Reducing Power in an FPGA via Computer-Aided Design

Steve Wilton
University of British Columbia

Power Reduction via CAD

How to reduce power dissipation in an FPGA:

- Create power-aware CAD tools
- Create power efficient architectures
- Use process enhancements
- Some combination of the above

In this part of the tutorial: Power-aware CAD tools

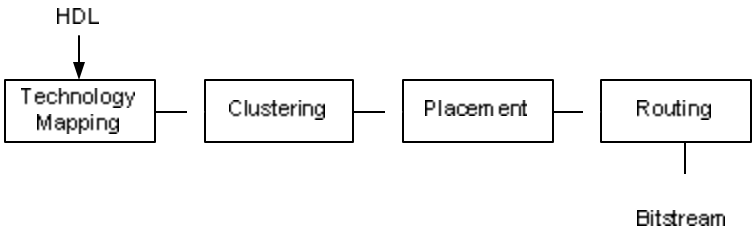
Key point: We can save a significant amount of power without modifying the FPGA architecture at all

What you'll know by the end of my talk:

- How FPGA CAD tools can be made power-aware
- Which steps of the FPGA CAD flow are most amenable to reducing power
- How much we can reduce power by optimizing CAD

FPGA CAD Flow

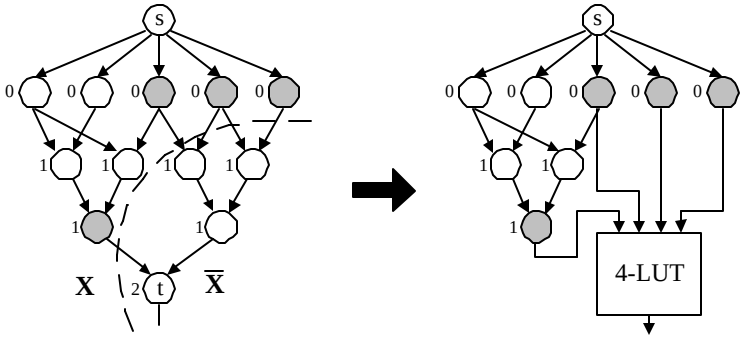
A typical FPGA CAD flow:



We'll talk about each of these independently and then put them together

Technology Mapping

Mapping gates to LUTs:



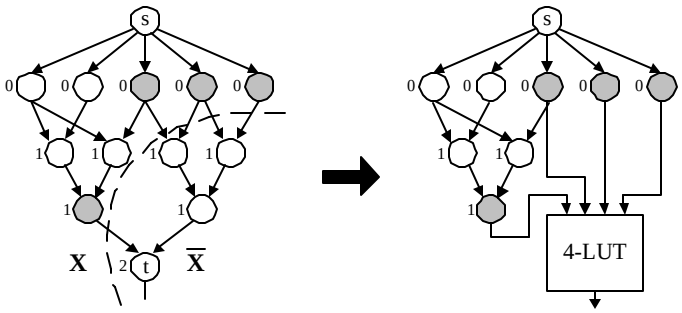
Each LUT can implement any function of its inputs

- FPGA Tech-mapping algorithms take advantage of this

Technology Mapping

Typical algorithm to map to a *k*-input LUT (Cong et al, UCLA)

- Find one or more “cuts” for each node
  - each cut has at most *k* signals cut
- Use the cut(s) for each node to construct the circuit



Technology Mapping

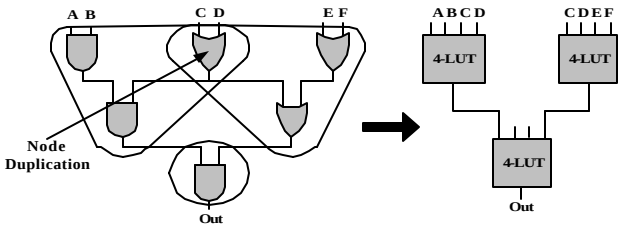
To make this power-aware:

1. Choose a cut for each node intelligently:
   - For nodes on the critical path, choose “highest” cut to optimize depth
   - For other nodes, prefer cuts that cut signals with low estimated activity values

Li et al (U. South Florida)

Technology Mapping

2. Reduce node duplication (Anderson, Najm, U. Toronto)



Necessary to find delay-optimal mapping, but bad for power:
- higher the depth, the less activity
- node duplication increases fan-out of fan-in nodes
  - the fan-in nodes have higher activity

Technology Mapping

Combine these ideas into a single algorithm:

Phase 1:

- Construct a set of K-feasible cuts for each node

Phase 2:

For each node:

- If the node is on the critical path
  - Choose a cut that is "min-height"
  - If there is more than one, use a cost function
- Otherwise
  - Choose the cut based on the cost function

Technology Mapping

The cost function:

$$\frac{1+\left|rooted(\overline{Xv})\right|}{1+\left|\overline{Xv}\right|-\left|rooted(\overline{Xv})\right|}\bullet\sum_{u\in input(\overline{Xv})}\frac{weight(u)\bullet(1+l\quad act(u))}{\left|output(u)\right|}$$

Limits node duplication by penalizing mappings in which LUTs overlap

Technology Mapping

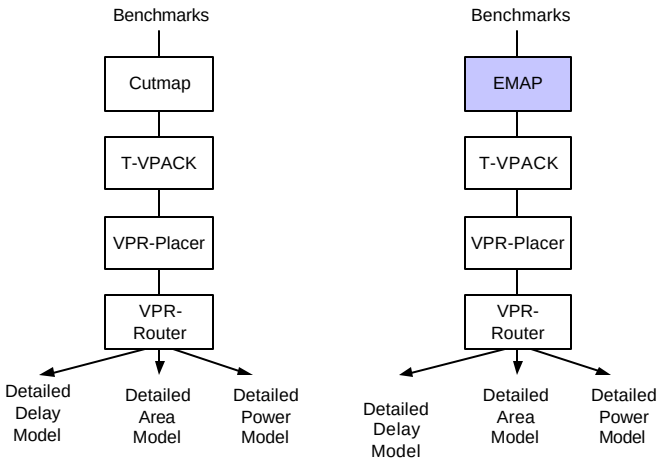
The cost function:

$$\frac{1+\left|rooted(\overline{Xv})\right|}{1+\left|\overline{Xv}\right|-\left|rooted(\overline{Xv})\right|}\bullet\sum_{u\in input(\overline{Xv})}\frac{weight(u)\bullet(1+l\ act(u))}{\left|output(u)\right|}$$

Estimated activity (Transition density model)

Sum over all cut signals

To evaluate the algorithm…



Detailed Power Model: Static, Short Circuit, Dynamic

Uses transition density model (Najm)

Technology Mapping Results:

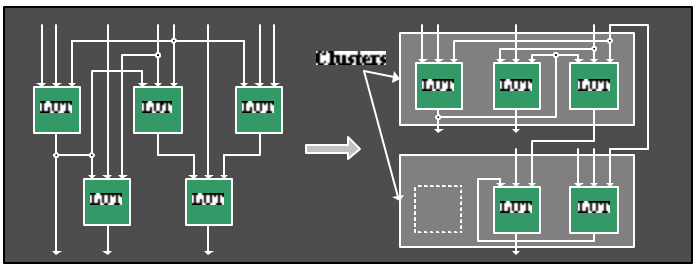
	LUTs	Connections	Activity	Energy (nJ)
CutMap	2576	10746	0.330	2.18
EMap	2441	9705	0.323	2.01
% Diff	-5.2	-9.7	-2.1	-7.6

For Emap, most of the savings come from minimizing unnecessary node duplication.

Clustering:

FPGA logic blocks usually contain several LUTs:

Altera: LABs Xilinx: CLBs



Clustering groups LUTs into LAB-sized clusters

- Connecting LUTs within a LAB is cheap (speed/power/area)

Clustering

Typical FPGA Clustering Algorithm: (TVPACK)

```
while there are unclustered LUTs {
    choose a LUT for a new cluster
    while this cluster is not full {
        choose another LUT and add it to cluster
    }
}
```

Cost Function 1

Cost Function 2

Clustering

Cost Function 1: Choosing a seed for a new cluster:

Original: Choose the most timing-critical LUT

Power-Aware: Choose LUT with the highest activity pins

Clustering

Cost Function 1: Choosing a seed for a new cluster:

Original: Choose the most timing-critical LUT

Power-Aware: Choose LUT with the highest activity pins

Cost Function 2: Choosing a LUT to add to a cluster:

Original: $\lambda * \text{Criticality}(B) + (1-\lambda) * \frac{|SharedNet(B,C)|}{K}$

Prefer LUTs that are timing critical

Prefer LUTs that share nets with those already in cluster

Clustering

Cost Function 1: Choosing a seed for a new cluster:

Original: Choose the most timing-critical LUT

Power-Aware: Choose LUT with the highest activity pins

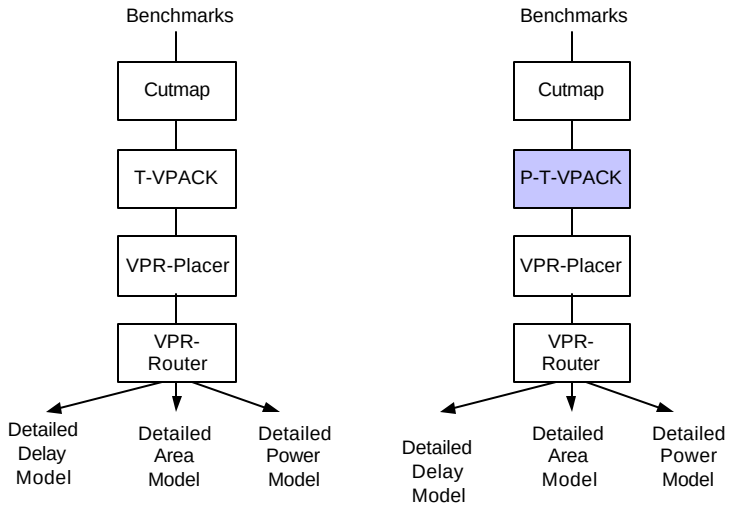
Cost Function 2: Choosing a LUT to add to a cluster:

Original: $\lambda * \text{Criticality}(B) + (1-\lambda) * \frac{|SharedNet(B,C)|}{K}$

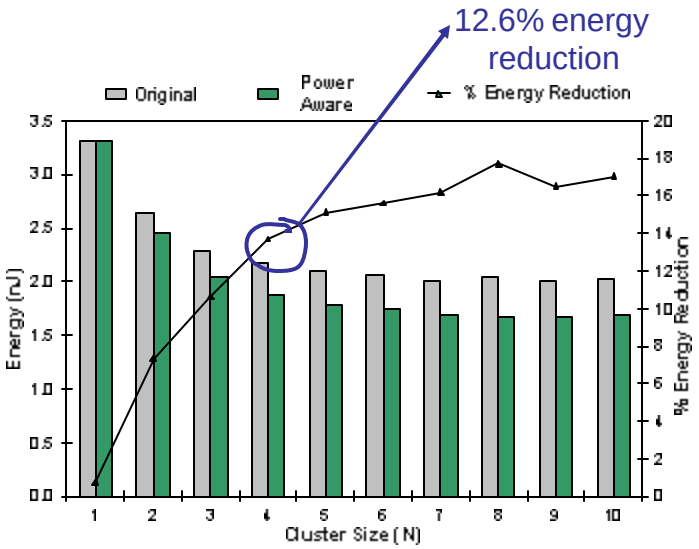
Power Aware: $\text{Criticality}(B) + \alpha \frac{\sum Weight(i) \mid i \in SharedNets(B,C)}{K} + \beta \frac{\sum Act(i) \mid i \in SharedNets(B,C)}{K * Act_{avg}}$

Prefer to encapsulate as much activity as possible

To evaluate the algorithm…



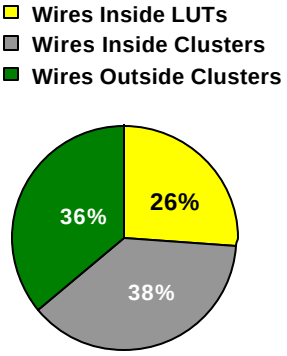
Clustering Results:



Clustering Results:

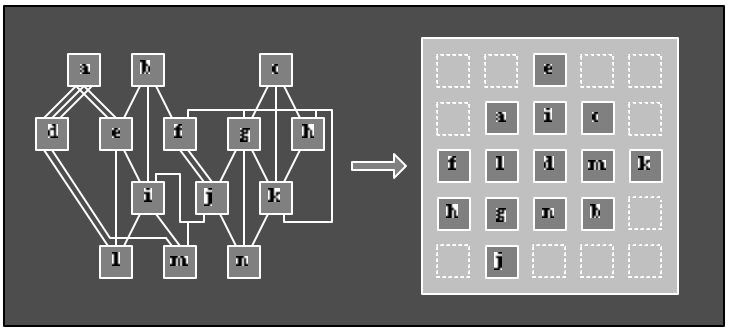
Clustering savings are greater than tech. map savings since clusters are bigger than LUTs

Savings are obtained by hiding high activity wires within clusters



Placement:

Assign physical location to clusters:



Goals:

Routability: place tightly connected blocks near each other

Speed: make critical paths short

Power: make high-activity nets short

Placement:

Original Algorithm:

$$\Delta C_{old} = l \frac{\Delta \text{timing cost}}{\text{previous timing cost}} + (1-l) \frac{\Delta \text{wiring cost}}{\text{previous wiring cost}}$$

Placement:

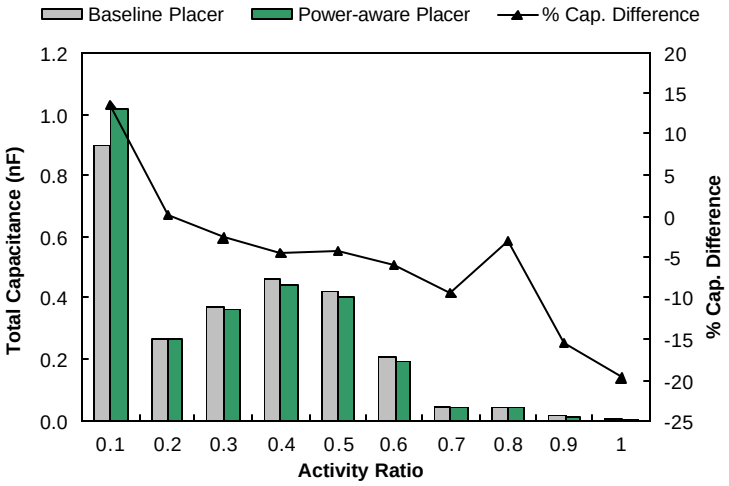
Original Algorithm:

$$\Delta C_{old} = l \frac{\Delta \text{timing cost}}{\text{previous timing cost}} + (1-l) \frac{\Delta \text{wiring cost}}{\text{previous wiring cost}}$$

Power-Aware Algorithm:

$$\Delta C_{old} = l \frac{\Delta \text{timing cost}}{\text{previous timing cost}} + (1-l) \frac{\Delta \text{wiring cost}}{\text{previous wiring cost}} + b \frac{\Delta \text{power cost}}{\text{previous power cost}}$$

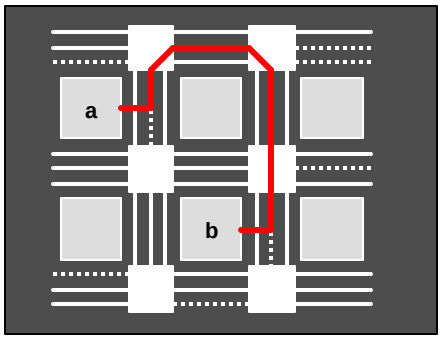
Placement:



-> 3% Energy Improvement

Routing:

Connect logic blocks using prefabricated routing tracks:



Routability: avoid congested areas if possible

Speed: make critical paths short

Power: make high-activity nets short

Routing:

Most FPGA Routers use negotiated congestion:

```
overuse cost of each resource is 0
while we do not have a legal route {
    route each net using shortest path algorithm
    increase the cost of sharing a resource
}
```

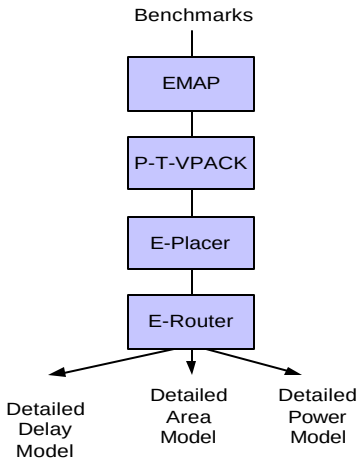
Cost Function

Initially, it is OK to overuse routing resources
- But this becomes more expensive as the algorithm runs

(McMurchie et al, U. Washington)

-> 2.7% Energy Improvement

Putting it all together:



Summary:

Technology Mapping:	7.6%
Clustering:	12.6%
Placement:	3.0%
Routing:	2.7%

Together, we got 22.6% reduction in energy.

This is with no modifications to the FPGA at all

- For the most part, these are orthogonal to the techniques you are seeing in the rest of the tutorial