Post-Placement Interconnect Entropy

Wenyi Feng, Jonathan Greene

Actel Corporation

SLIP ’06, Germany

Statement of Problem

- A PLD has logic cells, wires and switches
- The switches are controlled by configuration bits
  - **1-hot: one configuration bit for each switch**
  - **Binary encoding: log(n) bits for an n-input MUX**
- Configuration bits are costly in terms of area, but necessary for routing flexibility
- What is the minimum number of configuration bits per cell a PLD must have to route any reasonable, well-placed netlist?
- We obtain a tight lower bound

Outline

- Previous work and our approach
- Post-placement interconnect entropy
- Entropy properties
- Applications
- Summary

Previous Work on Configuration Bit Requirement

- [Dehon, FPGA'96]
  - **For a fully flexible programmable device, the number of configuration bits per cell must increase at a rate proportional to log(N), where N is the number cells in the device**
  - **Applicable to arbitrary netlists and placements**
  - **Pessimistic for practical netlists with a good placement**
- [Dehon, SLIP'01]
  - **Proposed "Tree-of-Meshes" hierarchical routing architecture**
  - **Used Rent's rule to characterize the bandwidth of bisection**
  - **Proved that a constant number of configuration bits per cell suffices for netlists of unlimited size**
- [Rubin & Dehon, FPGA'03]
  - **Proposed "Mesh-of-Trees" architecture that achieved constant configuration bits per cell**
  - **Number of required switches per LUT4 is up to 145**

Our Approach

- We take Rent's rule and the existence of good placements into account
- Let m be the total number of well-placed netlists
  - **A well-placed netlist is one with connection lengths distributed according to Rent's rule**
- A PLD with fewer than log(m) bits cannot accommodate all such netlists
- We derive bounds on log(m)
- We exhibit an architecture that achieves this bound
- We evaluate bounds for practical situations

Rent's Rule

- An empirical relationship between the size of a logic group and the number of its external connections:

$$T = T_0 N^{p}$$

- Is the base for many interconnect prediction techniques
  - **Average wire length [Donath, 1979]**
  - **Wire length distribution [Donath, 1981] [Stroobandt, 2001]**

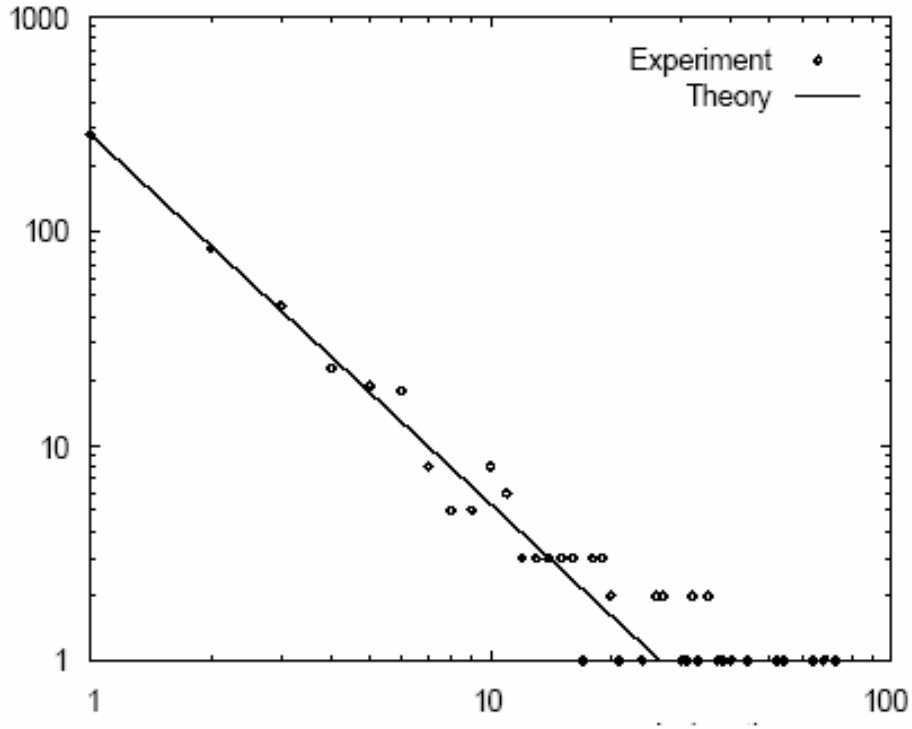
The Wire Length Distribution

- Rent's rule implies a power law distribution of wire length r (L: maximal possible length)

$$f(r) = gr^{Dp-(D+1)} \qquad (1 \le r \le L)$$

$$f(r) = 0 \qquad (r > L)$$

- D is the dimension in which cells are placed
  - **Typically 2, but could be larger**
- The distribution of source-sink connection lengths has a similar distribution (with slightly different scaling behavior)
  - **[Stroobandt, SLIP'01]**

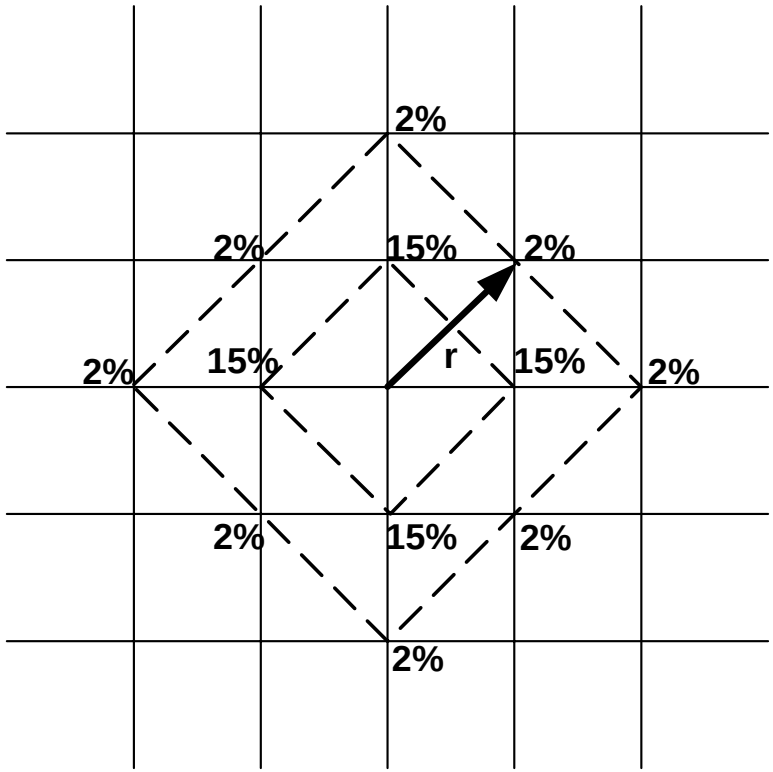


Log-log plot of number of Nets (vertical axis) vs. wire length (horizontal axis) [Stroobandt, 2001]

Probability Distribution of Source Pin Location for each Sink Pin

- RLOC: location of source pin relative to sink pin
- For a well-placed netlist, the RLOC length r obeys the power-law distribution f(r)
- Let C(r,D) be the number of locations at distance r in D dimensions
  - **Can be computed exactly; for example, C(r,2) = 4r**
- The probability of each location at distance r is f(r)/C(r,D)
- Example: for D = 2, p = 0.5
  - **r = 1: f(1) = 60%; probability of each of 4 locations = 15%**
  - **r = 2: f(2) = 16%; probability of each of 8 locations = 2%**

RLOC probability for D=2, p=0.5
(only shown for r=1 and 2 locations)



Entropy

- Entropy is a measure of uncertainty
- H(X) is the number of bits needed to describe a random variable X with distribution p(x)

$$H(X) = -\sum_{x} p(x) \log p(x)$$

H_{int}: Entropy of an RLOC

- Consider the relative source pin location (RLOC) as a random variable
- We can compute its entropy H_{int}

$$H_{\text{int}} = -\sum_{r=1}^{L}\sum_{i=1}^{C(r,D)} \frac{f(r)}{C(r,D)} \log \frac{f(r)}{C(r,D)} = -\sum_{r=1}^{L} f(r) \log \frac{f(r)}{C(r,D)}$$

- $-\sum_{r=1}^{L} f(r)\log f(r)$: is the number of bits to identify r;
- $\log C(r,D)$: is the number of bits to specify location.

Counting the Number of Well-Placed Netlists

- There are N cells, each with I inputs
- A combinatorial counting problem

$$m = \frac{NI!}{NIf(1)!*NIf(2)!*\cdots} * (C(1,D)^{NIf(1)} * C(2,D)^{NIf(2)} * \cdots)$$

- We can get tight bounds on log(m)

$$NIH_{\text{int}} - O(N^{1/D} \log N) \leq \log(m) \leq NIH_{\text{int}}$$

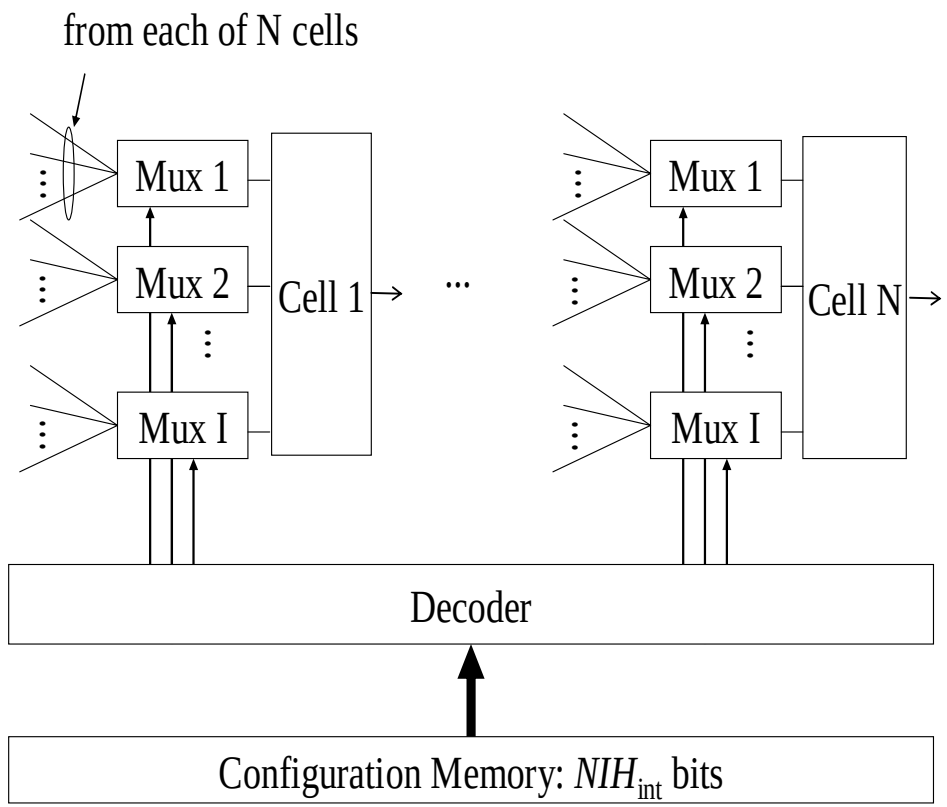
  - **See paper for details**
  - **On average, each input needs H_{int} bits**

Can We Go Below H_{int}?

- What if fewer than H_{int} configuration bits per input are provided?
  - **For N=131072, I=4, D=2, p=0.75, we compute H_{int} = 8.02**
  - **Suppose an interconnect provides only 8.0199 * NI configuration bits (just a tiny bit less)**
  - **Then the probability to route a randomly chosen well-placed netlist is smaller than $(0.5)^{131072 \times 4 \times 0.0001} = 1.65 \times 10^{-16}$**

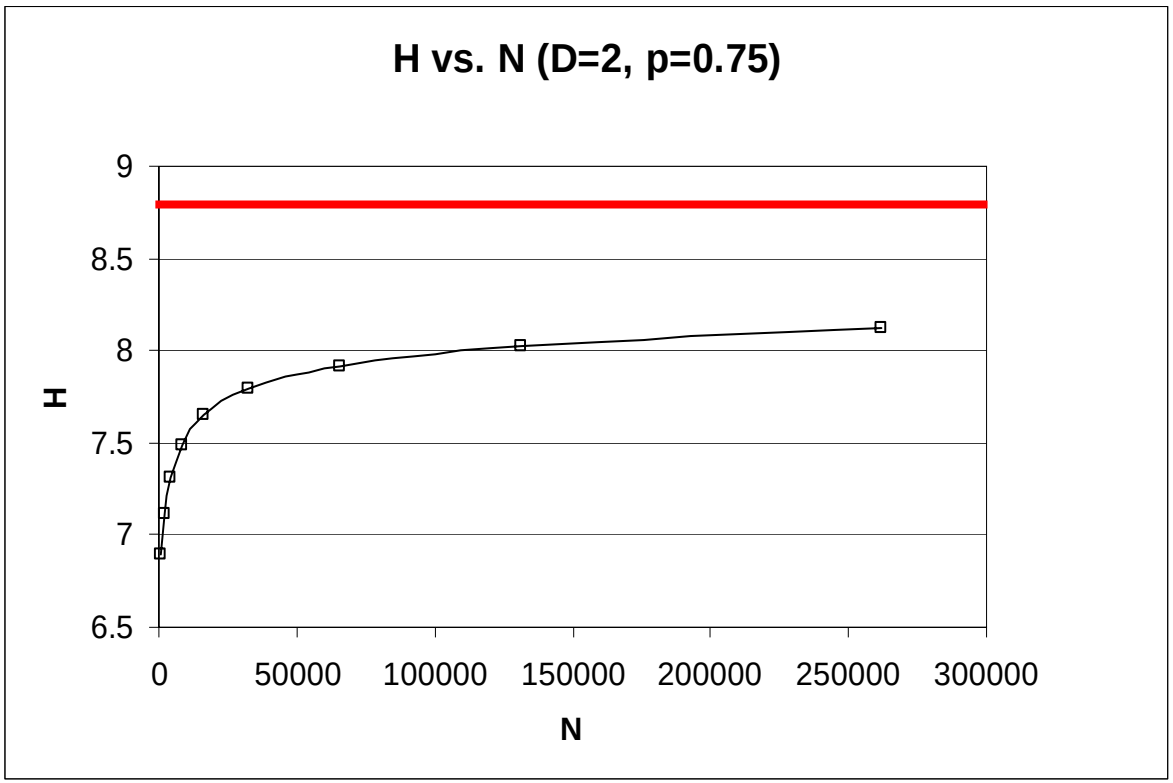
Achievability of Lower Bound

- Each cell has **I** input MUXes (one per input)
- A block decoder of NIH_{int} bits
- Each MUX selects from among the outputs of the N cells according to the control signal generated by the decoder
- Not a practical architecture, but it does prove achievability



$H_{int}(N, p, D)$ vs. N

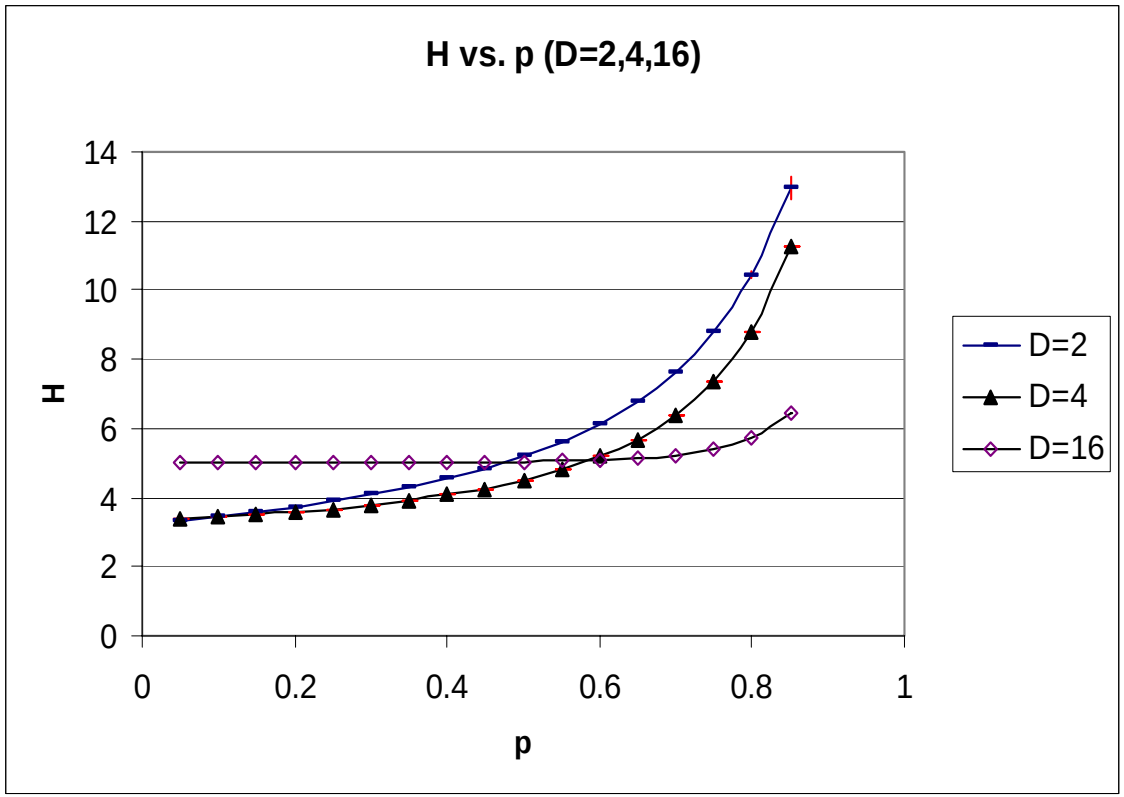
- N=number of cells; p=Rent exponent; D=placement dimension
- H_{int} converges as N→infinity
  - **A constant number of bits per cell suffices for infinitely big chips**



- D=2, p=0.75
- As N increases, H_{int} first sharply increases then flattens out
- H_{int} = 8.81 as N→ infinity (red line)

$H_{int}(p, D)$ vs. p for Large N

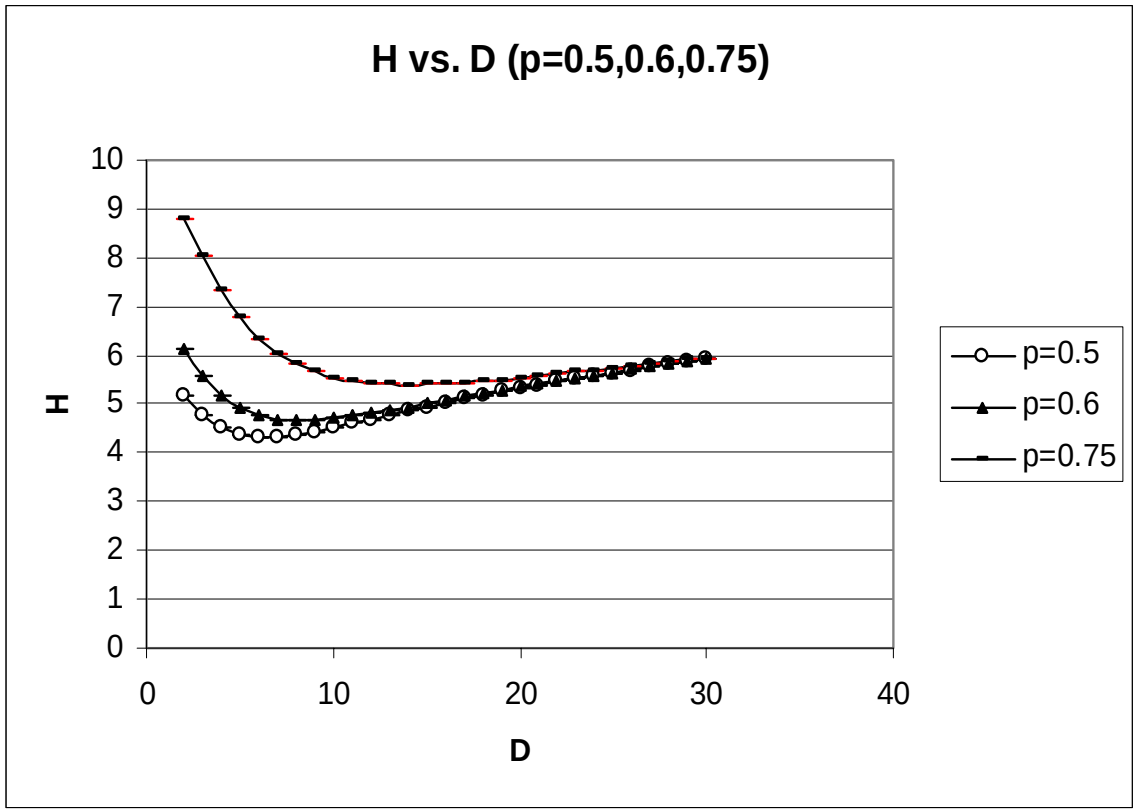
- H_{int} is a monotonically increasing function of p
- H_{int} goes to infinity as p→1



- D=2, 4, 16
- As p increases, H_{int} increases for all D
- The slope of increase is bigger for smaller D
- For D=16, the increase is unnoticeable until *p* reaches 0.7

$H_{int}(p, D)$ vs. D for Large N

- H_{int} has a minimal point vs. D
- The number of configuration bits per cell is minimized at a certain placement dimension



- P = 0.5, 0.6, 0.75
- As D increases, H_{int} first quickly drops, then reaches a minimum, then slowly increases
- All three H_{int} curves converge to 1+ log(D)
- For p=0.75, minimal H_{int} = 5.40 is reached when D=14

Computing the Bound for a Practical Case

- Practical example:
  - **Dimension D=2**
  - **Rent exponent p=0.75**
  - **Basic cell is a LUT4 (4-input look up table)**
- 31 configuration bits per cell for an infinite array
  - **$H_{int}(2, 0.75) = 8.81$**
  - **Per LUT = 4 * 8.81 = 35.2**
  - **Adjusted by LUT input swappability –log(4!)=-4.6**
  - **35.2 – 4.6 = 30.6 ~= 31**
- 27 configuration bits per cell for 65K cells
  - **$H_{int}(65K, 2, 0.75) = 7.91$**
  - **4*7.91 – 4.6 = 27**

How Close Does a Practical Architecture Get to the Bound?

- VPR-type architecture: k=4, N=6, I=14, L_{wire}=4, W=48, F_s=3, F_c=0.5, F_{out}=0.17, F_{cint}=1, F_{cfb} = 1 (Lemieux & Lewis 2004)

Signal	Quantity/Cluster	Number of inputs
LUT input	$kN = 24$	$F_{cint}I+F_{cfb}N = 20$
Cluster input	$I = 14$	$F_cW = 24$
Routing track	$2W/L_{wire} = 24$	$2F_s+L_{wire}NF_{out}/2 = 8$

  - **For each MUX with n inputs, we assume the minimum number of configuration bits, log(n)**
- Total number of configuration bits: 40 per LUT4
- This is 30% higher than 31 bits per LUT4 (for infinite array), and 48% higher than 27 (for up to 65K)

A “Practical” Application...

- In a desperate effort to meet cost goals, the manager of an FPGA startup suggests aggressively depopulating the switches in their VPR-like architecture.
- The architect carefully designs a depopulation scheme.
  - **Total number of configuration bits per BLE drops from 40 to 26.6**
- The router expert runs sample designs, and the routability stinks!
  - **He complains to the architect about the poor connectivity!!**
  - **The architect complains to the router expert about his bad algorithm!!!**
- After reading our paper, the architect and router expert tell their manager that his idea could not possibly have worked.
  - **Regardless of detailed connectivity of design**
  - **Regardless of routing algorithm**
- The startup pursues another approach, and later has a successful IPO.

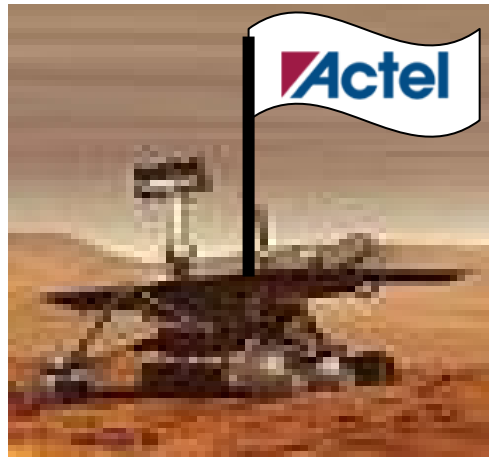
High Dimensional Programmable Interconnect

- Higher dimensional interconnect topology has been proposed
  - **Mitigates the growth of average connection length**
  - **Physical implementation limited to 2-D (or perhaps 3-D)**
    - **[Alexander and etc., ASIC'95]**
  - **Can also embed high dimensional routing in lower dimensional chips**
    - **[Schmit, FPL'03]**
    - **[Matsumoto and Masaki, IEICE'05]**
- Our results show the number of configuration bits required can be reduced in higher dimensions
- Example (Rent's exponent 0.75):
  - **2-D: H_{int} = 8.81**
  - **3-D: H_{int} = 8.02 (9% smaller)**
  - **4-D: H_{int} = 7.33 (17% smaller)**
  - **14-D: H_{int} = 5.40 (39% smaller)**

A 14-D Digression...

- Consider sending a netlist from Earth to Mars
  - **Netlist of N LUT4 cells**
  - **Per bit transmission cost is extremely high**
- Question: what is the number of bits needed for the transmission?
  - **One trivial answer is (16 + 4*logN) * N**
  - **We can do better: (16 + 17) * N = 33N**
    - **Use 14-D placement**
    - **Encoder is 14-D placer + RLOC encoding**

Corporate Headquarters – Mountain View, CA.



Summary of Our Work

- Introduced post-placement interconnect entropy: H_{int}
- Used it to bound the number of configuration bits required in a way that accounts for Rent's rule and good placements
- Studied H_{int} behavior vs. chip size, Rent exponent, and placement dimension
- Evaluated bound for practical situations
- Compared with VPR-type architecture
- Open problem: create a practical architecture that achieves the lower bound

Thank you!