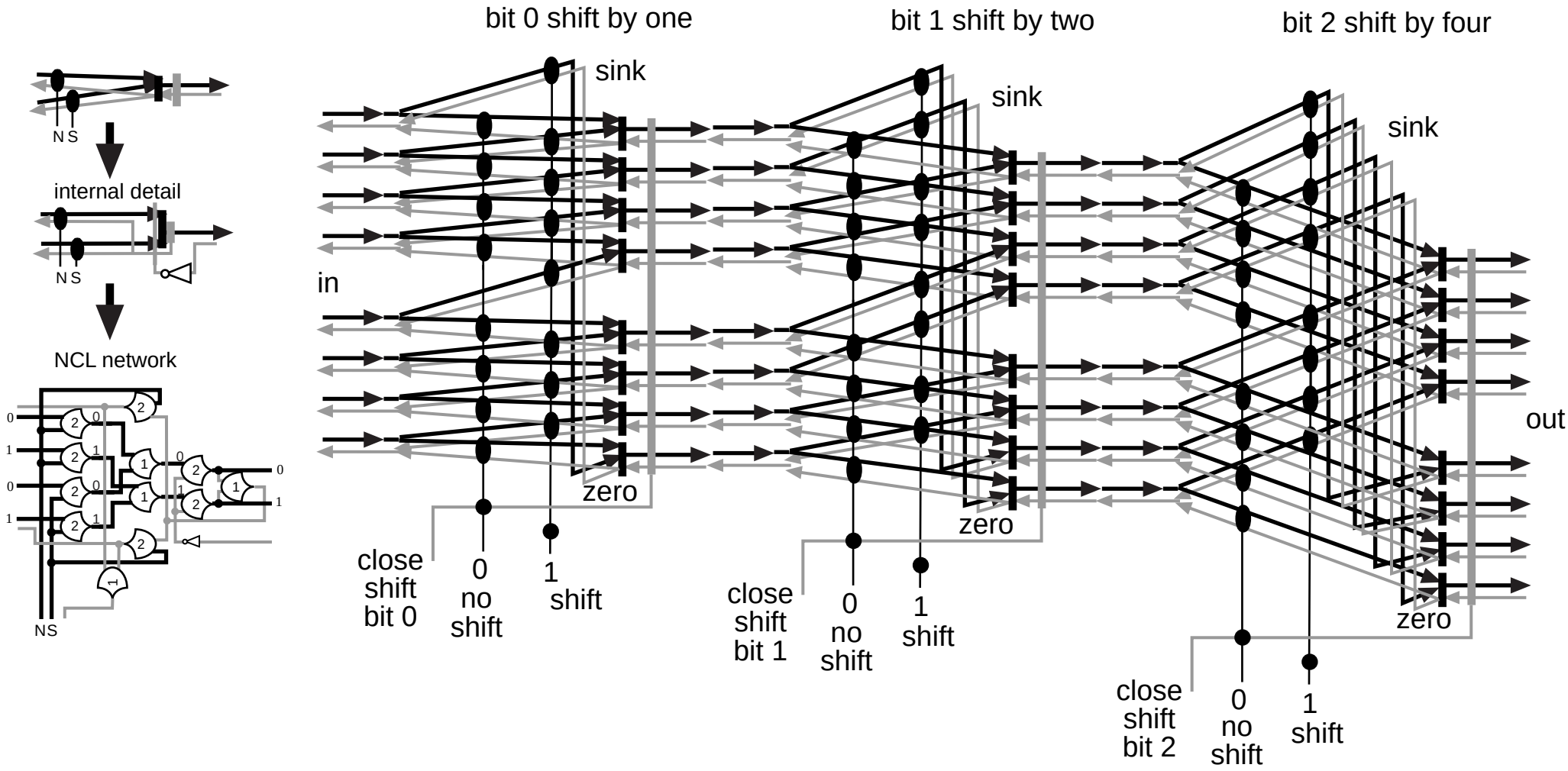
Sandbox 9

Bit Shifting

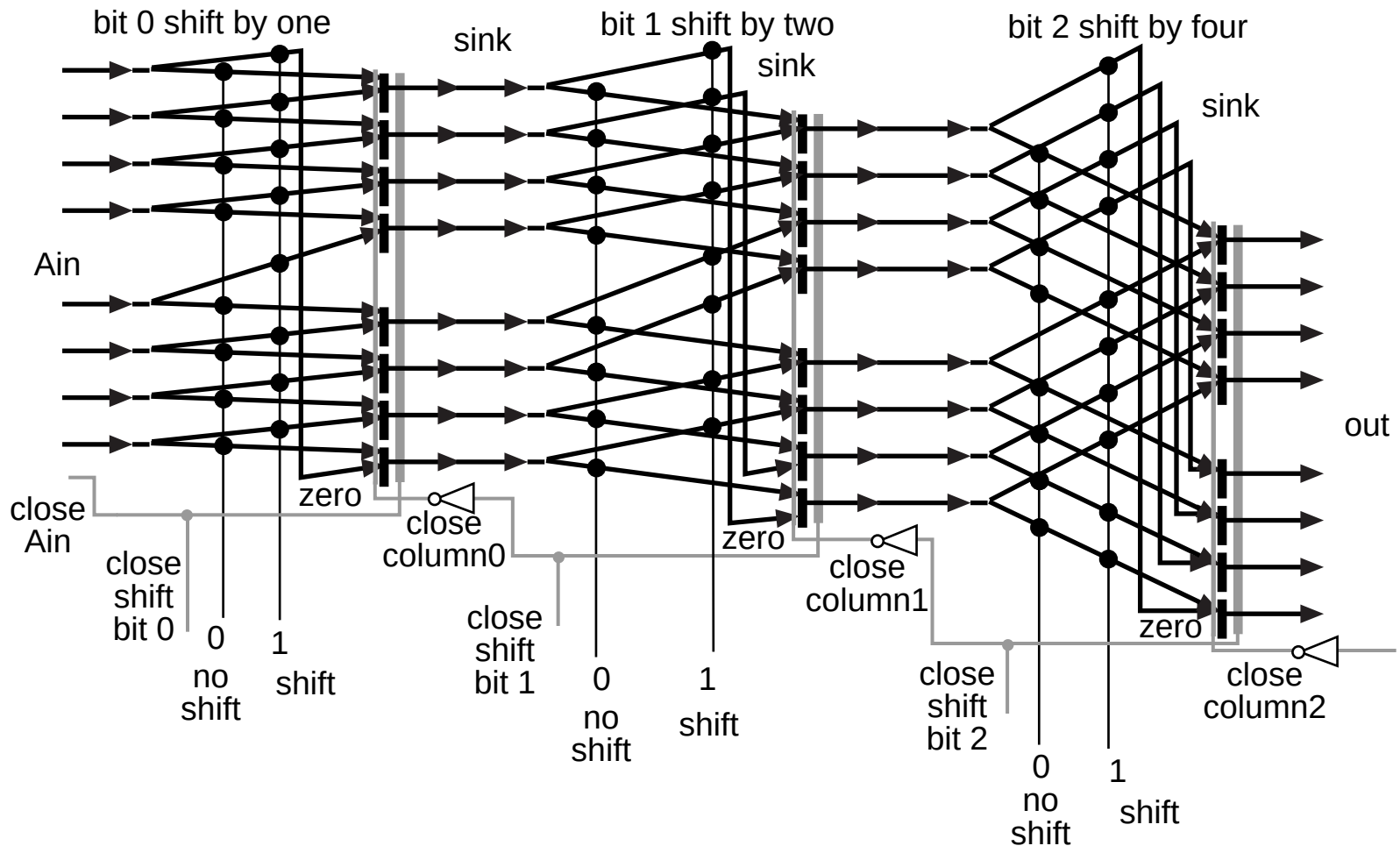
Pipelined Shifter
Digit Wise Completeness for Data Flow
Full Word Completeness for Each shift Bit

make fullwordshiftF	fullword_lshift32F.v	compliant input flow testbench
make fullwordshift2D	fullword_lshift2D32.v	2D wavefront input flow testbench
make fullwordshiftragB	fullword_lshiftragB32.v	ragged wavefront input flow testbench



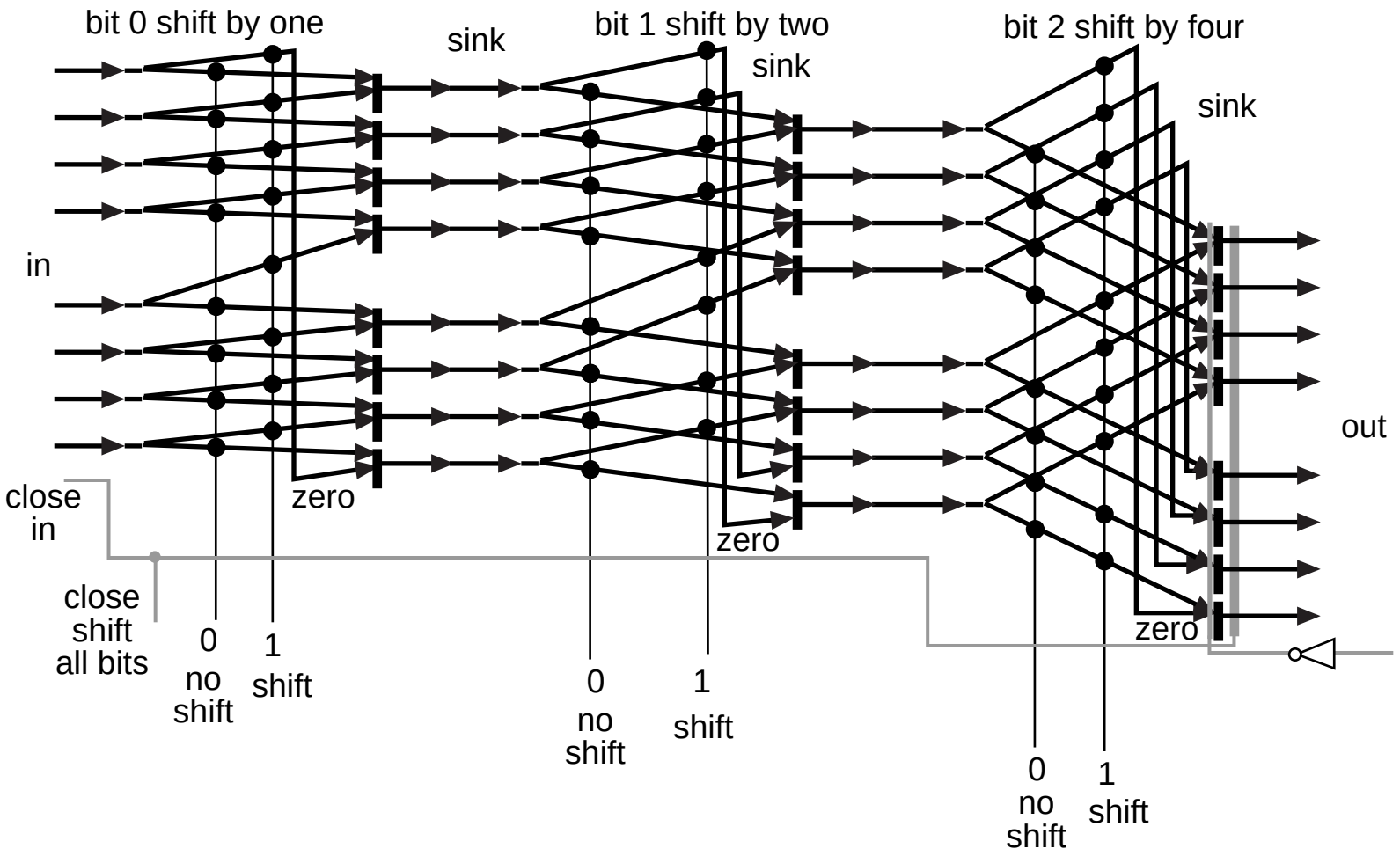
Pipelined Shifter
Full Word Completeness for Data Flow
Full Word Completeness for Each shift Bit

make fullwordshiftG fullword_lshift32G.v compliant input flow testbench



Combinational Shifter
Single Oscillation, Single Completeness

make fullcomboshift fullcombo_lshift32.v compliant input flow testbench



Pipelined Shifter
Digit Wise Completeness for Data Flow
Full Word Completeness for Each shift Bit

```
column0(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 1
// digit close in, fullword close steer

//  pass or endoffs to zero fill
({[in/0, steer/noshift], [in/31, steer/shift] -> zero} -> out/0);

//   pass or shift
({[in/[1-31], steer/noshift], [in/[0-30], steer/shift]} -> out/[1-31]);

close in/[0-30] <- {[steer/noshift, out/[0-30]#], [steer/shift, out/[1-31]/#};  // pass range
close in/31 <- {[steer/noshift, out/31#], [steer/shift, out/0#]};  // sink range
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

```
Lshift(A, shift -> shifted){
flow [A, steer] -> shifted;
token [31:0]{1:0} A, shifted;
token [3:0][31:0]{1:0} out;
token [4:0]{shift, noshift} shift;
column0(A, shift/0 -> out/0);
column1(out/0, shift/1 -> out/1);
column2(out/1, shift/2 -> out/2);
column3(out/2, shift/3 -> out/3);
column4(out/3, shift/4 -> shifted);
close A <- shifted/#;
close shift <- shifted/#;
close shifted <- ?/#;
}
```

Pipelined Shifter
Digit Wise Completeness for Data Flow
Full Word Completeness for Each shift Bit

```
column1(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 2
// digit close in, fullword close steer

//  pass or endoffs to zero fill
({[in/[0-1], steer/noshift], [in/[30-31], steer/shift] -> zero} -> out/[0-1]);

//   pass or shift
({[in/[2-31], steer/noshift], [in/[0-29], steer/shift]} -> out/[2-31]);

close in/[0-29] <- {[steer/noshift, out/[0-29]#], [steer/shift, out/[2-31]/#};  // pass range
close in/[30-31] <- {[steer/noshift, out/[30-31]#], [steer/shift, out/[0-1]/#]};  // sink range
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Digit Wise Completeness for Data Flow
Full Word Completeness for Each shift Bit

```
column2(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 4
// digit close in, fullword close steer

//  pass or endoffs to zero fill
({[in/[0-3], steer/noshift], [in/[28-31], steer/shift] -> zero} -> out/[0-3]);

//   pass or shift
({[in/[4-31], steer/noshift], [in/[0-27], steer/shift]} -> out/[4-31]);

close in/[0-27] <- {[steer/noshift, out/[0-27]#], [steer/shift, out/[4-31]/#};  // pass range
close in/[28-31] <- {[steer/noshift, out/[28-31]#], [steer/shift, out/[0-3]/#]};  // sink range
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Digit Wise Completeness for Data Flow
Full Word Completeness for Each shift Bit

```
column3(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 8
// digit close in, fullword close steer

//  pass or endoffs to zero fill
({[in/[0-7], steer/noshift], [in/[24-31], steer/shift] -> zero} -> out/[0-7]);

//   pass or shift
({[in/[8-31], steer/noshift], [in/[0-23], steer/shift]} -> out/[8-31]);

close in/[0-23] <- {[steer/noshift, out/[0-23]#], [steer/shift, out/[8-31]/#};  // pass range
close in/[24-31] <- {[steer/noshift, out/[24-31]#], [steer/shift, out/[0-7]/#]};  // sink range
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Digit Wise Completeness for Data Flow
Full Word Completeness for Each shift Bit

```
column4(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 16
// digit close in, fullword close steer

//  pass or endoffs to zero fill
({[in/[0-15], steer/noshift], [in/[16-31], steer/shift] -> zero} -> out/[0-15]);

//   pass or shift
({[in/[16-31], steer/noshift], [in/[0-15], steer/shift]} -> out/[16-31]);

close in/[0-15] <- {[steer/noshift, out/[0-15]#], [steer/shift, out/[16-31]/#};  // pass range
close in/[16-31] <- {[steer/noshift, out/[16-31]#], [steer/shift, out/[0-15 ]/#]};  // sink range
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Digit Wise Closure to Full Word Closure

Convert digit wise closure for column 2 into full word closure for column 2

Digit Wise Closure

close in/[0-27] <- {[steer/noshift, out/[0-27]#], [steer/shift, out/[4-31]/#}; // pass range
close in/[28-31] <- {[steer/noshift, out/[28-31]#], [steer/shift, out/[0-3]/#]}; // sink range

Remove redundancy of **close** in and out/#

close in/[0-31] <- out/[0-31]/#;

Simplify

close in <- out/#;

Pipelined Shifter
Full Word Completeness for Data Flow
Full Word Completeness for Each shift Bit
Column 0

Demote unnamed component references to combinational references

```
Lshift(A, shift -> shifted){
flow [A, steer] -> shifted;
token [31:0]{1:0} A, shifted;
token [3:0][31:0]{1:0} out;
token [4:0]{shift, noshift} shift;
column0(A, shift/0 -> out/0);
column1(out/0, shift/1 -> out/1);
column2(out/1, shift/2 -> out/2);
column3(out/2, shift/3 -> out/3);
column4(out/3, shift/4 -> shifted);
close A <- shifted/#;
close shift <- shifted/#;
close shifted <- ?/#;
}
```

```
column0(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 1
// fullword close in, fullword close steer

//  pass or endoffs to zero fill
{[in/0, steer/noshift], [in/31, steer/shift] -> zero} -> out/0;

//   pass or shift
{[in/[1-31], steer/noshift], [in/[0-30], steer/shift]} -> out/[1-31];

close in <- out/#
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Full Word Completeness for Data Flow
Full Word Completeness for Each shift Bit
Column 1

```
column1(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 2
// fullword close in, fullword close steer

//  pass or endoffs to zero fill
{[in/[0-1], steer/noshift], [in/[30-31], steer/shift] -> zero} -> out/[0-1];

//   pass or shift
{[in/[2-31], steer/noshift], [in/[0-29], steer/shift]} -> out/[2-31];

close in <- out/#
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Full Word Completeness for Data Flow
Full Word Completeness for Each shift Bit
Column 2

```
column2(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 4
// fullword close in, fullword close steer

//  pass or endoffs to zero fill
{[in/[0-3], steer/noshift], [in/[28-31], steer/shift] -> zero} -> out/[0-3];

//   pass or shift
{[in/[4-31], steer/noshift], [in/[0-27], steer/shift]} -> out/[4-31];

close in <- out/#
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Full Word Completeness for Data Flow
Full Word Completeness for Each shift Bit
Column 3

```
column3(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 8
// fullword close in, fullword close steer

//  pass or endoffs to zero fill
{[in/[0-7], steer/noshift], [in/[24-31], steer/shift] -> zero} -> out/[0-7];

//   pass or shift
{[in/[8-31], steer/noshift], [in/[0-23], steer/shift]} -> out/[8-31];

close in <- out/#
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Pipelined Shifter
Full Word Completeness for Data Flow
Full Word Completeness for Each shift Bit
Column 4

```
column4(in, steer -> out){
flow [in, steer] -> out;
token [31:0]{1:0} in, out;
token {shift, noshift} steer;
token {1:0} zero(/0);

// shift by 16
// fullword close in, fullword close steer

//  pass or endoffs to zero fill
{[in/[0-15], steer/noshift], [in/[16-31], steer/shift] -> zero} -> out/[0-15];

//   pass or shift
{[in/[16-31], steer/noshift], [in/[0-15], steer/shift]} -> out/[16-31];

close in <- out/#
close steer <- out/#;
close out/[0-31] <- ?/[0-31]#;
}
```

Combinational Shifter
Single Oscillation, Single Completeness
Column 0

< > Specifies that the referenced component be composed as a combinational network instead of as a component

```
Lshift(A, shift -> shifted){
flow [A, steer] -> shifted;
token [31:0]{1:0} A, shifted;
token [3:0][31:0]{1:0} out;
token [4:0]{shift, noshift} shift;
column0<A, shift/0 -> out/0);
column1<out/0, shift/1 -> out/1>;
column2<out/1, shift/2 -> out/2>;
column3<out/2, shift/3 -> out/3>;
column4<out/3, shift/4 -> shifted>;
close A <- shifted/#;
close shift <- shifted/#;
close shifted <- ?/#;
}
```

The completeness of out0 implies the completeness of A
The completeness of out1 implies the completeness of out0
The completeness of out2 implies the completeness of out1
The completeness of out3 implies the completeness of out2
The completeness of shifted implies the completeness of out3

Therefore
The completeness of shifted implies the completeness of A