Glitch Minimization Approaches in FPGAs

Jérémie Dumas

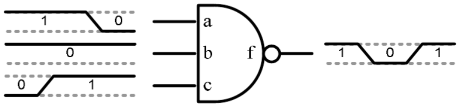
École Normale Supérieure de Lyon

December 2011

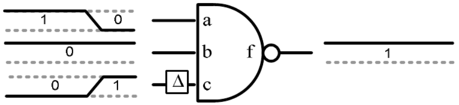
Motivations

- Performance driven algorithms for FPGAs
- Need for power-efficient methods : alternative goal
- Embeded technology, increased autonomy, green computing . . .
- Noise reduction

Definitions



(a) Original circuit with glitch



(b) Glitch removed by delaying input c

- Technology mapping, placement and routing
- Static power versus dynamic power (62% of total power)
- Functional transitions versus spurious transitions (glitches)
- *Intertial delay* of 0.2*ns* : smaller glitches are filtered

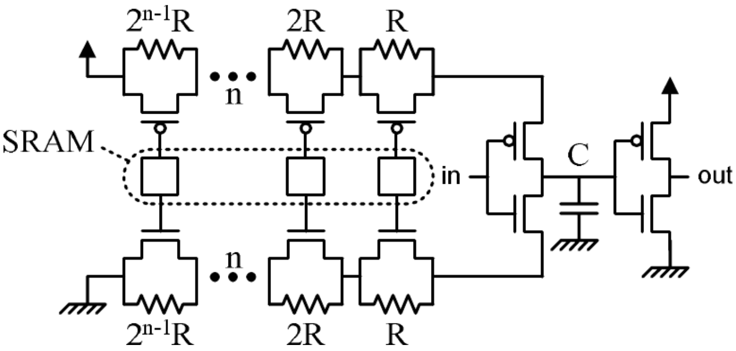
Approaches

GlitchMap, GlitchReroute, GlitchLess

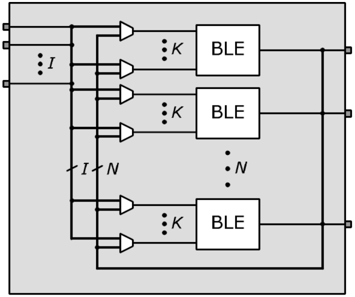
- Optimizing the technology mapping (GlitchMap)
- Optimizing the routing (GlitchReroute)
- Physically inserting new elements (GlitchLess)
- Blend it together

Here : GlitchLess and GlitchReroute. Takes place after routing.
Does not alter the length of the critical path.

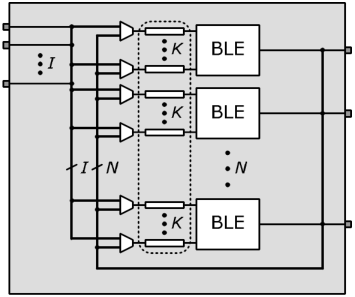
The Programmable Delay Element



Configuration Models



(a) Original VPR Logic Block



(b) Scheme 1: LUT inputs.

- *Needed_Delay*(n, f) for each input signal f of node n
- Setting the *Added_Delay*(n, f) using multiples of $i \cdot \tau + \mathrm{cte}$

Adding Delay Elements

Experimentation Protocol

- HSPICE simulation software
- Versatile Place and Route (VPR) used for routing
- Parameters prior to configuration : *min_in*, *max_in*, *num_in* . . .
- Calibrations done with ideal parameters
- Experiments using LUT with $K = 4, 5, 6$ inputs

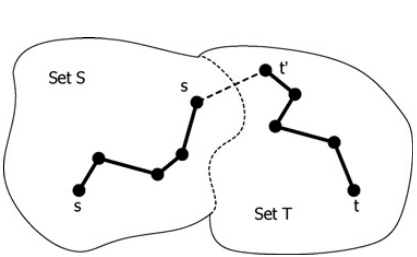
Path Balancing
A Rerouting Heuristic

- Graph with delays, *Rip-up and re-route* like VPR
- Target delay in range $d \pm \Delta$, with $\Delta = \frac{1}{2} \cdot \mathrm{intertial\_delay}$
- A heuristic for selecting a pair (s, t) to reroute
- Critical path remain unchanged
- Considers essentially first-level clusters

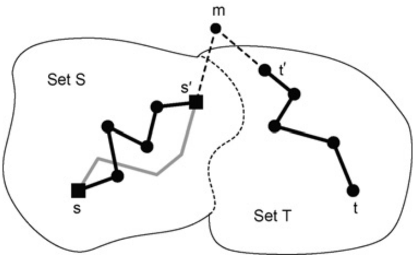
Selection heuristic

- Favors paths with more influence on the rest of the DAG
- Defavors paths that need great readjustment

Source to Sink Rerouting



(a) Single-edge path



(b) Two-edge and recursive

- Looking for simple paths (without loop). Heuristic approach.
- Uses Dijkstra's algorithm. Find paths $s \rightsquigarrow s' - t' \rightsquigarrow t$.
- Split the graph in two sets S and T.
- Two-edges and recursive extensions.

Some Numbers

Average Reductions

- Power savings : $\approx$ 18% for GlitchLess and $\approx$ 11% for GlitchReroute.
- Glitching activity : -91% for GlitchLess and -27% for GlitchReroute.

Ideal dynamic power saving of 22.6%.

Downsides

- Power overhead, common in both scheme.
- Area overhead for GlitchLess ($\approx 5\%$).
- Delay overhead for GlitchLess (from $\approx 0.2\%$ to $\approx 2\%$).

Moral of the Story

Dynamic power reduction techniques

- During the technology mapping, the routing process, or afterwards.
- Mapping : cut-enumeration based techniques. Low-power clustering.
- Problems are NP-Hard.

Experimental protocol

- Simulation, models for *signal propagation* and *power consumption*.
- Estimating the switching activity with a probabilist approach.
- Multi-dimensional calibration by varying one parameter at a time.

Thank you for your attention.
Feel free to ask your questions.