

Architectures and Algorithms for Synthesizable Programmable Logic Cores

Noha Kafafi, Kimberly Bozman, Steve Wilton

University of British Columbia
Vancouver, B.C., Canada

This work funded by Altera, Micronet, and NSERC

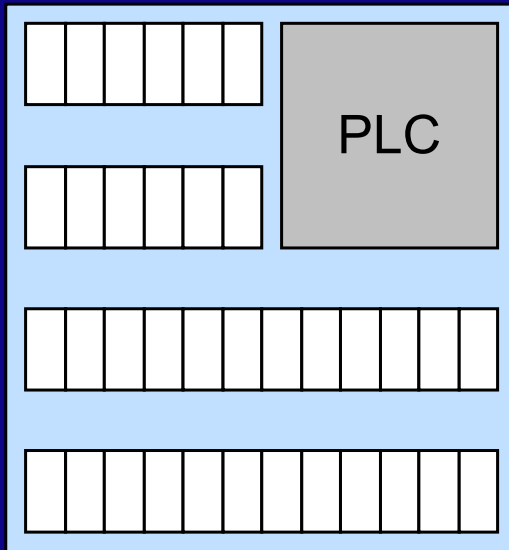
Configurable Logic Cores: Why?

1. Accommodate complexity of current designs
2. Postpone some decisions until late in design cycle
3. Fast upgrade path for products
4. Can "patch up" design errors

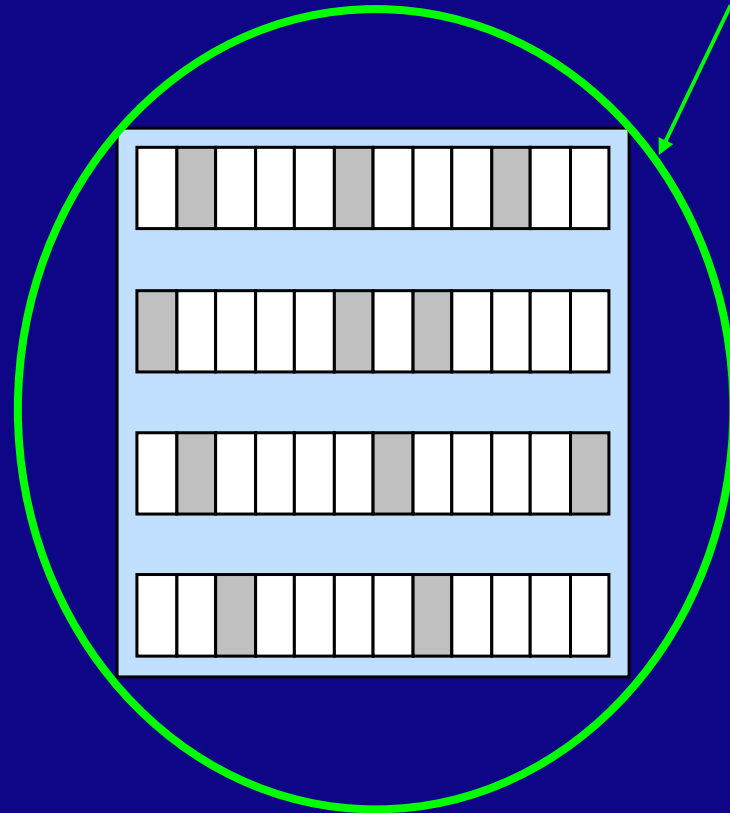
Soft Programmable Logic Cores:

Use standard cells to implement PLC:

This paper talks
about
this way

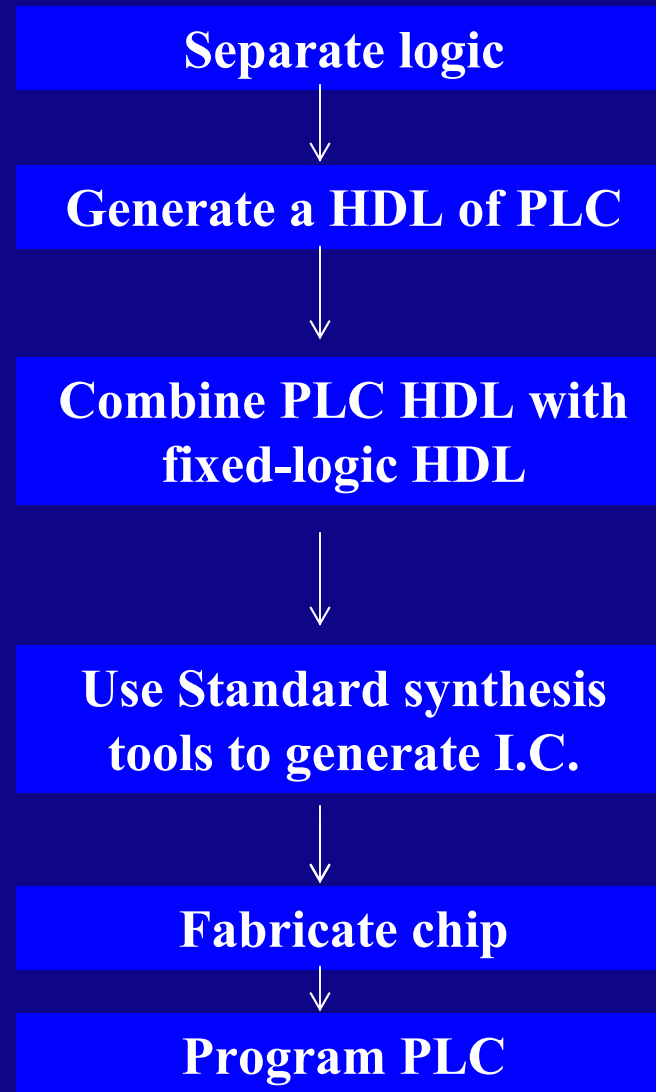


One way: Hard PLC



Alternative way: Soft PLC

How do we make a soft PLC?



Soft Programmable Logic Cores:

Advantages of using soft cores:

- + Easy to integrate. Place and route with the rest of the ASIC
- + Very flexible, can generate exactly the core you need
- + Easy to migrate to smaller technologies

Disadvantages:

- Really inefficient compared to hard core (estimate 6-7x bigger)

Our thought:

It makes sense if you only want a small PLC (a few hundred gates, perhaps)

e.g. next state logic in state machine

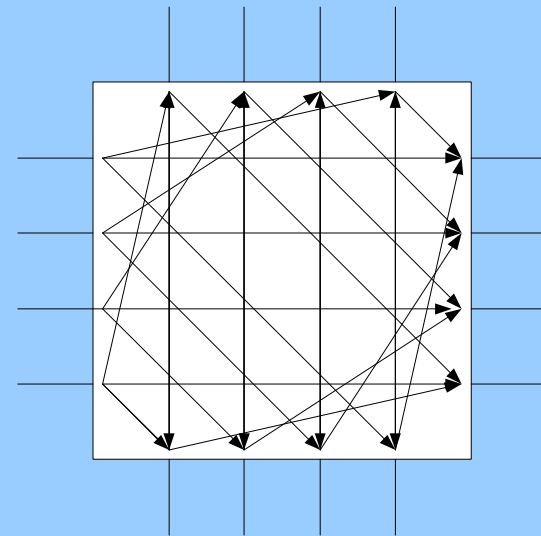
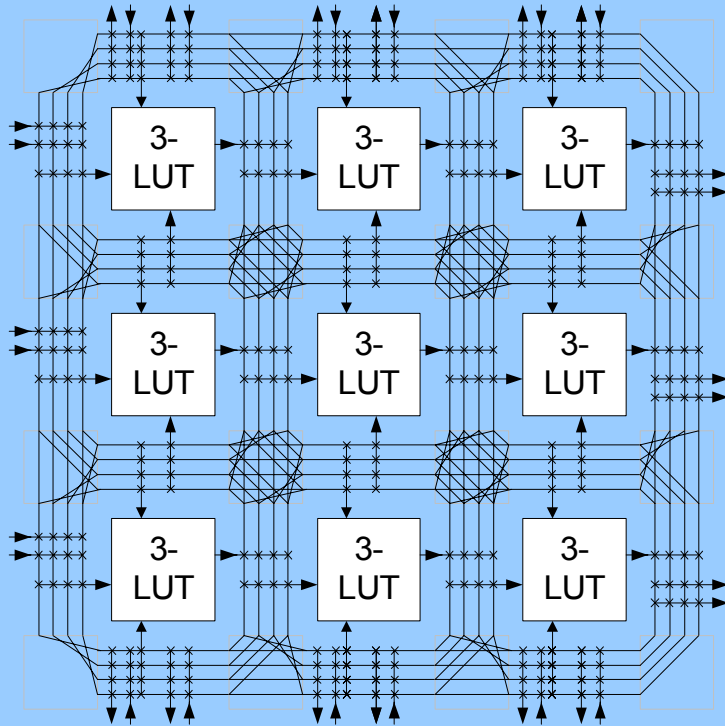
Interesting Tid-bit:

When we synthesized our programmable logic core, we had all sorts of problems with combinational loops, but an un-programmed FPGA is full of them!

Our solution:

We are now studying uni-directional architectures. Feedback would have to be done outside the PLC

Directional Architecture:

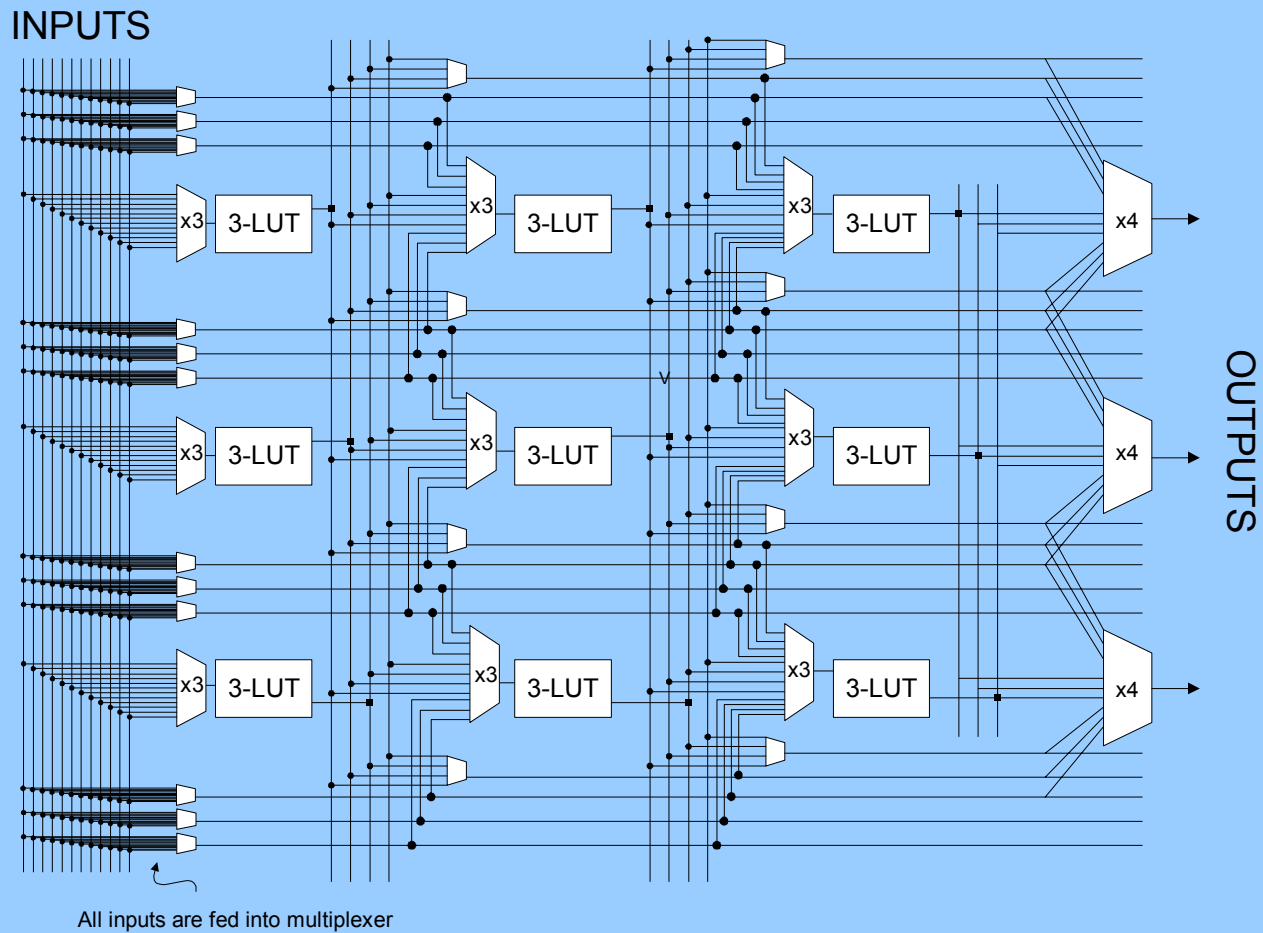


Close-up of Switch Block

A Few Interesting Observations:

1. Since we are only implementing small blocks, we can remove some flexibility
2. Since these blocks are hardwired to the rest of the chip, we still need lots of flexibility at the inputs and outputs
3. Each “tile” need not be identical

Gradual Architecture:



Computer –Aided Design Tools

Place and Route tools are needed to implement a user circuit on our core

- Need new algorithms for our architectures, to take into account:
 - directional aspect
 - new routing structure
- More details to come.....



More on Placement and Routing

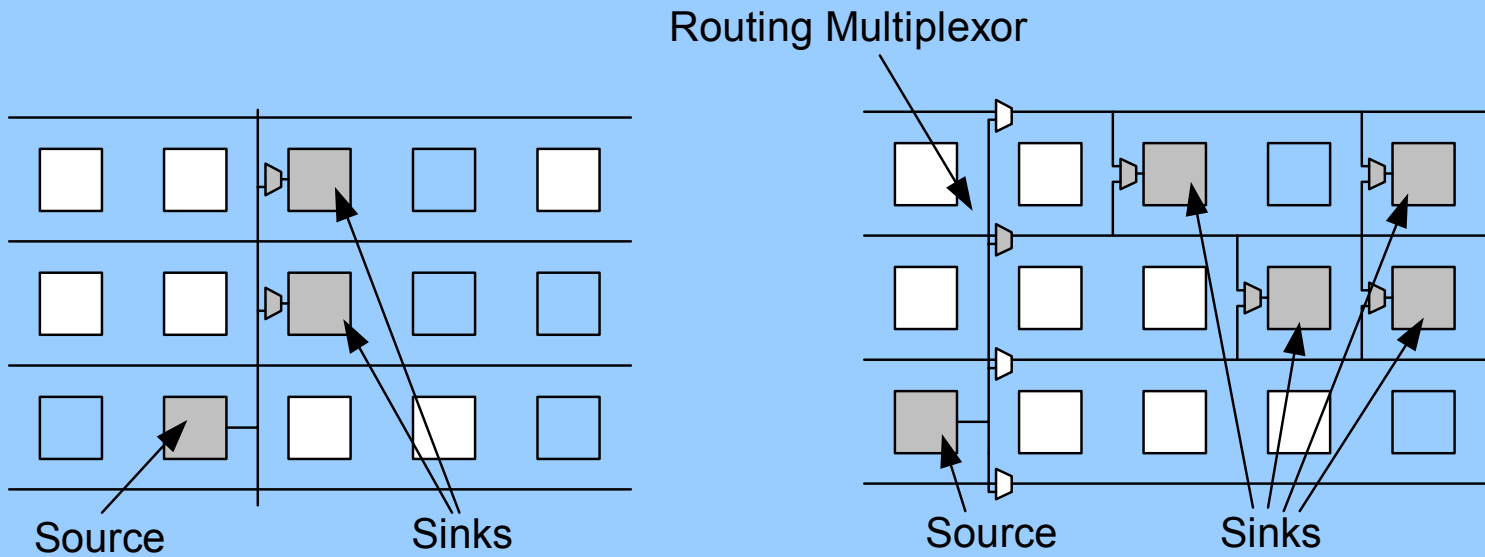
Placement:

- Used simulated annealing algorithm
- Normal FPGA algorithms based on wire length or critical path delay
- For our architecture, routing resources are very limited => minimize overuse of routing multiplexors
- Goal is to achieve “Placement for Routability”

Routing:

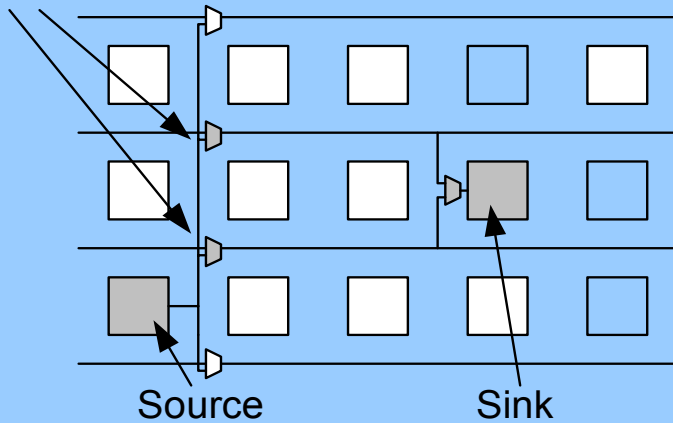
- Turns out this is an easy problem
- Normal FPGA routers work well

Some Good Placements

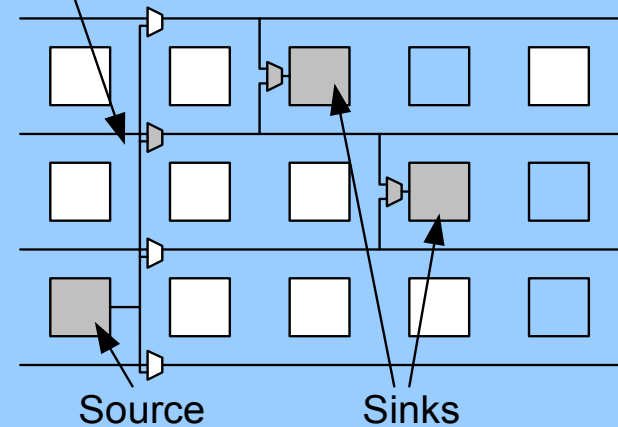


Estimating Mux Usage During Placement:

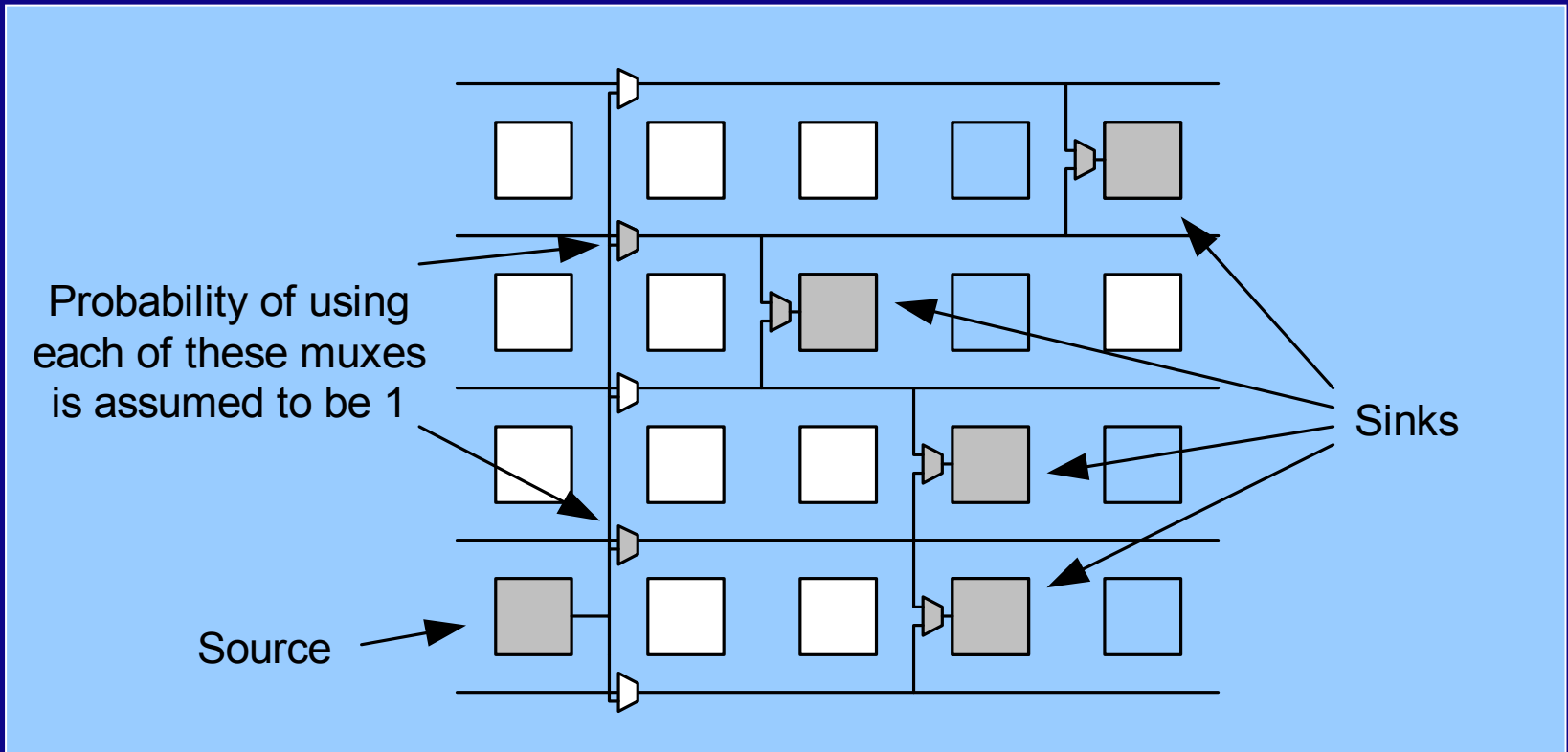
Probability of using each mux is 0.5



Probability of using this mux is about 1



Estimating Mux Usage During Placement:



Gradual Placement Costs

$$\text{cost} = \sum \sum [\text{MAX}(0, \text{Occ}(x,y) - \text{Cap}(x,y) + \gamma)]$$

Occupancy of mux at location (x,y), is an estimate of how many nets would potentially use that routing mux

Capacity of mux at location (x,y) is equal to 1

Measurements and Estimates:

Experimental Area Results:

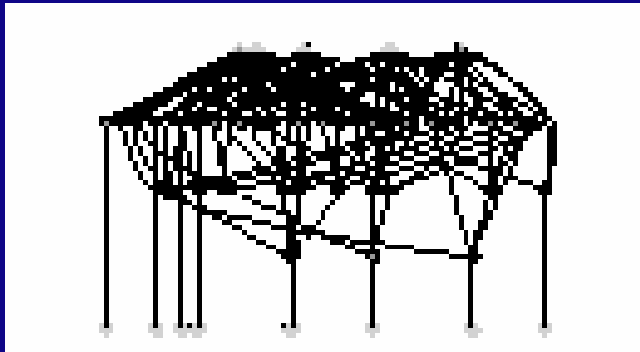
- Used 20 MCNC benchmark circuits
- Found the minimum sized core on which a circuit placed and routed successfully
- Synthesized a core of that size and obtained area from Design Analyzer
- Result: Gradual Architecture is 19% more dense than Directional Architecture

Area Estimate:

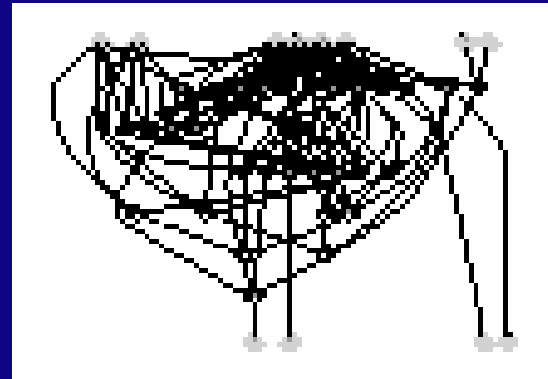
- Compared to the same size hard-FPGA, our soft FPGA is about 6.4x less dense

Combinational Circuits:

From Hutton et al:



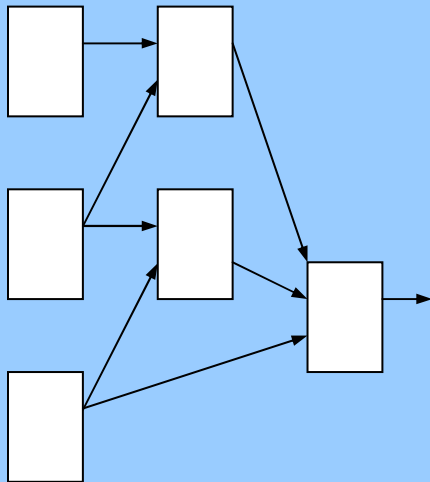
suar5.blif



sqrt8ml.blif

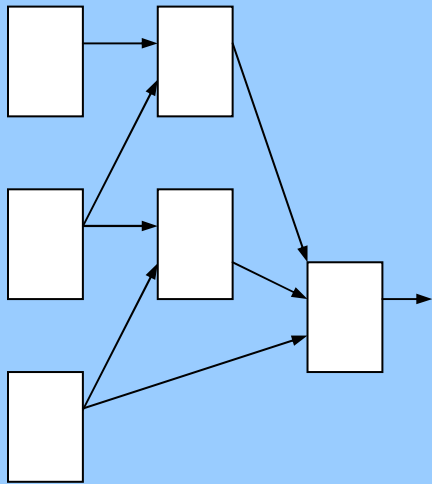
Combinational logic naturally has a “triangular” shape

Non-Rectangular Cores:

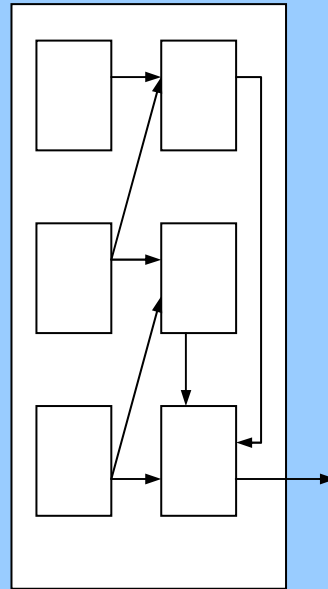


Original Circuit

Non-Rectangular Cores:

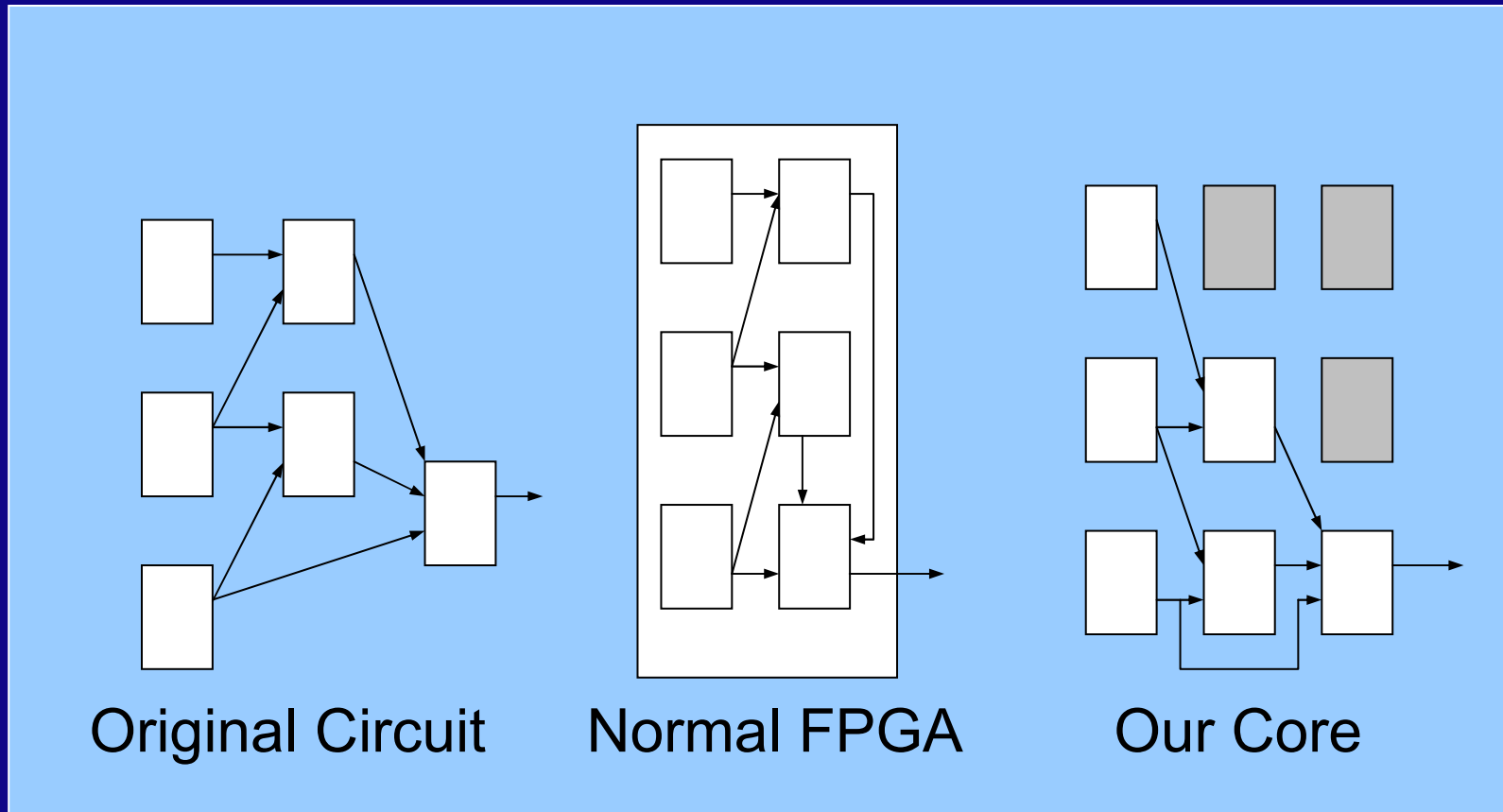


Original Circuit



Normal FPGA

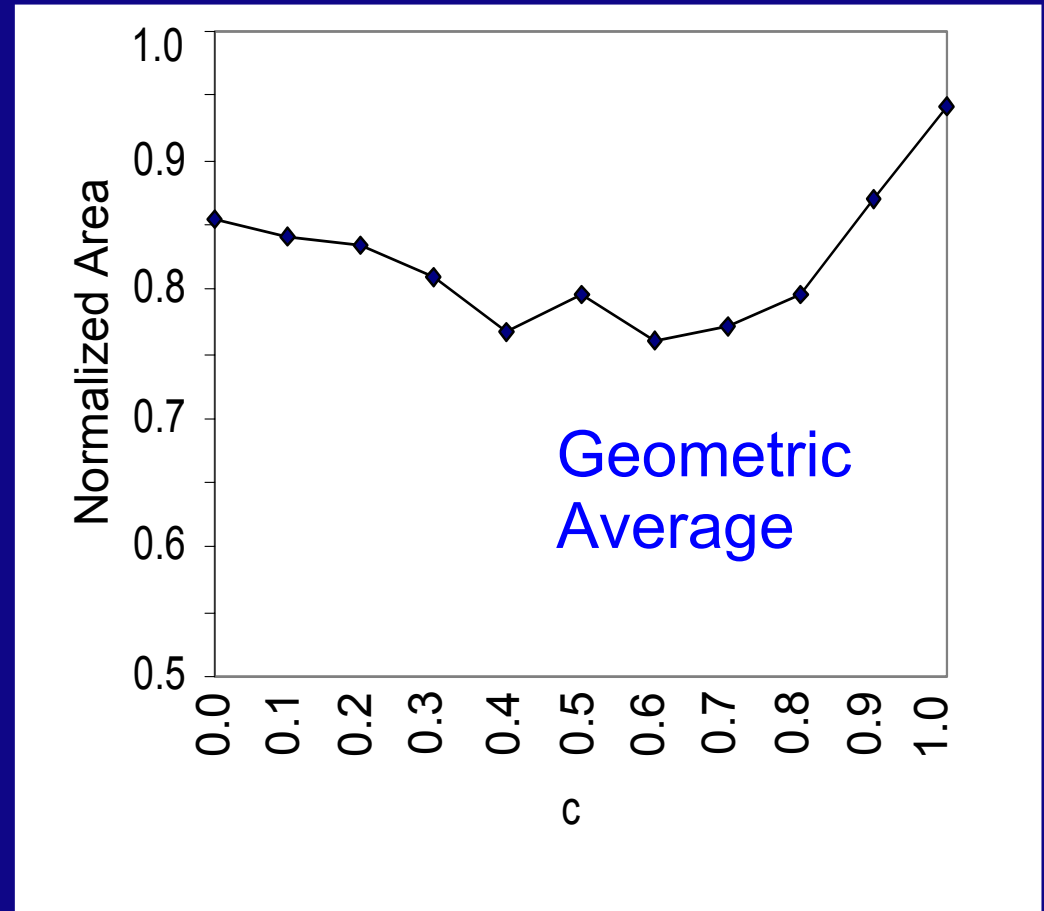
Non-Rectangular Cores:



Remember: Since we are synthesizing these cores with standard cells, the actual layout will not be triangular

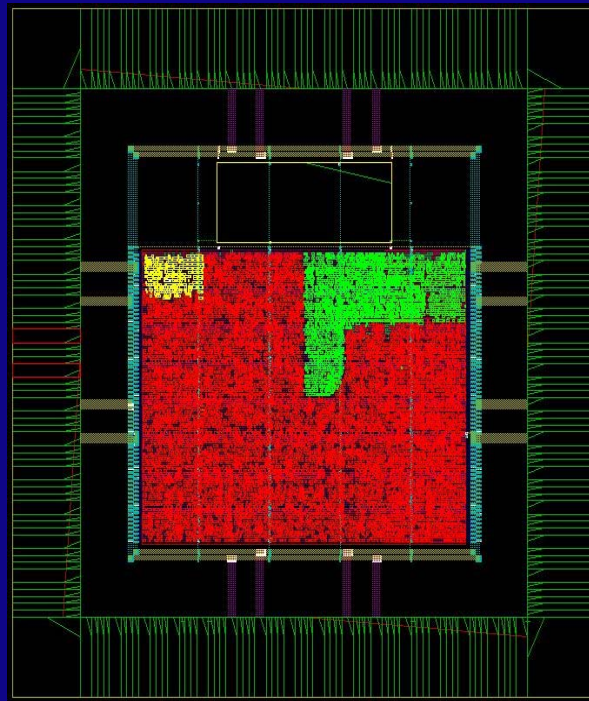
Non-Rectangular Cores:

- As c increases, cores are more triangular => less area, but eventually core size increases and area increases again
- Using a c value of 0.6 results in 11% area savings, on average



Chip Layout

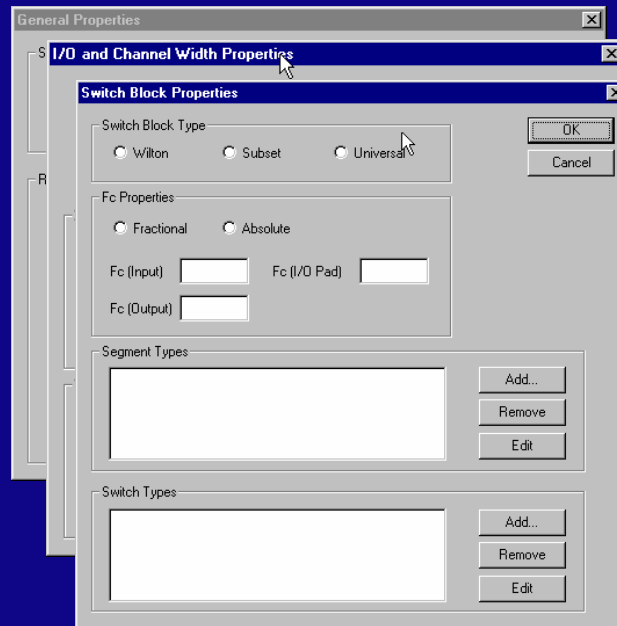
Chip with embedded 8x8 Gradual Core



884106 μm^2

What comes next?

Long-Term Goal: Programmable Logic Core Generator:



Architecture
Generator

```
-- File: decoder1.vhd
-- Date: January 20th, 2002
-- Authors: Noha Kafafi
--          Kimberly Bozman
--
-- Description: Decoder block for lut
-- inputs: DEN (decode enable)
--          shift_in (config bits in one bit)
--          output: shift_out (config bits out variable
length)
-- Notes: Nothing to change for size upgrade
--
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;

entity decoder1 is
    generic (config_width : INTEGER := 25);
    port ( signal DEN : in STD_LOGIC;
          signal shift_in: in STD_LOGIC;
          signal shift_out: out STD_LOGIC_VECTOR(config_width-1
DOWNTO 0) );
end decoder1;

architecture rtl of decoder1 is
    signal i_bus : std_logic_vector(config_width-1 downto 0);
    begin

    shift_out <= i_bus;

    -- When decode enable is low then shift in configuration bits
    process (DEN, i_bus)
    begin
        if rising_edge(DEN) then
            for i in 0 to config_width-2 loop
                i_bus(i+1) <= i_bus(i);
            end loop;
            i_bus(0) <= shift_in;
        end if;
    end process;

end rtl ;
```

Summary

Soft Cores are viable!

Compared to a hard-core, 6.4 x less dense

Our Gradual Soft Core Architecture is 19% more dense than directional architecture

We've built a real chip (it's at TSMC now)