A Detailed Architectural-Level Power Model for Router Buffers, Crossbars and Arbiters

Hangsheng Wang
Department of Electrical Engineering
Princeton University

January 31, 2004

1 Modeling methodology

This section explains the main methodology of our power modeling. Currently, we only model switching energy, $E = \frac{1}{2}\alpha C V_{dd}^2$, where α is the switching activity, C is the switching capacitance and V_{dd} is the supply voltage.

To model a function unit, we divide this function unit into several *atomic* components, whose switching activities and capacitance can be independently calculated. A component is *atomic* if it only consists of one capacitor, or multiple capacitors that have identical switching behavior. This property of atomic components ensures that an atomic component can be represented by one capacitance. Usually an atomic component consists of several physically connected capacitors. For example, a wordline is physically connected to the gate ends of pass transistors connecting the wordline and memory cells, so the wordline and the gate ends of pass transistors belong to one atomic component. There are cases that an atomic component consists of capacitors that are not physically connected. For example, the gate ends of the NMOS and PMOS transistors of an inverter and their drain ends belong to one atomic component because they always switch together.

The model of this function unit is thus developed as follows:

1. Divide the function unit into atomic components which are disjoint to each other and cover the whole function unit.
2. Derive capacitance equations for each atomic component.
3. Identify possible operations (functions) of the function unit, e.g. read, write, update, and their arguments. These operations and their arguments are supposed to be accurately reported by a simulator.
4. Derive equations that compute switching activities (thus energy) of each atomic component from operation arguments.

2 Capacitance notation

Capacitance computations comprise a large amount of the modeling effort. The capacitance notation used throughout this report is listed in Table 1.

Table 1: Capacitance notation

$c_g(w, l)$	transistor gate end capacitance. w is the gate width, l is the length of attached poly wire if any. l is usually 0 due to the prevailing multi-layer metal wires. So it can be abbreviated as $c_g(w)$.
$c_d(w, t, s)$	transistor drain end capacitance. w is the gate width, t indicates transistor type: N (NMOS) or P (PMOS), s is the number of stacking transistors.
$w(T, t)$	transistor gate width of gate T. t indicates transistor type.
$C_g(T)$	gate end capacitance of gate T. For an n-input logic gate, $C_g(T)$ denotes the gate end capacitance of one input, i.e. $C_g(T) = c_g(w(T, N)) + c_g(w(T, P))$. For a transmission gate, $C_g(T)$ denotes the gate end capacitance of the control signal(s), i.e. $C_g(T) = c_g(w(T, N)) + c_g(w(T, P))$ if T consists of both an NMOS transistor and a PMOS transistor, or $C_g(T) = c_g(w(T, N))$ if T consists of only an NMOS transistor.
$C_d(T)$	drain end capacitance of gate T. For an n-input NOR gate, $C_d(T) = n \cdot c_d(w(T, N), N, 1) + c_d(w(T, P), P, n)$, for an n-input NAND gate, $C_d(T) = c_d(w(T, N), N, n) + n \cdot c_d(w(T, P), P, 1)$. For a transmission gate, $C_d(T) = c_d(w(T, N), N, 1) + c_d(w(T, P), P, 1)$ if T consists of both an NMOS transistor and a PMOS transistor, or $C_d(T) = c_d(w(T, N), N, 1)$ if T consists of only an NMOS transistor. A transismission gate has 2 drain ends, or one drain and one source, $C_d(T)$ counts one of them.
$C_a(T)$	only applicable if T is an inverter. $C_a(T) = C_g(T) + C_d(T)$, which is essentially all capacitance of an inverter.
$C_{w1}(L)$	minimal spacing metal wire capacitance. L is the wire length. Minimal metal wire spacing, including wire width, is 5λ.
$C_{w2}(L)$	twice minimal spacing metal wire capacitance. L is the wire length.
$C_{w3}(L)$	thrice minimal spacing metal wire capacitance. L is the wire length.
$C_{w0}(L)$	any other metal wire capacitance. L is the wire length.

These capacitances are implemented as functions, except that $w(T, t)$ may be specified by parameters. We'll skip the details of their implementations for now and use them to elaborate our models, then go back to them at the end of this report.

C_{w0} is essentially infinite spacing metal wire capacitance, i.e. no coupling capacitance. We should have used different capacitances for different layers of

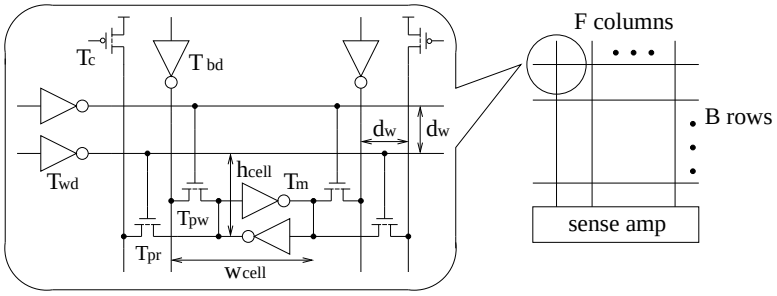

Figure 1: Structure of an input buffer with one read port and one write port. T_c is the pre-charging transistor, T_{wd} is the wordline driver, T_{bd} is the bitline write driver, T_m is the memory cell inverter, T_{pr} is the pass transistor connecting read ports to memory cells, T_{pw} is the pass transistor connecting write ports to memory cells. The lower wordline is a read wordline while the higher one is a write wordline. The outer bitlines are read bitlines while the inner ones are write bitlines.

metal wires. For now we simply assume they are all the same.

3 Power model of input buffers

Figure 1 shows the structure of an input buffer with one read port and one write port. We assume that an input buffer is organized as a FIFO, so we do not consider decoders and output drivers for it. Table 2 lists its architectural parameters.

Table 2: Input buffer model architectural parameters

B	buffer size in flits, which is also the number of rows
F	flit size in bits, which is also the number of columns
P_r	number of buffer read ports
P_w	number of buffer write ports

3.1 Atomic components and their capacitances

An input buffer is divided into 7 atomic components: read wordline, write wordline, read bitline, write bitline, memory cell, pre-charging circuitry and sense amplifier.

Read wordline A read wordline has 3 capacitance components:

1. Wire. Its length is F times the memory cell width. Memory cell width is the memory cell core width w_{cell} plus spacing of all bitlines. Bitline

spacing is d_w and there are $2(P_r + P_w)$ bitlines per column. So the total wire length is $F \cdot (w_{cell} + 2d_w(P_r + P_w))$. Wordline spacing is also d_w, which is 15λ, so the wire capacitance is $C_{w3}(F \cdot (w_{cell} + 2d_w(P_r + P_w)))$.

2. Gate ends of pass transistors connecting wordlines and memory cells. There are 2 pass transistors per column. So their total capacitance is $2F \cdot C_g(T_{pr})$.

3. Wordline driver T_{wd}. Its capacitance is $C_a(T_{wd})$. The transistor size of T_{wd} is computed according to its load capacitance, i.e. the sum the two items above, and assuming the wordline rising time is $\frac{1}{16}$ of one clock cycle. How to perform this automic transistor sizing is explained at the end of this report.

So the read wordline capacitance is:

$$C_{wr} = C_{w3}(F \cdot (w_{cell} + 2d_w(P_r + P_w))) + 2F \cdot C_g(T_{pr}) + C_a(T_{wd})$$

Write wordline A write wordline is almost identical with a read wordline except for the pass transistors. Pass transistors connecting read/write ports to memory cells have different sizes. So the write wordline capacitance is:

$$C_{ww} = C_{w3}(F \cdot (w_{cell} + 2d_w(P_r + P_w))) + 2F \cdot C_g(T_{pw}) + C_a(T_{wd})$$

Read bitline A read bitline has 3 capacitance components:

1. Wire. Its length is B times the memory cell height. Memory cell height is the memory cell core height h_{cell} plus spacing of all wordlines. Wordline spacing is d_w and there are $P_r + P_w$ wordlines per row. So the total wire length is $B \cdot (h_{cell} + d_w(P_r + P_w))$. Bitline spacing d_w is 15λ, so the wire capacitance is $C_{w3}(B \cdot (h_{cell} + d_w(P_r + P_w))$.

2. Drain ends of pass transistors. One read bitline connects to one pass transistor per row. So their total capacitance is $B \cdot C_d(T_{pr})$.

3. Drain end of the pre-charging transistor. The pre-charging transistor can be regarded as a single PMOS transistor transmission gate and its drain capacitance is $C_d(T_c)$.

So the read bitline capacitance is:

$$C_{br} = C_{w3}(B \cdot (h_{cell} + d_w(P_r + P_w))) + B \cdot C_d(T_{pr}) + C_d(T_c)$$

Write bitline A write bitline has 3 capacitance components:

1. Wire. Same as read bitline.

2. Drain ends of pass transistors. Their total capacitance is $B \cdot C_d(T_{pw})$.

3. Write driver T_{bd}. Its capacitance is $C_a(T_{bd})$. Transistor size of T_{bd} is computed according to its load capacitance and assuming a $\frac{1}{8}$ cycle rising/falling time.

So the write bitline capacitance is:

$$C_{bw} = C_{w3}(B \cdot (h_{cell} + d_w(P_r + P_w))) + B \cdot C_d(T_{pw}) + C_a(T_{bd})$$

Memory cell A memory cell has 2 capacitance components:

1. Inverters. There are 2 inverters per memory cell and their total capacitance is $2C_a(T_m)$.
2. Drain ends of pass transistors. There are $2P_r$ read pass transistors and $2P_w$ write pass transistors per memory cell. The total capacitance is $2(P_r \cdot C_d(T_{pr}) + P_w \cdot C_d(T_{pw}))$.

So the memory cell capacitance is:

$$C_{cell} = 2C_a(T_m) + 2(P_r \cdot C_d(T_{pr}) + P_w \cdot C_d(T_{pw}))$$

Pre-charging circuitry The pre-charging circuitry is just the pre-charging transistors, we only count gate end capacitance here because its drain end capacitance is already included in the read bitlines. Actual transistor size is computed based on its load capacitance, i.e. read bitline capacitance C_{br}, and assuming a $\frac{1}{8}$ cycle rising time. Because C_{br} has $C_d(T_c)$ as its component, we have a circular dependency here. We break the circular dependency as follows: first we compute C_{br} without $C_d(T_c)$, then use this C_{br} to compute the transistor size of T_c, then add $C_d(T_c)$ to C_{br}. Experiments show that this approximation introduces negligible error.

Again we regard the pre-charging transistor as a single PMOS transistor transmission gate, so its gate end capacitance is:

$$C_{chg} = C_g(T_c)$$

Sense amplifier The sense amplifier is an analog circuit so we don't derive its capacitance equation but use an empirical energy model instead.

3.2 Operation switching activity and energy dissipation

We consider two operations for an input buffer: *read* and *write*.

Before going into the computation details, we need to distinguish two types of switching used in this report. We call a rising/falling pair a full switch, and one rising or one falling a single switch. Full switch is convenient to count the activity of some atomic component which will reset to a fixed state after each operation, we will see wordline as a good example of this type. A single switch is used to count the activity of some component which will maintain

its last state and whose activity is purely determined by Hamming distance between pervious and current states.

We define basic switching energy E_x of an atomic component x as the energy dissipated by one switch of this component. We use $E_x^{''}$ if this atomic component uses full switch to count activity, or $E_x^{'}$ if single switch is used.

$$\begin{aligned} E_x^{''} &= C_x V_{dd}^2 \\ E_x^{'} &= \frac{1}{2} C_x V_{dd}^2 \end{aligned}$$

C_x is the atomic component capacitance. This definition is to clearly separate the effort of capacitance derivation and couting switching activity. It also eases implemention. For any atomic component whose capacitance equation has been derived, e.g. C_{wr}, we'll use its basic switching energy, e.g. $E_{wr}^{''}$ or $E_{wr}^{'}$, without further explanation.

We use $H(x, y)$ to denote the Hamming distance between x and y.

3.2.1 Read operation

A read operation takes no argument. Switching activity and energy dissipation of each atomic component is computed as follows:

- Read wordline. One read wordline rises to 1 during the reading process and then falls to 0 after the read is finished. This counts as one full switch. So its energy dissipation is:

  $$E_{read\_wordline} = E_{wr}^{''}$$

- Write wordline. No activity.

- Read bitline. No matter what value is read out, due to the pre-charging scheme, one of the two read bitlines per column falls to 0 and then rises to 1 during the next pre-charging. So among the total $2F$ read bitlines which belong to one read port, F read lines have a full switch. Read bitline energy dissipation is:

  $$E_{read\_bitline} = F \cdot E_{br}^{''}$$

  Read bitlines usually have low voltage swing. We assume $\frac{1}{2} V_{dd}$ here, so $E_{br}^{''}$ is $C_{br} V_{dd} (\frac{1}{2} V_{dd})$ rather than $C_{br} V_{dd}^2$.

- Write bitline. No activity.

- Memory cell. No activity.

- Pre-charging circuitry. The pre-charging signal rises to 1 before the reading process and then falls to 0 after the read is finished. So each pre-charging transistor gate end has a full switch and one read port has $2F$ pre-charging transistors. Pre-charging energy dissipation is:

$$E_{pre\_charg} = 2F \cdot E''_{chg}$$

- Sense amplifier. E_{sense_amp}.

So the energy dissipation of one read operation is:

$$E_{read} = E_{read\_wordline} + E_{read\_bitline} + E_{pre\_charg} + E_{sense\_amp}$$

3.2.2 Write operation

A write operation has 2 arguments: p, the write port, and f, the flit which is written. Switching activity and energy dissipation of each atomic component is computed as follows:

- Read wordline. No activity.
- Write wordline. Similar to the read wordline in a read operation, one write wordline has a full switch. So its energy dissipation is:

$$E_{write\_wordline} = E''_{ww}$$

- Read bitline. No activity.
- Write bitline. We assume that wordlines maintain the last written flit until the next write operation (presumably by write latches). So the number of switching columns is the Hamming distance between the last flit f_b and the current flit f. In each column, both bitlines have a single switch, one from 1 to 0 and another from 0 to 1, which is equivalent to one bitline having a full switch. So the energy dissipation is:

$$E_{write\_bitline} = H(f, f_b)E''_{bw}$$

- Memory cell. The number of switching memory cells is the Hamming distance between the previously stored flit f_m and the current flit f. A switching memory cell has a single switch, although a half from 1 to 0 while another half from 0 to 1. So the energy dissipation is:

$$E_{memory\_cell} = H(f, f_m)E'_{cell}$$

- Pre-charging circuitry. No activity.
- Sense amplifier. No activity.

So the energy dissipation of one write operation is:

$$E_{write} = E_{write\_wordline} + E_{write\_bitline} + E_{memory\_cell}$$

It is clear that the simulator needs to maintain state variables f_b and f_m for each write port in order to compute write energy.

4 Transistor size and other technology parameters

Table 3 lists all transistor size and other technology parameters used in the input bufer model.

Table 3: Transistor size and technology parameters of the input buffer model

Gate	$w(T,N)$	$w(T,P)$
T_m	12λ	6λ
T_{pr}	10λ	N/A
T_{pw}	5λ	N/A

Parameter	Value
d_w	15λ
h_{cell}	40λ
w_{cell}	20λ

Sizes of T_c, T_{wd} and T_{bd} are not listed since they are computed based on load capacitance.

5 Modeling assumptions

This section lists the modeling assumptions whose correctness we are uncertain of.

- The wordline length $F \cdot (w_{cell} + 2d_w(P_r + P_w))$ and the bitline length $B \cdot (h_{cell} + d_w(P_r + P_w))$ probably mean that we do not consider the overlaping between transistors and metal wires. These equations are taken from Wattch [**?**] and seem to be correct.
- We assume there is one physical buffer per input port. If multiple virtual channels exist, they share the physical buffer by splitting the buffer into several consecutive ranges. This is a simplified assumption and implies static buffer space allocation. If we want dynamic allocation to increase buffer utilization, its link-list indexing will introduce a tag array.

6 Power model of crossbars

We model two types of crossbars: matrix crossbar and multiplexor-tree crossbar. Figure 2 and Figure 3 show their structures respectively.

6.1 Power model of crossbar connectors

A cross-point in the crossbar is called a connector, which is a common subcomponent of both types of crossbars. We model 2 connector implementations: transmission gate and tri-state buffer.

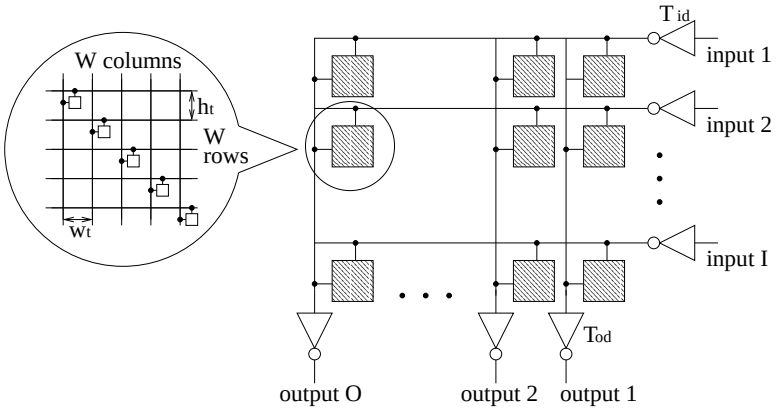

Figure 2: Structure of a matrix crossbar. T_{id} is the input driver, T_{od} is the output driver, the small squares are connectors.

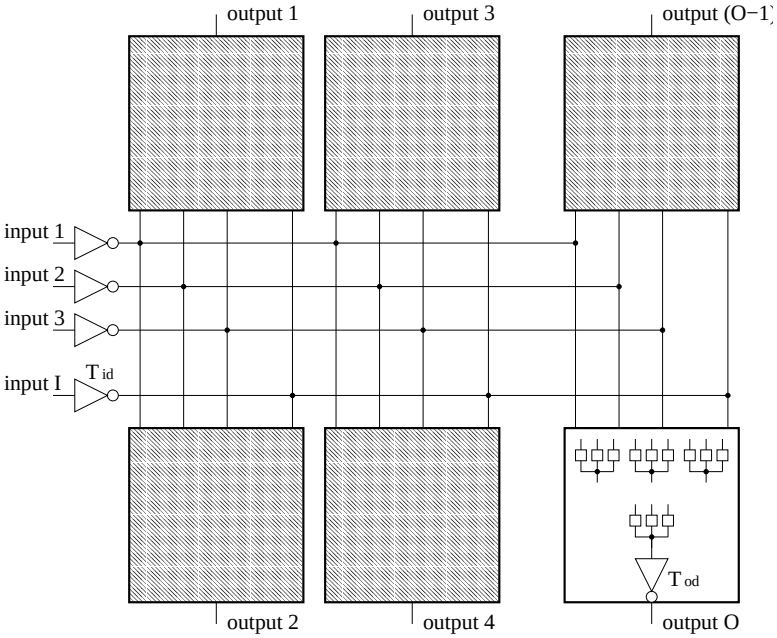

Figure 3: Structure of a multiplexor-tree crossbar. T_{id} is the input driver, T_{od} is the output driver, the small squares are connectors. The unshaded box shows the structure of one tree with degree 3, each box has W such trees, where W is the crossbar port width in bits.

6.1.1 Transmission gate connector

A transmission gate is divided into 3 atomic components: input end, output end and control end. There is one parameter, *trn_type*, to indicate whether

the transmission gate consists of only one NMOS transistor or both one NMOS transistor and one PMOS transistor.

Input end The input end has the drain capacitance of the transmission gate T.

$$C_{in} = C_d(T)$$

Output end We assume that a transmission gate is symmetric, so its output end has the same capacitance as the input end.

$$C_{out} = C_d(T)$$

Control end The control end has the gate capacitance of the transmission gate T. Although there are 2 complemented control signals, we count them as one atomic component since they are guaranteed to switch together.

$$C_{ctr} = C_g(T)$$

Transistor size Table 4 lists all transistor sizes used in the transmission gate connector model.

Table 4: Transistor size of transmission gate connector model

Gate	$w(T, N)$	$w(T, P)$
T	10λ	20λ

We do not resize transistors because we believe that large scale (high load capacitance) crossbars should use tri-state buffer connectors.

6.1.2 Tri-state buffer connector

Figure 4 illustrates the circuit structure of a tri-state buffer.

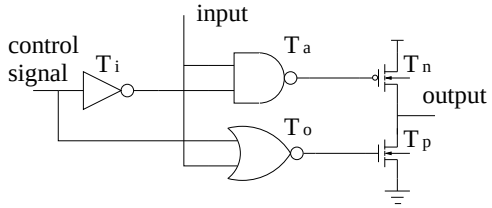

Figure 4: Structure of a tri-state buffer

It has the same atomic component division as a transmission gate: input end, output end and control end.

Input end The input end connects to the gate end of both the NAND gate and the NOR gate.

$$C_{in} = C_g(T_a) + C_g(T_o)$$

Output end The output end connects to the drain end of the output driver.

$$C_{out} = C_d(T_n) + C_d(T_p)$$

Control end The control end connects to the gate end of both the NAND gate and the NOR gate.

$$C_{ctr} = C_g(T_a) + C_g(T_o)$$

Since each time the control signals switch, one of the two internal nodes A and B will switch. To avoid maintaining states for each tri-state buffer, we assume A and B switches equally likely and we put this capacitance into C_{ctr}.

$$\begin{aligned} C_A &= C_d(T_a) + C_g(T_p) \\ C_B &= C_d(T_o) + C_g(T_n) \\ C_{ctr} &= C_g(T_a) + C_g(T_o) + \tfrac{1}{2}(C_A + C_B) \end{aligned}$$

Transistor size Table 5 lists all transistor sizes used in the tri-state buffer connector model.

Table 5: Transistor size of tri-state buffer connector model

Gate	$w(T, N)$	$w(T, P)$
T_a	60λ	25λ
T_o	15λ	100λ

When a tri-state buffer is used in crossbars, the sizes of the output driver transistor T_n and T_p are computed based on load capacitance given by the crossbar model and assuming a $\frac{1}{3}$ cycle rising/falling time.

The default values listed here are taken from Cacti **??** and may not be appropriate for a crossbar because Cacti is a cache delay model. Actually they conflict with the assumption below that track width and track height in a matrix crossbar are both 15λ. But they are the only real numbers we have, so we use the transmission gate connector in our paper.

6.1.3 Interface with crossbar model

From the derivation above, we see that a connector can be abstracted by 3 capacitances: C_{in}, C_{out} and C_{ctr}, no matter what the actual implementation is. We use $C_{in}(\text{CNT})$, $C_{out}(\text{CNT})$ and $C_{ctr}(\text{CNT})$ to denote them in the derivation of our crossbar model. On the other side, the crossbar model tells the connector model its output line capacitance C_{xb_out} as the load capacitance to determine transistor sizes.

6.2 Power model of matrix crossbars

Table 6 lists the architectural parameters of the matrix crossbar power model.

Table 6: Matrix crossbar model architectural parameters

I	number of crossbar input ports
O	number of crossbar output ports
W	crossbar port width in bits
cnt_type	connector type, either transmission gate or tri-state buffer
trn_type	used by transmission gate connector model

6.2.1 Atomic components and their capacitances

Besides its sub-component, connector, which is modeled separately, a matrix crossbar is divided into 3 atomic components: input line, output line and control line.

Input line An input line has 3 capacitance components:

1. Wire. Its length is the total width across all output lines. There are O output ports and each is W-bit wide, so the wire length is $O \cdot W \cdot w_t$, where w_t is the track width. Input line spacing is the track height h_t, which is 15λ, so the wire capacitance is $C_{w3}(O \cdot W \cdot w_t)$.
2. Input capacitance of connectors. Each input line connects to O output lines (or $O - 1$ if no U-turn), so the total capacitance is $O \cdot C_{in}(\text{CNT})$.
3. Input driver T_{id}. Its capacitance is $C_a(T_{id})$ and its transistor size is computed based on its load capacitance, i.e. the sum of the two items above, and assuming a $\frac{1}{3}$ cycle rising/falling time.

So the input line capacitance is:

$$C_{xb\_in} = C_{w3}(O \cdot W \cdot w_t) + O \cdot C_{in}(\text{CNT}) + C_a(T_{id})$$

Output line An output line has 3 capacitance components:

1. Wire. Its length is the total height across all input lines. There are I input ports and each is W-bit wide, so the wire length is $I \cdot W \cdot h_t$. Output line spacing w_t is 15λ, so the wire capacitance is $C_{w3}(I \cdot W \cdot h_t)$.
2. Output capacitance of connectors. Each output line connects to I input lines (or $I - 1$ if no U-turn), so the total capacitance is $I \cdot C_{out}(\text{CNT})$.
3. Output driver T_{od}. Its capacitance is $C_a(T_{od})$.

So the output line capacitance is:

$$C_{xb\_out} = C_{w3}(I \cdot W \cdot h_t) + I \cdot C_{out}(\text{CNT}) + C_a(T_{od})$$

Control line A control line has 3 capacitance components:

1. Wire. Control lines come from arbiters, so the position of the arbiters determines control line length. We use the Alpha-21364 router as our reference, which puts arbiters at the left and right side of the crossbar ("left" and "right" regarding to Figure 2). Control lines for different crosspoints have different length. To avoid maintaining states for each control line individually, we make a simplification by using average wire length, which is one half of the input line length.

   Control lines are not as dense as input lines. One control line drives the connectors of W input lines and W is usually larger than or equal to 32. So we can assume wires of infinite spacing for control lines. The wire capacitance is $C_{w0}(\frac{1}{2} \cdot O \cdot W \cdot w_t)$.

2. Control end capacitance of connectors. Each control line drives W connectors, so the total capacitance is $W \cdot C_{ctr}(\text{CNT})$.

3. Inverter T_i. This component is present only if the connector type needs complemented control signals, i.e. tri-state buffers and transmission gates with both NMOS and PMOS transistors. Define a boolean variable B_{inv} to be:

$$B_{inv} = \begin{cases} 1, \text{if complemented control signals needed} \\ 0, \text{otherwise} \end{cases}$$

   Then the inverter capacitance is $B_{inv} \cdot C_a(T_i)$.

So the control line capacitance is:

$$C_{xb\_ctr} = C_{w0}\left(\frac{1}{2} \cdot O \cdot W \cdot w_t\right) + W \cdot C_{ctr}(\text{CNT}) + B_{inv} \cdot C_a(T_i)$$

6.2.2 Transistor size and other technology parameters

Table 7 lists all transistor size and other technology parameters used in the matrix crossbar model.

Table 7: Transistor size and technology parameters of the matrix crossbar model

Gate	$w(T,N)$	$w(T,P)$
T_i	12.5λ	25λ
T_{od}	120λ	200λ

Parameter	Value
h_t	15λ
w_t	15λ

The size of the input driver T_{id} is not listed since it is computed based on its load capacitance. The size of the output driver T_{od} should also depend on its

load capacitance, which is usually the interface circuitry between the crossbar and the output link interface. Since that is the part for which we still lack good understanding, we use the default values listed in the table for the time being.

6.3 Power model of multiplexor-tree crossbars

Table 8 lists architectural parameters of the multiplexor-tree crossbar power model.

Table 8: Multiplexor-tree crossbar model architectural parameters

I	number of crossbar input ports
O	number of crossbar output ports
W	crossbar port width in bits
d	degree of multiplexing
cnt_type	connector type, either transmission gate or tri-state buffer
trn_type	used by transmission gate connector model

We can see that the multiplexor-tree crossbar model has one more parameter than the matrix crossbar model: d, which is the number of fan-in of each multiplexor. Not all muxes have the same degree, e.g. if $I = 11$ and $d = 4$, then 2 first-level muxes have degree 4, one first level mux and the last level mux have degree 3. This situation is handled by the model, but to simplify the illustration, we assume we always have $I = d^n$, where n is an positive integer, so that all muxes have degree d. We call n the depth of the multiplexor-tree, which is a derived parameter.

6.3.1 Atomic components and their capacitances

Besides its sub-component, connector, which is modeled separately, a multiplexor-tree crossbar is divided into 4 atomic components: input line, output line, control line and internal node. Input lines are inputs to all first level muxes, output lines are outputs of all last level muxes, and internal nodes are all other data path nodes between two levels of muxes.

Input line An input line has 3 capacitance components:

1. Wire. Each input line can be divided into horizontal segment and vertical segments. The horizontal segment crosses over the total width of all multiplexor trees. The width of one multiplexor tree is the number of input ports I times the port width W times the track width w_t. We assume that the multiplexor trees are placed on both sides of input lines and if O is odd, one multiplexor tree is placed at the right end of input lines. So on each side there are $\lfloor \frac{1}{2} O \rfloor$ multiplexor trees and the horizontal segment length is $\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot w_t$.

There are $\lfloor \frac{1}{2} O \rfloor$ vertical segments, one per pair of multiplexor trees. The height of each segment is the number of input ports I times the port width W times the track height h_t, so the total length of all vertical segments is $\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot h_t$.

Horizontal segment spacing is the track height h_t, which is the minimal pitch 5λ since along the vertical direction only wires are placed. Vertical segment spacing is the track width w_t, which is 15λ due to the spacing between connectors. So the total wire capacitance is $C_{w1}(\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot w_t) + C_{w3}(\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot h_t)$.

2. Input capacitance of connectors. Each input line connects to O output lines (or $O - 1$ if no U-turn), so the total capacitance is $O \cdot C_{in}(\text{CNT})$.

3. Input driver T_{id}. Its capacitance is $C_a(T_{id})$ and its transistor size is computed based on its load capacitance, i.e. the sum of the two items above, and assuming a $\frac{1}{3}$ cycle rising/falling time.

So the input line capacitance is:

$$C_{xb\_in} = C_{w1}\left(\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot w_t\right) + C_{w3}\left(\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot h_t\right) + O \cdot C_{in}(\text{CNT}) + C_a(T_{id})$$

Output line An output line has 2 capacitance components:

1. Output capacitance of connectors. Each output line is the multiplexed output of a d-input mux, so the total capacitance is $d \cdot C_{out}(\text{CNT})$.

2. Output driver T_{od}. Its capacitance is $C_a(T_{od})$.

So the output line capacitance is:

$$C_{xb\_out} = d \cdot C_{out}(\text{CNT}) + C_a(T_{od})$$

There is no wire component since an output line is simply the output of a mux. We still call it "output line" for terminology consistency with the matrix crossbar model.

Control line A control line has 5 capacitance components:

1. Wire. Only control lines of first level muxes have wires. We make the same assumptions about the placement and position of control lines as we do in deriving the matrix crossbar model. But here the control line length is one half of the length of the horizontal segment of an input line. Define a boolean variable B_{wire} to be:

$$B_{1st} = \begin{cases} 1, \text{if first level mux} \\ 0, \text{otherwise} \end{cases}$$

Then the wire capacitance is $B_{1st} \cdot C_{w0}(\frac{1}{2}\lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot w_t)$.

2. Control end capacitance of connectors. Each control line drives W connectors, so the total capacitance is $W \cdot C_{ctr}(\text{CNT})$.

3. Inverter T_i. This component is present if the connector type needs complemented control signals, i.e. tri-state buffer and transmission gate with both NMOS and PMOS transistors. Since the control signal of any non-first level mux is the ORed result of all control signals of its predecessor mux, as shown in Figure 3, an inverter is also needed for any non-first level control signal to negate the output of the NOR gate even if we use NMOS transistor only transmission gates. Define a boolean variable B_{inv} to be:

$$B_{inv} = \begin{cases} 1, \text{if an inverter is needed} \\ 0, \text{otherwise} \end{cases}$$

   Then the inverter capacitance is $B_{inv} \cdot C_a(T_i)$.

4. Drain end capacitance of a previous level NOR gate. This component is present for control lines of non-first level muxes and its capacitance is $\overline{B_{1st}} \cdot C_d(T_o)$

5. Gate end capacitance of a next level NOR gate. This component is present for control lines of non-last level muxes. Define a boolean variable B_{last} to be:

$$B_{last} = \begin{cases} 1, \text{if last level mux} \\ 0, \text{otherwise} \end{cases}$$

   Then the gate end capacitance is $\overline{B_{last}} \cdot C_g(T_o)$.

So the capacitance of an individual control line at mux level i is:

$$\begin{aligned} C_{xb\_ctr}(i) \quad = \quad & B_{1st}(i) \cdot C_{w0} \left(\frac{1}{2} \lfloor \frac{1}{2} O \rfloor \cdot I \cdot W \cdot w_t \right) + W \cdot C_{ctr}(\text{CNT}) + \\ & B_{inv}(i) \cdot C_a(T_i) + \overline{B_{1st}(i)} \cdot C_d(T_o) + \overline{B_{last}(i)} \cdot C_g(T_o) \end{aligned}$$

Compared with a matrix crossbar, a multiplexor-tree crossbar may have some intermediate control lines, which, rather than coming from arbiters, are locally derived using NOR gates and inverters. Their values are determined by the first level control lines. In order not to degrade our switching activity computation to RTL simulation, we make a simplified assumption that after each crossbar traversal, arbiters clear grant signals, which drive crossbar control lines, to 0 so as not to let unexpected data pass the crossbar. While this may not always be true, it does somehow mimic the glitches generated along the crossbar data path when two grant signals are turned on and off back to back. Under this assumption, all control lines are initially 0, when some grant signal is set to 1, it will propagate through the NOR gates and inverters from the first level mux to the last level mux, thus effectively building a path from one leaf to the root of the multiplexor tree so that each control line on the path is 1, all other control lines are 0 (in the same tree). The path is reset to 0 after the

crossbar traversal according to our assumption. Due to the mutual-exclusive nature of grant signals, only one path can exist in a tree at one time. So we can combine all control lines along the path as one compound control line, whose total capacitance has switched. The path may differ from arbitration to arbitration, but the switching capacitance is a constant:

$$C_{xb\_ctr} = \sum_{i=1}^{n} C_{xb\_ctr}(i)$$

where n is the depth of the tree. This is the equivalent control line capacitance corresponding to one output port, which is also the counterpart of C_{xb_ctr} in the matrix crossbar model.

6.3.2 Transistor size and other technology parameters

Table 9 lists all transistor size and other technology parameters used in the multiplexor-tree crossbar model.

Table 9: Transistor size and technology parameters of miltiplexor-tree crossbar model

Gate	$w(T,N)$	$w(T,P)$
T_i	12.5λ	25λ
T_o	13.5λ	76λ
T_{od}	120λ	200λ

Parameter	Value
h_t	5λ
w_t	15λ

The size of the input driver T_{id} is not listed since it is computed based on its load capacitance. Other default tranistor sizes may not be appropriate since they are taken from Cacti [**?**] rather than a real crossbar design.

6.4 Operation switching activity and energy dissipation

We consider one operation for crossbars: *flit_traversal*.

A flit_traversal operation has three arguments: p_{in}, the input port, p_{out}, the output port, and f, the flit. Switching activity and energy dissipation of each atomic component is computed as follows:

- Input lines. We assume that the input lines maintain the last traversed flit until the next flit_traversal operation (presumably by the output driver of input buffers). So the number of switching input lines is the Hamming distance between the last flit entering through input port p_{in}, denoted as f_{ip}, and the current flit f. Each switching input line has a single switch, so the energy dissipation is:

$$E_{input\_line} = H(f, f_{ip}) E'_{xb\_in}$$

- Output lines. Since we don't consider leakage current for now, output lines should maintain their states. So the number of switching output lines is the Hamming distance between the last flit leaving through output port p_{out}, denoted as f_{op}, and the current flit f. Each switching output line has a single switch, so the energy dissipation is:

$$E_{output\_line} = H(f, f_{op}) E'_{xb\_out}$$

- Control lines. Since control lines are physically connected with arbiter grant signals (they are just two names), we absorb control lines as part of arbiter grant signals and do not count their switching activities in the crossbar model.

So the energy dissipation of one flit_traversal operation is:

$$E_{flit\_traversal} = E_{input\_line} + E_{output\_line}$$

The simulator is responsible to maintain the state variable f_{ip} for each input port and f_{op} for each output port in order to compute flit traversal energy

7 Power model of arbiters

We model two types of arbiters: matrix arbiter and queuing arbiter. A queuing arbiter is implemented as a FIFO buffer and is identical to an input buffer in terms of power modeling, so we only derive the matrix arbiter model in this section.

7.1 Atomic components and their capacitance

Figure 5 shows the structure of a matrix arbiter.

Table 10 lists its architectural parameters.

Table 10: Matrix arbiter model architectural parameters

R	number of requests per arbiter. For a virtual channel arbiter, R is the number of virtual channels. For a switch arbiter, R is the number of switch input/output ports (minus 1 if no U-turn).

A matrix arbiter is divided into 4 atomic components: request node, priority node, grant node and internal node, the node between two levels of NOR gates.

Request node A request node has 4 capacitance components:

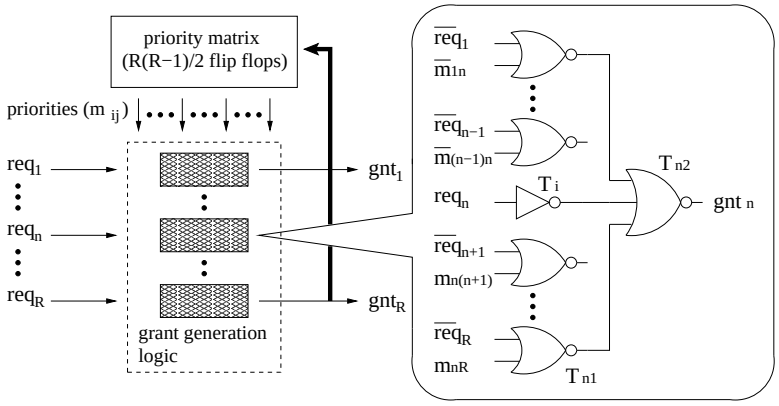

Figure 5: Structure of a matrix arbiter. T_{n1} is the first level NOR gate, T_{n2} is the second level NOR gate, T_i is the request inverter.

1. Wire. We only count the request wire length for switch arbiters. In the Alpha 21364 router, local arbiters and global arbiters are placed at the opposite sides of the crossbar, so request lines have the same length as crossbar input lines. This estimation does not hold for other layouts, so generally we use a parameter *req_len* to specify the wire length. Request lines are sparse because each port or virtual channel has one request line, so its capacitance is $C_{w0}(req\_len)$.

2. Gate end capacitance of the first level NOR gates T_{n1}. A request signal is involved in computing all the other grants, besides its own, so there are $R-1$ gate ends. The total capacitance is $(R-1)C_g(T_{n1})$.

3. Gate end capacitance of the second level NOR gate T_{n2}, which is not driven by the request signal directly, but by its complemet from T_i. Its capacitance is $C_g(T_{n2})$.

4. Inverter T_i, which generates the complemented request signal for grant computation. Its capacitance is $C_a(T_i)$.

So the request node capacitance is:

$$C_{req} = C_{w0}(req\_len) + (R-1)C_g(T_{n1}) + C_g(T_{n2}) + C_a(T_i)$$

Priority node A priority node has 2 capacitance components.

1. Gate end capacitance of the first level NOR gates T_{n1}. Each priority signal represents the priority relationship between two requesters, so it is used to compute two grants, and so the gate end capacitance is $2C_g(T_{n1})$.

2. Flip-flop switching capacitance C_{FF} since priorities are maintained by flip-flops.

So the priority node capacitance is:

$$C_{pri} = 2C_g(T_{n1}) + C_{FF}$$

Grant node A grant node has 2 capacitance components:

1. Drain end capacitance of the second level NOR gate T_{n2}. The capacitance is $C_d(T_{n2})$.
2. Crossbar control line capacitance C_{xb_ctr}. As mentioned before, the crossbar control line capacitance is counted as a part of the arbiter grant node to ease the computation of switching activity.

So the grant node capacitance is:

$$C_{gnt} = C_d(T_{n2}) + C_{xb\_ctr}$$

Internal node An internal node has 2 capacitance components:

1. Drain end capacitance of the first level NOR gate T_{n1}. The capacitance is $C_d(T_{n1})$.
2. Gate end capacitance of the second level NOR gate T_{n2}. The capacitance is $C_g(T_{n2})$.

So the internal node capacitance is:

$$C_{int} = C_d(T_{n1}) + C_g(T_{n2})$$

7.2 Operation switching activity and energy dissipation

We consider one operation for arbiters: *arbitration*.

An arbitration operation has one argument: *req*, the bit map of all request signals, with each bit representing whether a request signal is on or off. Switching activity and energy dissipation of each atomic component are computed as follows:

- Request nodes. The number of switching request nodes is simply the Hamming distance between req_0, the previous request bit map, and *req*, the current bit map. Each switching request node has a single switch, so the energy dissipation is:

$$E_{request} = H(req, req_0)E'_{req}$$

- Priority nodes. Supposing that the priority bit map before the arbitration is pri_0, after the arbitration, priorities are updated to become *pri*, so the number of switching priority nodes is the Hamming distance between pri_0 and *pri*. Each switching priority node has a single switch, so the energy dissipation is:

$$E_{priority} = H(pri, pri_0)E'_{pri}$$

- Grant nodes. Each arbitration operation grants one requester. If this arbitration grants the same requester as the previous arbitration, there is no activity on grant nodes. If this arbitration grants a different requester, then one grant node rises from 0 to 1, while another grant node falls from 1 to 0, which counts as a full switch. Let r_0 and r be the id's of the previously granted requester and the currently granted requester, the energy dissipation is:

$$E_{grant} = \begin{cases} E''_{gnt}, \text{ when } r \neq r_0 \\ 0, \text{ when } r = r_0 \end{cases}$$

- Internal nodes. The number of switching internal nodes is given by the Hamming distance between int_0, the internal node bit map before arbitration, and int, the internal node bit map after arbitration. Each switching internal node has a single switch, so the energy dissipation is:

$$E_{internal} = H(int, int_0)E'_{int}$$

So the energy dissipation of one arbitration operation is:

$$E_{arbitration} = E_{request} + E_{priority} + E_{grant} + E_{internal}$$

7.3 Transistor size

Table 11 lists all transistor sizes used in the matrix arbiter model.

Table 11: Transistor size of matrix arbiter model

Gate	$w(T, N)$	$w(T, P)$
T_{n1}	13.5λ	76λ
T_{n2}	13.5λ	76λ
T_i	12.5λ	25λ

These default values are not recommended to use without customization.

8 Capacitance routines and transistor sizing

This section explains all the low level details we have skipped so far: how c_g and c_d are computed, and how transistor size is determined based on its load capacitance.

8.1 Low level capacitance routines

Table 12 lists the capacitance parameters used in deriving $c_g(w)$ and $c_d(w, t, s)$.

Table 12: Low level capacitance parameters

Symbol	**Meaning**	**Value** (F/μm^2 or F/μm)	
		N-type	P-type
C_{poly}	poly capacitance per unit area	1.95×10^{-15}	
C_{diff_area}	diffusion area capacitance per unit area	1.37×10^{-16}	3.43×10^{-16}
C_{diff_side}	diffusion side capacitance per unit length	2.75×10^{-16}	2.75×10^{-16}
C_{diff_ovlp}	diffusion overlap capacitance per unit width	4.01×10^{-16}	4.76×10^{-16}

$\boldsymbol{c_g(w)}$ w: gate width.

Gate end capacitance is computed as the product of gate area and poly capacitance per unit area.

$$c_g(w) = w \cdot L \cdot C_{poly}$$

where L is the feature size, so $w \cdot L$ is the gate area, and C_{poly} is the poly capacitance per unit area.

$\boldsymbol{c_d(w,t,s)}$ w: gate width, t: transisor type, s: number of stacking transistors.

Drain end capacitance is the sum of three kinds of capacitance:

1. Area capacitance. Area capacitance is the product of diffusion capacitance per unit area and the total diffusion area. Diffusion length of stacked transistors is L, and diffusion length of the non-stacked transistor is $3L$ because more area is needed for metal wire connection. So the total diffusion area is $w \cdot (3L + (s-1)L)$ and the total area capacitance is:

$$C_{area} = w \cdot (3L + (s-1)L) \cdot C_{diff\_area}$$

2. Side capacitance. Side capacitance is the product of side capacitance per unit length and the total edge length between diffusions and the well. From the analysis above, we see the total edge length is $(6L + (s-1) \cdot 2L)$ since each diffusion has two edges with the well. So the total side capacitance is:

$$C_{side} = (6L + (s-1) \cdot 2L) \cdot C_{diff\_side}$$

3. Overlap capacitance. Overlap capacitance is the product of overlap capacitance per unit width and the total overlaping width between diffusions and gates. Each transistor has 2 edges overlaping with its gate, whose width is w. One edge belongs to the diffusion connected to V_{dd} or ground, so it never switches. So the total overlap capacitance is:

$$C_{over} = w \cdot (2s - 1) \cdot C_{diff\_ovlp}$$

If w is greater than 25λ, we assume that the transistor is two-way folded to conserve area. This affects the drain capacitance in the following ways:

1. Diffusion area of the non-stacked transistor is reduced by one half, other diffusion areas do not change. So the total diffusion area is $w \cdot (\frac{3}{2}L + (s-1)L)$ and the total area capacitance is:
$$C_{area} = w \cdot \left(\frac{3}{2}L + (s-1)L\right) \cdot C_{diff\_area}$$
2. Two-way folding doubles the number of edges of stacked transistors with the well because the diffusion of a stacked transistor is split into two parts, yet diffusion length is unchanged, so the total edge length is $(6L + (s-1) \cdot 4L)$ and the total side capacitance is:
$$C_{side} = (6L + (s-1) \cdot 4L) \cdot C_{diff\_side}$$
3. Overlap capacitance does not change since two-way folding reduces the width of each diffusion-gate edge by one half and doubles the number of diffusion-gate edges, so the total edge width is unchanged.

So the overall drain capacitace is:
$$c_d(w,t,s) = C_{area} + C_{side} + C_{over}$$

8.2 Load dependent transistor sizing

Table 13 lists the resistance parameters used in transistor resizing.

Table 13: Low level resistance parameters

r_N^0	resistance of 1μm-wide N-type diffusion	9723 ($\Omega \cdot \mu$m)
r_P^0	resistance of 1μm-wide P-type dissusion	22400 ($\Omega \cdot \mu$m)

These numbers are from a 0.8μm technology process and the diffusion length is the feature size. They scale linearly with fabrication technology.

The procedure of determining a transistor size according to its loading capacitance takes two arguments: C_{load}, the load capacitance, and t, the expected rise or fall time. It has 2 steps:

1. From C_{load} we can compute the resistance r, which, if serially connected with C_{load}, can achieve rise time or fall time t. If t is the rise time, then r is the resistance of the PMOS transistor when it is on. If t is the fall time, then r is the resistance of the NMOS transistor when it is on.
2. Transistor size is then $\frac{r_N^0}{r}$ or $\frac{r_P^0}{r}$.

If both rise time and fall times are given, we can determine both PMOS and NMOS transistor sizes using this procedure. If only one expected time is present, we treat it as both rise and fall times. If only one transistor is required, e.g. precharging, we only compute once.