calculates H_r for both the *mesh8x8* and *cmesh4x4* topologies.

```
def calc_avg_hops_per_src( size, src_x, src_y ):
  sum = 0
  for x in xrange(size):
    for y in xrange(size):
```

```
      sum += abs(x-src_x) + abs(y-src_y) + 1
  return sum/(size*size*1.0)

def calc_avg_hops( size ):
  sum = 0
  for x in xrange(size):
    for y in xrange(size):
      sum += calc_avg_hops_per_src( size, x, y )

  return sum/(size*size*1.0)

print "H_r for mesh8x8  =", calc_avg_hops(8)
print "H_r for cmesh8x8 =", calc_avg_hops(4)
```

The `calc_avg_hops_per_src` function calculates the average number of router hops from a given router to every other router (including itself). The `calc_avg_hops` function calls `calc_avg_hops_per_src` for every router. Both functions are parameterized by the number of routers. Using this script we find that H_r is 6.25 for the *mesh8x8* topology, and H_r is 3.5 for the *cmesh4x4* topology. Therefore, for the *mesh8x8* topology, the zero-load latency is:

$$t_0 = H_r \times t_r + H_c \times t_c + (L/b) = 6.25 \times 2 + 5.25 \times 1 + 128/64 = 19.75 \text{ cycles}$$

For the *cmesh4x4* topology, the zero-load latency is:

$$t_0 = H_r \times t_r + H_c \times t_c + (L/b) = 3.5 \times 3 + 2.5 \times 2 + 128/128 = 16.5 \text{ cycles}$$

The zero-load latency of the *cmesh4x4* topology is slightly lower than the zero-load latency of the *mesh8x8* topology for two reasons: (1) the *cmesh4x4* has lower serializer latency, and (2) the increased router/channel latencies are outweighed by the much lower router/channel hop counts.

Part 1.C Mesh Network Energy and Area

Qualitatively compare and contrast the energy and area of both mesh topologies.

In terms of area, we must consider both the channel and router area. The total cumulative length of all of wires used in the channels will actually be the same across both topologies. Again, the *cmesh4x4* topology has half as many channels compared to the *mesh8x8* topology, but these channels are twice as wide. The *cmesh4x4* channels are twice as long, but we still need the same amount of metal wires in the *mesh8x8* topology. The might be a small difference in the number of channel pipeline registers between the two topologies. The *cmesh4x4* topology has 16 routers compared to 64 routers in the *mesh8x8* topology, but it is difficult to say how the router area scales with radix. It could scale quadratically with radix because of the router cross-bar, but then again there is additional area consumed in buffers which likely scales linearly with radix. Leveraging multiple metal layers in the cross-bar can also impact the scaling. Also note that the total chip area is 256 mm^2 which is quite large. We might expect that the area of either network is a small percentage of the overall chip area, especially if we can route the channels over other logic in the tile.

In terms of energy, we need to consider the amount of work required to route a packet through each topology. If we consider a single channel, then sending a packet across the longer channels in the *cmesh4x4* topology will take more energy compared to sending a packet across the shorter channels in the *mesh8x8*. Note that the channel bitwidth does not matter here; the number of

bits in the packet is the same across both topologies. However, the total energy spent in the channels to send a packet from one terminal to another will likely be very similar across both topologies; each channel is longer in the *cmesh4x4* topology, but a packet travels across fewer of these channels. Another way to view this, is that both networks basically send a packet across the (roughly) minimal Manhattan distance. The router energy will definitely be different; a higher radix router will certainly require more energy than a lower radix router. However, as with the area comparison it is difficult to draw any concrete conclusions. In the *cmesh4x4* topology, each packet must use more energy to travel through each high-radix router, but the packet goes through fewer routers overall.

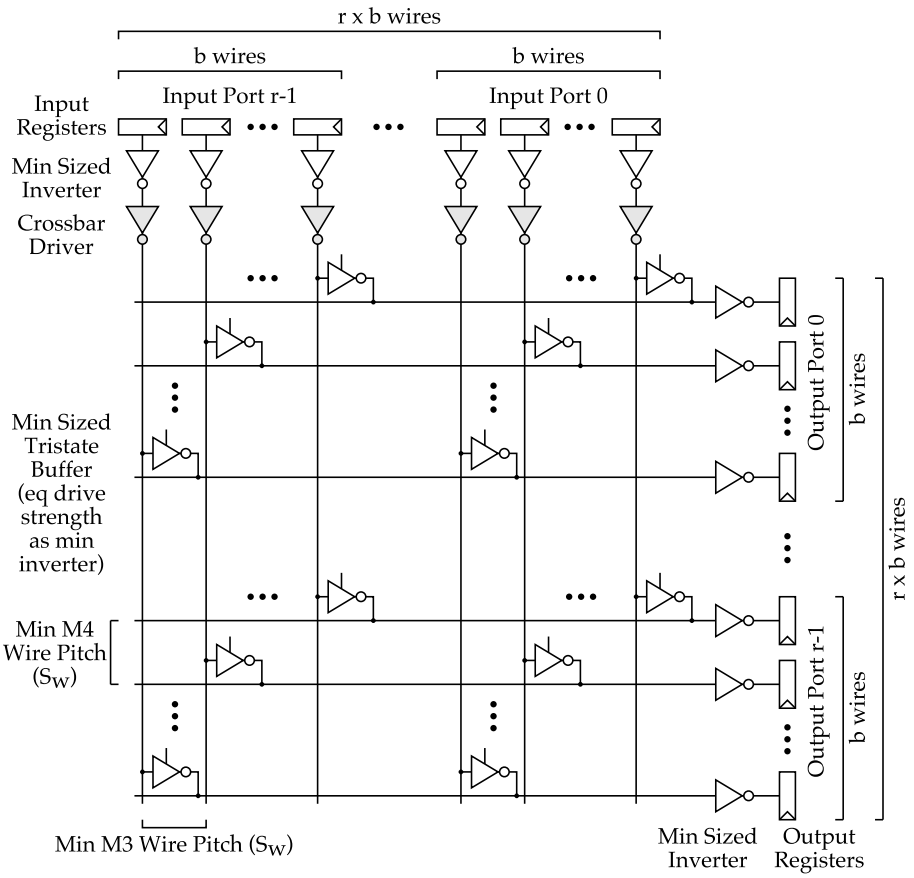
From this discussion, it should be clear that it is actually quite difficult to draw any compelling qualitative conclusions about area or energy from this architecture-level analysis. In the remainder of this problem set, we will attempt to use first-order circuit-level analysis to challenge some of our initial assumptions on router/channel latencies and to craft a more compelling area/energy comparison between the two topologies.

Problem 2. Network Router Crossbar Design

In this problem, you will be exploring the cycle-time, energy, and area of a crossbar suitable for use in mesh networks discussed in the first problem. The crossbar has the same number of input and output ports, and each port can support a certain number of bits per cycle. The number of ports is also called the crossbar's radix (r) and the number of bits per cycle is also called the channel bandwidth (b). Assume that we can send one bit per wire per cycle, so b is also the number of wires per port. For the purposes of this problem you can ignore the impact the crossbar control signals have on cycle-time, energy, and area; you will be focusing on the datapath portion of the crossbar. The crossbar microarchitecture is shown below.

Notice that the input ports and output ports are registered, so we have allocated a single cycle to traverse the crossbar. This is a "cross-point crossbar" which uses tri-states in the middle of the crossbar to control which input port can write which output port. You should assume that the first and last inverters in the pipeline stage are minimum sized. *You should assume that the tri-states are sized such that they still have the same drive strength as the canonical minimum-sized inverter.* The tri-states are implemented with a single stage of static CMOS logic (i.e., they do not use transmission gates). Just to be super clear, you do not need to do any optimal sizing on the tri-states; just size the tri-states such that they have the same drive-strength as the canonical minimum-sized inverter. This means the only gate which needs to be sized is the crossbar driver (shown in gray).

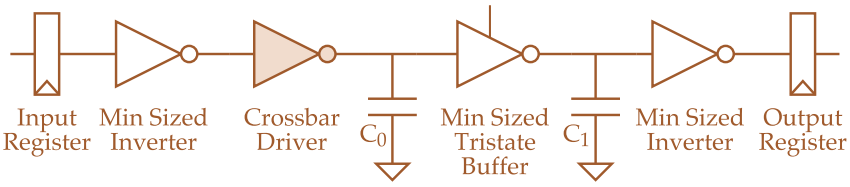
You should assume that the vertical wires are on metal layer 3 and the horizontal wires are on metal layer 4. All wires are spaced at minimum pitch. The tri-states are hidden underneath the crossbar wiring, even though for clarity in the diagram this appears not to be the case. The wires are short enough that you can ignore wire resistance, but you must factor in the wire capacitance in the crossbar wiring. See Appendix A for technology parameter assumptions for this problem.



Part 2.A Crossbar Cycle-Time

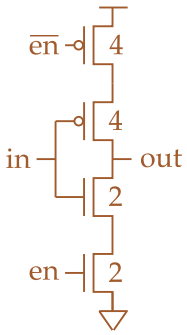
Derive the cycle time for the crossbar traversal stage as a function of r and b in units of τ. The cycle time should include the clock-to-q propagation time for the registers at the beginning of the stage, the worst case path through the crossbar, and the setup time for the registers at the end of the stage. You should optimally size the crossbar driver to minimize delay. *You must factor in the capacitance of the crossbar wiring, but you can ignore the resistance of this wiring.* **Calculate the cycle time in picoseconds for two scenarios: $r = 5, b = 64$ and $r = 8, b = 128$. You must show your work.**

The critical path is basically from the left most bit of the left most port to the bottom bit of the bottom port in the figure. This bit will have to drive the longest crossbar wires. Here is an abstract view of just the critical path:



C_0 is the wire capacitance for the vertical crossbar wire along with the associated input gate capacitance of the tristate buffers connected to that wire, and C_1 is the wire capacitance for the horizontal crossbar wire along with the associated diffusion capacitance for the the tristate buffers that can drive the same wire.

Recall that a tri-state buffer is implemented as shown to the right, and thus has a logical effort of 2 and a parasitic delay of 2. A canonical minimum sized tri-state buffer has an input gate capacitance of 6C and corresponding diffusion capacitance of 6C.

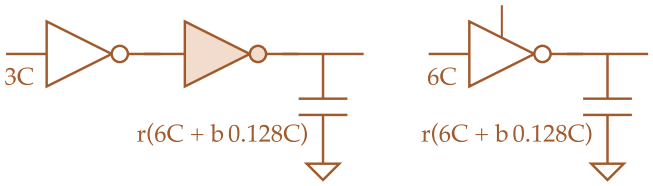


This path has large fixed capacitances so as discussed in lecture we can simplify our analysis by breaking this path into two subpaths: the first subpath starts at the minimum sized inverter after the input register and goes to the gate of the tristate; the second subpath starts at the tristate and goes to the gate of the minimum sized inverter right before the output register.

The output load capacitance for the first subpath (C_0) will be the gate capacitance of all of the tristates along the vertical wire along with the wire capacitance itself. The output load capacitance for the second subpath (C_1) will be the parasitic diffusion capacitance of the tristates along the horizontal wire along with the wire capacitance itself. These values are calculated below.

$$\begin{aligned} C_0 &= r\,6C + (r\,b\,(0.32 \times 0.4C)) = r\,(6C + b\,0.128C) \\ C_1 &= r\,6C + (r\,b\,(0.32 \times 0.4C)) = r\,(6C + b\,0.128C) \end{aligned}$$

Not surprisingly these are the same, since the gate capacitance and diffusion capacitance for an inverter are assumed to be similar and the wire lengths for both paths are the same. Note that we ignored the gate capacitance for the minimum sized inverter before the output register in C_1; this is negligible compared to the wire and diffusion capacitances. These two subpaths and known gate and load capacitances are shown below.



First Subpath Second Subpath

So the cycle time will be made up of five parts:

$$t_{pd,xbar} = t_{pcq} + t_{pd,0} + t_{pd,1} + t_{inv} + t_{setup}$$

where $t_{pd,0}$ is the delay through the first subpath and $t_{pd,1}$ is the delay through the second subpath. We know t_{pcq}, t_{inv}, t_{setup}, so basically we just need to find out $t_{pd,0}$ and $t_{pd,1}$. We can use logical effort to determine the optimal delay (in units of τ) through the first subpath.

$$\begin{aligned} H &= C_{out}/C_{in} = r\,(6C + b\,0.128C)/3C \\ &= r\,(2 + 0.043\,b)) \\ F &= GBH = 1 \times 1 \times (r\,(2 + 0.043\,b)) \\ P &= 1 + 1 = 2 \\ D_{opt} &= NF^{1/N} + P = 2\sqrt{r\,(2 + 0.043\,b)} + 2 \end{aligned}$$

We can also use logical effort to quickly calculate the delay through the second subpath. Alternatively, we could use a little RC circuit with the Elmore delay; both approaches should give the same answer. The logical effort of the tristate is 2 and the parasitic delay is also 2. We need to be a little careful because we included the parasitic delay of the driving tristate in the load capacitance for the second subpath. Let's ignore this.

$$\begin{aligned} h &= C_{out}/C_{in} = r\,(6C + b\,0.128C)/6C \\ &= r\,(1 + 0.021\,b) \\ d &= gh + p = 2 \times (r\,(1 + 0.021\,b)) + 2 \\ &= (r\,(2 + 0.043\,b)) + 2 \end{aligned}$$

Now we have enough information to be able to create an equation for the cycle time as a function of r and b.

$$\begin{aligned} t_{pd,xbar} &= t_{pcq} + t_{pd,0} + t_{pd,1} + t_{inv} + t_{setup} \\ &= 8 + (2\sqrt{r\,(2 + 0.043\,b)} + 2) + (r\,(2 + 0.043\,b) + 2) + 2 + 10 \\ &= 2\sqrt{r\,(2 + 0.043\,b)} + r\,(2 + 0.043\,b) + 24 \end{aligned}$$

For the $r = 5$, $b = 64$ configuration the cycle time is as follows:

$$\begin{aligned} t_{pd,xbar} &= 2\sqrt{r\,(2 + 0.043\,b)} + r\,(2 + 0.043\,b) + 24 \\ &= 2\sqrt{5\,(2 + 0.043 \cdot 64)} + 5\,(2 + 0.043 \cdot 64) + 24 \\ &= 57.51\tau = 426\,\text{ps} \end{aligned}$$

For the $r = 8$, $b = 128$ configuration the cycle time is as follows:

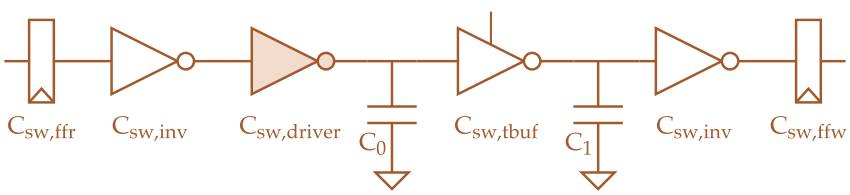
$$\begin{aligned} t_{pd,xbar} &= 2\sqrt{r\,(2 + 0.043\,b)} + r\,(2 + 0.043\,b) + 24 \\ &= 2\sqrt{8\,(2 + 0.043 \cdot 128)} + 8\,(2 + 0.043 \cdot 128) + 24 \\ &= 99.53\tau = 736\,\text{ps} \end{aligned}$$

These cycle times are pretty fast compared to a 500 MHz clock (i.e., cycle time of 2 ns or equivalently $1/500\,\text{MHz} \times 0.135\,\tau/\text{ps} = 270\,\tau$).

Part 2.B Crossbar Energy

Derive the worst-case energy for a single phit to go through the crossbar traversal stage as a function of r and b in units of Joules. A phit is b-bits of data going from an input port to the destination output port. To simplify our analysis, just calculate the energy for the worst-case path for a single bit assuming an activity factor of one and multiply by the number of bits in a phit. **Calculate the worst-case energy for a single phit in Joules for two scenarios: $r = 5, b = 64$ and $r = 8, b = 128$. You must show your work.**

To determine the energy for a single phit to traverse the crossbar, we first need to calculate the energy for a single bit to traverse the crossbar, and that means we need to calculate the total switched capacitance for a single bit. The diagram below shows all of the gate and parasitic capacitances (ignoring intra-gate capacitances).



The values for $C_{sw,ffr}$ and $C_{sw,ffw}$ are given in the appendix. The value for $C_{sw,inv}$ is 3C for the gate capacitance and 3C for the diffusion capacitance for a total switched capacitance of 6C. The value for $C_{sw,tbuf}$ is 6C for the gate capacitance and 6C for the diffusion capacitance (ignoring the energy overhead of the control signals to setup the crossbar). Technically, we shouldn't include $C_{sw,tbuf}$ because we account for the tri-state's gate capacitance in C_0 and the tri-state's diffusion capacitance in C_1. The values for C_0 and C_1 were calculated in the previous part.

To determine the value of $C_{sw,driver}$ we have to figure out the input gate capacitance for the buffer as sized using logical effort earlier in the problem.

$$\begin{aligned} F &= r\,(2+0.043\,b) \\ f_{opt} &= \sqrt{r\,(2+0.043\,b)} \\ C_{in} &= C_{out} \times g / f_{opt} = \frac{r\,(6+0.128\,b)}{\sqrt{r\,(2+0.043\,b)}} = \frac{3r\,(2+0.043\,b)}{\sqrt{r\,(2+0.043\,b)}} = 3\sqrt{r\,(2+0.043\,b)} \end{aligned}$$

Now we can calculate the total switched capacitance in the worst case as a function of r and b. Notice that the switched capacitance for the driver is twice the input capacitance calculated above because we need to account for the switched parasitic capacitance as well.

$$\begin{aligned} C_{sw} &= C_{sw,ffr} + C_{sw,inv} + C_{sw,driver} + C_0 + C_1 + C_{sw,inv} + C_{sw,ffw} \\ &= 25 + 6 + 2 \cdot 3\sqrt{r\,(2+0.043\,b)} + r\,(6+0.128\,b) + r\,(6+0.128\,b) + 6 + 25 \\ &= 6\sqrt{r\,(2+0.043\,b)} + 2r\,(6+0.128\,b) + 62 \end{aligned}$$

This is units of C where C is the gate capacitance of an NMOS in a canonical minimum sized inverter. Again, note that we do not explicitly include $C_{sw,tbuf}$ because this is factored into C_0 and C_1.

We can now derive an equation for the energy per bit (E_b) and energy per phit (E_{phit}) as a function of r and b.

$$\begin{aligned} E_b &= 0.5 C_{sw} V_{DD}^2 \\ &= 0.5 \left(6\sqrt{r\,(2+0.043\,b)} + 2r\,(6+0.128\,b) + 62 \right) C V_{DD}^2 \\ E_{phit} &= b\,0.5 \left(6\sqrt{r\,(2+0.043\,b)} + 2r\,(6+0.128\,b) + 62 \right) C V_{DD}^2 \end{aligned}$$

For the $r = 5$, $b = 64$ configuration the energy per phit is as follows assuming $V_{DD} = 1\,\text{V}$ as listed in the appendix.

$$\begin{aligned} C_{sw} &= 6\sqrt{r\,(2+0.043\,b)} + 2r\,(6+0.128\,b) + 62 \\ &= 6\sqrt{5\,(2+0.043 \cdot 64)} + 2 \cdot 5\,(6+0.128 \cdot 64) + 62 = 233C \\ E_{phit} &= b \times 0.5 \times 233C \times V_{DD}^2 \\ &= 64 \times 0.5 \times 233 \times 0.5 \times (1)^2 \\ &= 3.7\,\text{pJ} \end{aligned}$$

For the $r = 8$, $b = 128$ configuration the energy per phit is as follows assuming $V_{DD} = 1\,\text{V}$ as listed in the appendix.

$$\begin{aligned} C_{sw} &= 6\sqrt{r\,(2+0.043\,b)} + 2r\,(6+0.128\,b) + 62 \\ &= 6\sqrt{8\,(2+0.043\cdot 128)} + 2\cdot 8\,(6+0.128\cdot 128) + 62 = 467C \\ E_{phit} &= b \times 0.5 \times 467C \times V_{DD}^2 \\ &= 128 \times 0.5 \times 467 \times 0.5 \times (1)^2 \\ &= 14.8\,\text{pJ} \end{aligned}$$

Part 2.C Crossbar Area

Derive the area for the crossbar traversal stage as a function of r and b in units of square micron. Assume that the area of the crossbar is dominated by the crossbar wiring, so you can ignore the area of the pipeline registers, minimum sized inverters, and crossbar driver. *You can also assume that the tri-states are hidden beneath the crossbar wiring, so that the crossbar wiring area is just a function of the number of wires and the wire pitch.* **Calculate the area in square micron for two scenarios: $r = 5, b = 64$ and $r = 8, b = 128$. You must show your work.**

Since we are ignoring the area of the pipeline registers, buffers, and tri-states, the area of the crossbar is quite straight-forward. It is just the number of bits times the number of ports times the wire pitch per side.

$$A_{xbar} = (r \times b \times S_w)^2$$

For $r = 5$, $b = 64$ configurations the area in square micron is as follows:

$$\begin{aligned} A_{xbar} &= (r \times b \times S_w)^2 = (5 \times 64 \times 0.32)^2 \\ &= (102.40)^2 = 10,486\ \mu\text{m}^2 \end{aligned}$$

For $r = 8$, $b = 128$ configurations the area in square micron is as follows:

$$\begin{aligned} A_{xbar} &= (r \times b \times S_w)^2 = (8 \times 128 \times 0.32)^2 \\ &= (327.68)^2 = 107,374\ \mu\text{m}^2 \end{aligned}$$

Note that 107,374 μm^2 seems large, but this is equivalent to 0.1 mm^2 or far less than 1% of the total die area. In other words, it is not clear if the factor of 10× increase in area will really be a significant disadvantage.

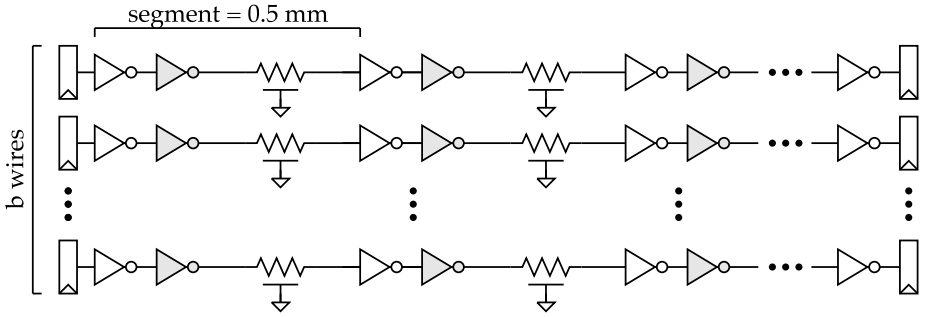
Problem 3. Network Channel Design

In this problem, you will be exploring the cycle-time, energy, and area of a channel suitable for use in mesh networks discussed in the first problem. Each channel has b wires and includes n non-inverting repeaters. The channel microarchitecture is shown below.

Notice that the input and output of the channel are registered, so we have allocated a single cycle to traverse the channel. Also notice that we always use a single minimum sized inverter as the final receiver in the channel. This means the channel as a whole is inverting, but we can easily compensate for this by using the complemented output of the final register. The first inverter in each non-inverting repeater is always minimum size.

Each segment of the channel is always 0.5 mm long. You can assume that the total length of the channel is always a multiple of 0.5 mm (i.e., there will always be an integer number of segments

in the channel). You should assume that the channel is implemented on a combination of wires on metal layer 3 and 4 (depending on the channel's orientation and whether there are any turns in the channel). All wires are spaced at minimum pitch. The non-inverting repeaters are hidden underneath the channel wiring, even though for clarity in the diagram this appears not to be the case. The wires are long enough that you cannot ignore wire resistance; you must factor in both the wire resistance and capacitance in the channel wiring. See Appendix A for technology parameter assumptions for this problem.



Part 3.A Channel Cycle-Time

Derive the cycle time for the channel traversal stage in units of τ as a function of the number of segments n. The cycle time should include the clock-to-q propagation time for the input registers, the delay through the repeaters, the wire delay, and the setup time for the output registers. You should optimally size the second inverter of each non-inverting repeater. Remember that the first inverter of each non-inverting repeater is always minimum size. **Set the cycle time to be 2 ns (or equivalently 1/500 MHz $\times$ 0.135 τ/ps = 270 τ) and determine the longest channel (i.e., the number of segments) that can still meet this timing constraint.** Round this length down so we can use an integer number of segments. This is essentially how far we can go in a cycle when running at 500 MHz. **You must show your work.**

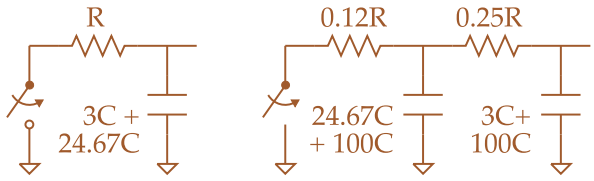
First, we calculate the wire segment capacitance (C_{wseg}) and the wire segment resistance (R_{wseg}). Let l_{wseg} be the length of each segment (given to be 500 µm).

$$\begin{aligned} C_{wseg} &= l_{seg} \times C_w = 500 \times 0.4C = 200C \\ R_{wseg} &= l_{seg} \times R_w = 500 \times 0.5R/1000 = 0.25R \end{aligned}$$

Now we can use logical effort to determine the optimal sizing for the second inverter in the non-inverting repeater. As a rough first approximation we can just lump all of the wire capacitance and the gate capacitance of the next segment as the output capacitance for the repeater.

$$\begin{aligned} F &= GBH = 1 \times 1 \times (203/3) = 67.67 \\ f_{opt} &= F^{1/N} = 67.67^{1/2} = 8.23 \\ C_{in,1} &= C_{out} \times g_i / f_{opt} = 203C \times 1/8.23 = 24.67C \end{aligned}$$

We cannot really calculate the optimal delay of the repeater using logical effort, since we need to factor in the wire resistance. So we can instead directly create a simple RC circuit to approximate the propagation delay through the repeater and wire. The RC model is shown below using a simple π model for the interconnect.



Now we use the Elmore delay to calculate the delay through the first inverter and through the second inverter driving the interconnect.

$$\begin{aligned} t_{pd,seg} &= 27.67RC + (0.12R)(124.67C) + (0.12R + 0.25R)(103C) \\ &= 27.67RC + 14.96RC + 38.11RC = 80.75RC \\ &= (80.75RC \times \tau)/(3RC) = 26.92\tau \end{aligned}$$

The cycle time for the entire channel is simply the propagation delay out of the input register, the delay through each segment, the delay through the final minimum sized inverter, and the setup time of the output register.

$$\begin{aligned} t_{pd,channel} &= t_{pcq} + t_{pd,seg} + t_{pd,inv} + t_{setup} \\ &= (8 + (26.92 \times n) + 2 + 10)\tau = (26.92 \times n + 20)\tau \end{aligned}$$

For $t_{pd,channel}$ = 2 ns the length of the channel is as follows:

$$\begin{aligned} t_{pd,channel} &= (26.92 \times n + 20)\tau \\ 270\tau &= (26.92 \times n + 20)\tau \\ 250 &= 26.92 \times n \\ 9.28 &= n \end{aligned}$$

We round n down to 9 which means the channel length is 4.5 mm. So in other words, if we assume a 500 MHz clock then we can go 4.5 mm in a single cycle.

Part 3.B Channel Energy

Derive the worst-case energy in units of Joules for a single phit to traverse a channel as a function of n and b. A phit is b-bits of data going from the input to the output of the channel. Assume an activity factor of one. **Calculate the energy for the following two scenarios: $n = 4, b = 64$ and $n = 8, b = 128$. You must show your work.**

To determine the energy for a single phit to traverse the channel, we first need to calculate the total switched capacitance for a single segment.

$$\begin{aligned} C_{sw,seg} &= C_{repeater} + C_{wseg} \\ &= (3C + 3C) + (24.67C + 24.67C) + 200C = 255.34C \end{aligned}$$

Notice that we factor in both the gate and parasitic capacitance for the inverters. Now we can calculate the total switched capacitance for one bit of the channel.

$$\begin{aligned} C_{sw,bit} &= C_{sw,ffr} + (n \times C_{sw,seg}) + C_{sw,inv} + C_{sw,ffw} \\ &= 25C + (n \times 255.34C) + 6C + 25C \\ &= n \times 255.34C + 56C \end{aligned}$$

Now we can determine the energy per bit and the energy per phit for the channel as a function of b and n.

$$\begin{aligned} E_{bit} &= (1/2) \times C_{sw,bit} \times V_{dd}^2 \\ &= (1/2) \times (n \times 255.34 + 56) \times 0.5 \times 1^2 \\ &= n \times 63.84 + 14 \\ E_{phit} &= b \times E_{bit,seg} \\ &= b \times n \times 63.84 + 14 \end{aligned}$$

For $n = 4, b = 64$, the energy is as follows:

$$\begin{aligned} E_{phit} &= b \times E_{bit,seg} \\ &= 64 \times 4 \times 63.84 + 14 = 16 \text{ pJ} \end{aligned}$$

For $n = 8, b = 128$, the energy is as follows:

$$\begin{aligned} E_{phit} &= b \times E_{bit,seg} \\ &= 128 \times 8 \times 63.84 + 14 = 65 \text{ pJ} \end{aligned}$$

Part 3.C Channel Area

Derive the area for the channel in units of square micron as a function of b and n. Assume that the area of the channel is dominated by the channel wiring, so you can ignore the area of the pipeline registers and repeaters (i.e., the repeaters are hidden between the channel wiring). **Calculate the area for $n = 4$ and $b = 64$. You must show your work.**

Since we are ignoring the area of the pipeline registers and repeaters, the area of the channel is quite straight-forward. It is just the length of the channel times the number of wires in the channel times the wire pitch.

$$A_{channel} = (n \times 500) \times (b \times S_w)$$

For $n = 1, b = 64$, the area is as follows:

$$\begin{aligned} A_{channel} &= (n \times 500) \times (b \times S_w) \\ &= (4 \times 500) \times (64 \times 0.32) \\ &= 2,000 \times 20.48 \\ &= 40,960 \, \mu\text{m}^2 \end{aligned}$$

Problem 4. Final Mesh Network Analysis

In Problem 1, we were able to apply what we learned in ECE 4750 to quantitatively analyze the ideal terminal throughput and zero-load latency in cycles. However, we had to make several assumptions about the cycle time in order to analyze the latency (e.g., an eight-port router requires two cycles for switch traversal). We were only able to qualitatively compare and contrast the energy and area of both mesh topologies. In this problem, you will apply your results from Problems 2 and 3 to revisit our initial assumptions when calculating the zero-load latency and quantitatively estimate the energy and area.

Part 4.A Mesh Network Zero-Load Latency

We begin by revisiting the assumptions we made about the cycle-time of each mesh network topology in order to more accurately compare the zero-load latency of the two topologies. Remember that the processors and caches can run at 500 MHz, and that we will pipeline the routers and the channels so that they do not create a critical path and limit the cycle time. In Problem 1, we made the following assumptions. Five-port routers have a two-cycle latency: one cycle for route computation and switch arbitration, and one cycle for switch traversal. Eight-port routers have a three-cycle latency: one cycle for route computation and switch arbitration, and two cycles for switch traversal due to the larger number of ports. In both topologies it takes one cycle for a phit to traverse the edge of one tile (i.e., 2 mm). **Revisit each of these assumptions using your results from Problems 2 and 3.** Assume we have done circuit-level analysis to estimate that the route computation and switch arbitration logic in each router, and this logic is estimated to take approximately 750 ps regardless of the number of input ports. You can pack route computation, switch arbitration, and switch traversal into a single cycle as long as all three steps fit within the target cycle time for 500 MHz, otherwise you will need to distribute these steps over two or more cycles. Assume that the minimum latency of the router is one full cycle, and the minimum latency of any channel is also one full cycle.

Recalculate the average zero-load latency in cycles for both mesh topologies assuming uniform random traffic. Remember that we are completely ignoring the channels from the input/output terminals to the first/last routers. **Qualitatively explain how your new results differ from your initial estimates in Problem 1.**

We first revisit the router latency assumptions from Problem 1a. Recall that we assumed the *mesh8x8* router required two cycles and the *cmesh4x4* router required three cycles, which seemed reasonable given the fact that the *cmesh4x4* router is larger and more complex. For each topology, we need to determine the total time for route computation, switch arbitration, and crossbar traversal. The time for route computation and switch arbitration is given as 750 ps. The total time divided by 2 ns is roughly the router latency in cycles.

The *mesh8x8* topology uses five-port crossbars with a 64-bit phit size. In Problem 2a, we found that the traversal time for such a crossbar was 426 ps, so the total time for route computation, switch arbitration, and crossbar traversal is $750 + 426 = 1,176$ ps. Since this is less than 2 ns, the router latency for the *mesh8x8* topology should be a single cycle which is less than the two-cycle latency assumed in Problem 1.

The *cmesh4x4* topology uses eight-port crossbars with a 128-bit phit size. In Problem 2a, we found that the traversal time for such a crossbar was 736 ps, so the total time for route computation, switch arbitration, and crossbar traversal is $750 + 736 = 1,486$ ps. Since this is less than 2 ns, the router latency for the *cmesh4x4* topology should be a single cycle which is less than the three-cycle latency assumed in Problem 1.

We now revisit the channel latency assumptions from Problem 1. Recall that we assumed the shorter *mesh8x8* channels only required a single cycle and that the longer *cmesh4x4* channels required two cycles, which seemed reasonable since the *cmesh4x4* channels are twice as long.

In Problem 3a, we found that it should be possible to send a signal 4.5 mm in a single cycle (assuming a 2 ns cycle time). Each tile is 2×2 mm, so the shorter *mesh8x8* channels are 2 mm and the longer *cmesh4x4* channels are 4 mm. In other words, the channel latency for both topologies should be a single cycle.

So it should be clear that our preliminary assumptions about the router and channel latency in Problem 1a were overly conservative. We can easily complete route computation, switch arbitration, and crossbar traversal in a single cycle for both router designs, and we can easily

traverse all channels in both topologies in a single cycle. The updated zero-load latency for the *mesh8x8* topology is:

$$t_0 = H_r \times t_r + H_c \times t_c + (L/b) = 6.25 \times 1 + 5.25 \times 1 + 128/64 = 13.5 \text{ cycles}$$

The updated zero-load latency for the *cmesh4x4* topology is:

$$t_0 = H_r \times t_r + H_c \times t_c + (L/b) = 3.5 \times 1 + 2.5 \times 1 + 128/128 = 7 \text{ cycles}$$

With our preliminary assumptions, the zero-load latency of the *cmesh4x4* topology was slightly lower than the zero-load latency of the *mesh8x8* topology, but with our more realistic assumptions the zero-load latency of the *cmesh4x4* topology is significantly better (almost a factor of two) than the *mesh8x8* topology. The *cmesh4x4* topology has the same router/channel latencies, yet requires fewer router/channel hops.

Part 4.B Mesh Network Energy

Calculate the worst-case energy for a single packet to travel from the tile in the upper-left corner to the lower-left corner for both topologies. Note that this is for a full packet, not a phit. So since the *mesh8x8* topology requires two phits per packet you will need to carefully take this into account.

For the worst-case energy for a single packet to travel from the tile in the upper-left corner to the lower-left corner for both topologies, we simply sum the energies from traversing the channels and the routers until the packet reaches its destination. The formula for the total energy is thus:

$$E_{tot} = (L/b) \times (N_r \times E_r + N_c \times E_c)$$

where N_r is the number of routers on the path, E_r is the energy per router, N_c is the number of channels on the path, and E_c is the energy per phit for traversing a channel. Note that $N_c = N_r - 1$. The L/b term accounts for the fact that the router and channel energies were calculated in terms of energy per phit. A packet in the *mesh8x8* topology requires two phits, while a packet in the *cmesh4x4* topology only requires a single phit. It is important to ensure when we are comparing energy we are comparing how much work is required to complete the same task (i.e., sending a packet of the same length).

For the *mesh8x8* topology, N_r is 8. From Problem 2b, we know the crossbar traversal energy is 3.7 pJ/phit. Since the channels in the *mesh8x8* topology are 2 mm long, they will require four segments, and from Problem 3b, we know that such a channel will require 16 pJ/phit. So we can now calculate the total energy to send a packet from the upper-left corner to the lower-left corner.

$$E_{tot} = (L/b) \times (N_r \times E_r + N_c \times E_c) = (128/64) \times (8 \times 3.7 + 7 \times 16) = 283 \text{ pJ}$$

Of course this does not account for the extra energy required to do route computation, switch arbitration, and buffering in the router, but it is a good rough approximation.

For the *cmesh4x4* topology, N_r is 4. From Problem 2b, we know the crossbar traversal energy is 14.8 pJ/phit. Since the channels in the *cmesh4x4* topology are 4 mm long, they will require eight segments, and from Problem 3b, we know that such a channel will require 65 pJ/phit. So we can now calculate the total energy to send a packet from the upper-left corner to the lower-left corner.

$$E_{tot} = (L/b) \times (N_r \times E_r + N_c \times E_c) = (128/128) \times (4 \times 14.8 + 3 \times 65) = 254 \text{ pJ}$$

From this analysis, we can see that even though the energy per phit for a single router or channel is higher in the *cmesh4x4* topology, once we consider a full packet and the reduced number of router/channel hops, the *cmesh4x4* topology requires similar or even less energy than the *mesh8x8* topology.

Part 4.C Mesh Network Area

Calculate the area for the on-chip network for both topologies. Assume a floorplan as shown in the figure in Problem 1 and assume the router area is dominated by the crossbar wiring. *Assume that the channels can be routed over the tiles, and so the channel area does not directly impact the overall network area!*

There are 64 routers in the *mesh8x8* topology. From Problem 2c, we know that each of these routers requires 10,486 μm^2 for a total area of 671,104 μm^2 or 0.67 mm^2. There are 16 routers in the *cmesh4x4* topology. From Problem 2c, we know that each of these routers requires 107,374 μm^2 for a total area of 1,717,984 μm^2 or 1.7 mm^2.

These results indicate that the cmesh4x4 topology requires 2.5$\times$ more area than the mesh8x8 topology. However, it is important that we also keep this in the context of the entire chip. The total chip area is 256 mm^2, so both topologies require less than 1% of the entire chip area.

Part 4.D Mesh Network Comparison

Given the analysis in the previous parts, make a compelling quantitative and qualitative argument for which of these two networks will be better in terms of zero-load latency, energy, and area. If you would like, you can also revisit any of the assumptions we have made in this and the previous problems which you find questionable, and you can argue how changing these assumptions might impact your conclusions.

Our analysis in Problem 1 was relatively inconclusive, but it should now be clear that the *cmesh4x4* topology can offer compelling benefits compared to the *mesh8x8* topology. The *cmesh4x4* topology has similar ideal terminal throughput, but significantly lower zero-load latency. Even though this analysis was for uniform random traffic, keep in mind that other traffic patterns will likely also see significant benefit from the *cmesh4x4* topology owing to its lower hop count and serialization latency. In addition, the *cmesh4x4* topology will likely have similar or better energy efficiency. The *cmesh4x4* topology requires more area than the *mesh8x8* topology, but also note that absolute area of either network is a tiny fraction of the entire chip. So these results suggest the *cmesh4x4* topology could potentially achieve better performance compared to the *mesh8x8* topology at similar energy with no real impact on the overall chip area.

Note that these first-order results nicely match recent research on high-radix, low-diameter on-chip networks. There has been significant interest in moving away from low-radix, high-diameter topologies such *mesh8x8*, and instead using concentrated topologies, flattened butterfly topologies, or even global crossbar topologies.

Appendix A: Technology Parameters for 90 nm Process

Parameter	Value	Description
τ	7.4 ps	Delay unit ($3RC$)
FO4 Delay	37 ps	Fan-out-of-four delay (5τ)
V_{dd}	1 V	Nominal supply voltage
μ_n/μ_p	2	Ratio of NMOS mobility to PMOS mobility
C	0.5 fF	Gate capacitance for NMOS in cannonical minimum sized inverter
R	4.9 kΩ	Effective resistance for NMOS in cannonical minimum sized inverter
S_w	0.32 µm	Minimum wire pitch for metal layers 1–4
C_w	$0.4\,C/\mu$m	Wire capacitance for metal layers 1–4 per micron
R_w	$0.5\,R/$mm	Effective resistance for metal layers 1–4 per millimeter
$C_{in,ff}$	$3C$	Gate capacitance for data input of flip-flop
$C_{sw,ffr}$	$25C$	Switched capacitance for reading a flip-flop
$C_{sw,ffw}$	$25C$	Switched capacitance for writing a flip-flop
t_{pcq}	$8\,\tau$	Clock-to-q delay for flip-flop assuming driving minimum sized inverter
t_{setup}	$10\,\tau$	Setup time for flip-flop