Optimization in VLSI Design.

Cycle Time and Slack Optimization with Parametric Shortest Paths.

Jens Egeblad

October 7th, 2004
Overview

- Cycle Time Optimization (Overview)
- Single Source Shortest Paths (Recap)
- Parametric Shortest Paths (PSP)
- Algorithm for PSP
- Efficient Implementation
- Applications of PSP
- Latches and Flip-Flops
- LP-Formulation
- Computing the Optimal Cycle Time
- Balancing Slacks
- Summary
**CPU-Clock (recap)**

The chip has storage circuits (modules). These are:

- Latches, Flip-Flops, Registers, Memory Banks etc.

The chip also has combinational logic circuits. E.g.:

- and-, or-gates, adders, ALUs etc.

The input to combinational logic comes from storage units. The output is stored in storage units.

![Diagram of combinational logic](image)

Storage units have a clock-signal as input.

![Clock-Signal](image)

To ensure that storage units are stable they only store their input signal when the clock is high (latches) or at the clock edge (flip-flops). 
(S\(_1\) should not change before we have a new valid input at \(S_2\))
Cycle-Time Optimization (Overview)

Cycle-Time optimization is to minimize the length of each period in the clock-signal such that the signals through all combinational logics are stable.

\[ t_{12} \]

(Here we cannot have a clock period less than \( t_{12} \))

The smaller cycle time the faster will signals propagate through the chip. As anyone knows higher frequency means faster CPU.

This problem can be formulated as a linear program (LP).

If this formulation has a special form we can solve the problem as a Parametric Shortest Paths problem.

Before we consider the actual formulation of the LP we will consider the Parametric Shortest Paths problem.
Single Source Shortest Paths (Recap)

The well-known Single Source Shortest Path problem:

Given a $G = (V, E)$ with vertices $V$, edges $E$ and edge costs $c(e)$ for $e \in E$ find the shortest path from some vertex $s \in V$ to every other vertex in $V$.

(The Shortest Path Tree)
Two different algorithms (Recap)

**Dijkstra:**

1. Set the distances $d(s) = 0$ and $d(v) = \infty$, $v \in V$
2. Let $Q$ be a priority-queue over $d(v)$.
3. Repeat step 4-6 until $Q$ is empty.
4. Extract vertex $u$ with minimum distance from $Q$.
5. Visit neighbors $v$ of $u$:
6. if $d(v) > d(u) + c(u, v)$ set $d(v) = d(u) + c(u, v)$

**Running time:** $O(|V|^2)$ or $O(|V| \log |V| + |E|)$ w. Fibonacci.

**Bellman-Ford (Graphs with negative weights):**

1. Set the distances $d(s) = 0$ and $d(v) = \infty$, $v \in V$
2. Do step 3-4 $|V| - 1$ times.
3. for $(u, v) \in E$ do
4. if $d(v) > d(u) + c(u, v)$ set $d(v) = d(u) + c(u, v)$.
5. For $(u, v) \in E$ do
6. if $d(v) > d(u) + c(u, v)$ then
7. "There is negative weight cycle!"

**Running time:** $O(|V||E|)$. 
Bellman-Ford Example

... 3 more steps but no changes
**Parametric Shortest Paths**

Graph $G = (V, E)$. Edges have costs $c'(e) = c(e) - \lambda$ for $e \in E' \subseteq E$ and $c'(e) = c(e)$ for all others, where the parameter $\lambda$ is some value. The edges $E'$ are called *parametric edges*.

![Diagram of a graph with parametric shortest paths](image)

**The Problem**: Given some vertex $s$, find single source shortest path trees $T_\lambda$ for all values of $\lambda \in ]-\infty, \lambda_{\text{max}}]$. $\lambda_{\text{max}}$ is the maximal value of $\lambda$ such that there is no is negative weight cycle.

**Output**: A sequence of values:

$$\lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{\text{max}}$$

and sequence of trees:

$$T_0, T_1, \ldots, T_{\text{max}}$$

such that $T_i$ is a shortest path tree for all $\lambda \in [\lambda_i, \lambda_i + 1]$. 

7
Parametric Shortest Paths Example

Example:

\[ \lambda > -\infty \]

\[ \lambda > 1 \]

\[ \lambda > 3 \]

\[ \lambda > 4.5 \]
PSP - Solution Method

To find $T_0$: set $\lambda_0 = -\sum_{c \in E} |c(e)|$. This way we will choose a path from $s$ to $v \in V$ with no parametric edges if there is one. Now find the shortest path in $G$ for $\lambda = \lambda_0$ with e.g. Bellman-Form ($O(|V||E|)$).

To find $T_{i+1}$: We start from $T_i$ (the current tree). Now let:

- $p(e)$ be the path from $s$ to $u$ in $T_i$ plus edge $e = (u, v) \in E$.
- $t_v$ be the path from $s$ to $v \in V$ in the current tree $T$.
- $\delta(p(e))$ and $\delta(t_v)$ be the number of parameterized edges along path $p(e)$ and $t_v$, respectively'.

For all edges $e \in E$ store the key:

$$k(e) = \frac{c(p(e)) - c(t_v)}{\delta(p(e)) - \delta(t_v)},$$

for $\delta(p(e)) - \delta(t_v) > 0$ and $k(e) = \infty$ otherwise.

To find the next tree we look the smallest $\lambda_{i+1} \geq \lambda_i$ such that for some edge $e$ and vertex $v$: $c'(p(e)) \leq c'(t_v)$. I.e. we look for the $e$ with smallest $k(e)$ and $\lambda_{i+1} = k(e)$. 

9
PSP - Solution Method (continued)

Given this edge \((u, v)\). Remove the current unique edge \((w, v) \in T_i\) and replace it by \((u, v)\) to get \(T_{i+1}\). Update all keys.

Calculating the key:

\[ k(e) = \frac{c(p(e)) - c(t_v)}{\delta(p(e)) - \delta(t_v)}, \]

can be done in constant time if we know \(c(t_u), c(t_v), \delta(t_u), \delta(t_v)\). So these values can be calculated while traversing \(T_{i+1}\). Only \(E\) below \((u, v)\) in \(T_{i+1}\) needs to be updated.

Running Time (If we use heap-structure):

- Finding the minimal \(k(e)\) takes \(O(\log |V|)\) time.
- For each edge \(e\) we update \(k(e)\) at most \(2|V|\) times, since the number of parameterized edges from \(s\) to \(v\) increases with each tree and there cannot be more than \(|V| - 1\) edges in a shortest tree path.
- The total number of edge updates is at most \(2|V||E|\). Each update takes \(O(\log |V|)\) time.
- **Total running time:** \(O(|V||E| \log |V| + |V|^2 \log |V|) = O(|V||E| \log |V|)\).

**With Fibonacci-heaps:** \(O(|V||E| + |V|^2 \log |V|)\).
PSP - Applications

The Minimum Mean Cycle Problem. Find a directed cycle in the graph that minimizes the average cost of the edges.

The Minimum Balance Problem. Given a strongly connected graph $G$, find a potential such that all subsets of a $V$ except $E$ and $\emptyset$ are minimum balanced.

A subset of $V' \subseteq V$ is minimum balanced if the minimum cost among edges entering $V'$ is equal to the minimum cost among edges leaving $V'$.

A potential is a value $\pi(v)$ assigned to all vertices $v \in V$, such that the modified edge costs $c'(u, v) = c(u, v) + \pi(u) - \pi(v)$.

We will look at a form of this in a moment.
Back to Cycle Time Optimization

A closer look at storage elements:

**D-Latch:**

```
  C
  +---+---+
  |   |   |
  +---+---+
    D
```

**Flip-Flop:**

```
  C
  +---+---+
  |   |   |
  +---+---+
    D
```

(Based on flip-flips and latches we can build anything...)

**Now let:** $S$ be set of storage elements.

Let the clock input at $s \in S$ be 1 from time $a_s$ to $b_s$, $a_s + T$, $b_s + T$, $a_s + 2T$ to $b_s + 2T$ etc. for $n \in \mathbb{Z}_+$.  

When the clock input is 1, the stored bit in $s$ changes.  
When the clock input is 0, the stored bit is unchanged.  

We can read the value of $s$ at any time.
Cycle Time formulation (Shifting)

For each storage element we may **shift** the clock input by some value $y_s$. So the clock input has value 1 in intervals:

$$[a_s + y_s, b_s + y_s], [a_s + y_s + T, b_s + y_s + T], \ldots$$

![Diagram of clock signal and shifted clock signal]

**Note:** $y_s$ is a variable that we use to shift the clock-signal. This way the cycle period can be shorter than the maximum delay between storage elements.

Shifting times have to be realized by a clock-tree. A clock-tree routes the the clock signal through the chip. We also impose **lower** and **upper bounds** on shifting times:

$$l_s \leq y_s \leq u_s$$
Cycle Time formulation (Valid Time)

For each storage element we let \( x_s \) denote the time the data signal at the input is valid. Of course to use the data signal we must have:

\[
a_s + y_s \leq x_s \leq b_s + y_s
\]

and the signal must be valid in the interval \([x_s, b_s + y_s] \), otherwise we would not store a valid signal in \( s \).
Cycle Time formulation (arrival time)

Some of the latches may be transparent. E.g. a data signal can go straight through it (This will be more apparent in a minute).

Now consider a signal which departs from storage unit $s$, arrives to storage unit $v$ at $x_v$ passes through $v$ and arrives at $w$ at $x_w$:

Let $t_{vw}^{max}$ be the longest time it takes for a signal to get from $v$ to $w$ and $t_{vw}^{min}$ be the shortest, and let $\zeta_{vw} = 0$ if the signal starting at $s$ is supposed to arrive at $w$ in this cycle and 1 if in the next cycle.

since $x_w$ must wait for the signal from $s$:

$$x_v + t_{vw}^{max} \leq x_w + \zeta_{vw}T.$$ 

and to avoid receiving a signal from $s$ before the last cycle is over:

$$a_v + y_v + t_{vw}^{min} \geq b_w + y_w + (\zeta_{vw} - 1)T.$$
Cycle Time formulation (Reformulation)

To summarize we have four different constraints:

Shift Bounds: \( l_s \leq y_s \leq u_s \)

Valid time: \( a_s + y_s \leq x_s \leq b_s + y_s \)

Arrival time (1): \( x_v + t_{vw}^{max} \leq x_w + \zeta_{vw}T \)

Arrival time (2): \( a_v + y_v + t_{vw}^{min} \geq b_w + y_w + (\zeta_{vw} - 1)T \)

Given above variables and constraints we wish to minimize \( T \).

Set \( \lambda = -T \).

All four constraints can be rewritten either of the two forms:

\[
\begin{align*}
    z_i + c_{ij} & \geq z_j \\
    z_i + c_{ij} - \lambda & \geq z_j 
\end{align*}
\]

If we add artificial variable \( z_0 = 0 \):

Shift Bounds: \( y_s - l_s \geq z_0, \quad z_0 + u_s \geq y_s \),

Valid time: \( x_s - a_s \geq y_s, \quad y_s + b_s \geq x_s \)

Arrival time (1): \( x_w - t_{vw}^{max} \geq x_v \\
\quad x_w - t_{vw}^{max} - \lambda \geq x_v \)

Arrival time (2): \( y_v + a_v + t_{vw}^{min} - b_w - \lambda \geq y_w \\
\quad y_v + a_v + t_{vw}^{min} - b_w \geq y_w \)
Converting to a graph.

We now consider the problem:

\[
\begin{align*}
\text{Maximize} & \quad \lambda \\
z_i + c_{ij} & \geq z_j \\
z_i + c_{ij} - \lambda & \geq z_j
\end{align*}
\]

We use this to create a graph \( G = (V, E) \):

- For each variables \( z_i \) we create a vertex \( v_i \).
- For each constraint we add an edge \( e \) with cost \( c_{ij} \) or \( c_{ij} - \lambda \).

The solution to our problem is determining \( \lambda_{\text{max}} \) from the Parameterized Shortest Path Problem associated with \( G \). But why?
Example:

While you think about it, consider the following example:

A *master-slave latch* is two connected latches. One will allow signals to pass through when clock signal is 1, the other when clock signal is 0.

This system consist of four storage units.

The signal from 1 must pass through 2 and be available at 3 in the next clock cycle.

Let us draw the graph using the four different constraint types:

- **Shift Bounds**: \( y_s - l_s \geq z_0, \quad z_0 + u_s \geq y_s \),
- **Valid time**: \( x_s - a_s \geq y_s, \quad y_s + b_s \geq x_s \),
- **Arrival time (1)**: \( x_w - t_{vw}^{\text{max}} \geq x_v \), \( x_w - t_{vw}^{\text{max}} - \lambda \geq x_v \),
- **Arrival time (2)**: \( y_v + a_v + t_{vw}^{\text{min}} - b_w - \lambda \geq y_w \), \( y_v + a_v + t_{vw}^{\text{min}} - b_w \geq y_w \)
Solved by Parametric Shortest Paths

Ok, so why is this problem solved by PSP solution?

Because for any vertex $v_i$, we have $z_i \leq z_j + c_{ij}$ for any edge $(v_j, v_i)$.

Now if $z_j$ is the distance to vertex $v_j$ ($z_0 = 0$), then $z_i$ must be less or equal to the shortest distance to $v_i$.

So we can solve the problem in $O(|V||E| + |V|^2 \log |V|)$ time.
Data Path Slack Balancing (Formulation)

We may allow some slack on data paths. I.e.: since \( x_w \) must wait for the signal from \( s \):

\[
x_v + t_{vw}^{\max} + \epsilon_{vw}^{\max} \leq x_w + \zeta_{vw}T.
\]

and to avoid receiving a signal from \( s \) before the last cycle is over:

\[
a_v + y_v + t_{vw}^{\min} - \epsilon_{vw}^{\min} \geq b_w + y_w + (\zeta_{vw} - 1)T.
\]

for some slack variables \( \epsilon_{vw}^{\min}, \epsilon_{vw}^{\max} \).

Now the problem is to maximize \( \epsilon_{vw}^{\min}, \epsilon_{vw}^{\max} \) for as many datapaths as possible. Or:

**Slack Balancing Problem:** Find solution vector of \( \epsilon \)-values:

\[
\mathcal{E} = (\epsilon_1, \ldots, \epsilon_k), \quad (\epsilon_i \leq \epsilon_j \text{ for } i < j)
\]

such that for any other solution to the slack constraints

\[
\mathcal{E}' = (\epsilon_1', \ldots, \epsilon_k') \quad (\epsilon_i \leq \epsilon_j \text{ for } i < j)
\]

we have \( \epsilon_j < \epsilon_j' \) for some \( 1 \leq j \leq k \) and \( \epsilon_i \leq \epsilon_j \) for all \( 1 \leq i < j \).
Data Path Slack Balancing (Solution)

We can use our PSP solution method as before but fixing the clock period.

We start with the single source shortest path tree for $\lambda_{max}$ with all slacks $\epsilon_i = 0$.

When we increase $\lambda > \lambda_{max}$ the graph will contain negative weight cycles.

To solve this problem all $\epsilon_i$ on their corresponding edges will be set to $\lambda - \lambda_{max}$.

Then we contract the negative weight cycle $C$ to a single vertex $z$.

Let $w \in C$ be vertex closest to root of $T_k$ (when we have reached $\lambda_k$).

Then for each leaving edge $(v, u)$, $v \in C$, the edge $(z, u)$ will have cost $c(P(w, v)) + c(v, u)$.

...and for each entering edge $(v, u)$, $v \in C$, the edge $(z, u)$ will have cost $-c(P(w, v)) + c(u, v)$.

In total an $O(|V||E| + |V|^2 \log |V|)$ running time.
Clock Tree Slack Balancing

But we can do more...

Exact clock signal arrival-times cannot be predicted and therefore not realized. To solve this problem we may have an interval of clock-shifts:

\[ [y_s - \epsilon_s, y_s + \epsilon_s] \]

Now the constraints must be met for all values \( y'_s \in [y_s - \epsilon_s, y_s + \epsilon_s] \).

Once again this is a balancing problem this time for inequalities:

\[
\begin{align*}
    a_s + y_s + \epsilon_s &\leq x_s \leq b_s + y_s - \epsilon_s \\
    a_V + y_v - \epsilon_v + t_{vw}^{\text{min}} &\geq b_w + y_w + \epsilon_w + (\zeta_{vw} - 1)T
\end{align*}
\]

However there are two different \( \epsilon \) in each inequality so they do not have the form \( z_i + c_{ij} - \lambda < z_j \)

However, this can be modelled by adding additional variables.

In fact both data path and clock tree slack can be balanced at the same time by combining inequalities and adding variables.

**Running time:** \( O(|E||V| + |V|^2 \log |V|) \).
Results (from the Article)

Considers chip PU:

<table>
<thead>
<tr>
<th>Chip</th>
<th>Cycle period</th>
<th>modules</th>
<th>nets</th>
<th>pins</th>
<th>latches</th>
<th>data-paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
<td>6.5 ns</td>
<td>164,056</td>
<td>171,666</td>
<td>591,410</td>
<td>17,265</td>
<td>2,670,459</td>
</tr>
</tbody>
</table>

19.45 seconds to optimize cycle time.
Summary

We have considered the cycle time optimization problem.

- Storage units and clock signals.
- Shortest Paths.
- Parametric Shortest Paths.
- Solution method for Parametric Shortest Paths in $O(|V| |E| + |V|^2 \log |V|)$ time.
- Formulation of LP for cycle time optimization.
- Solved the LP in efficient time by converting to PSP problem.
- Used the PSP solution method to balance slacks on data and clock paths.
- ... And we saw 164,056 modules.