Formulating the Problem of Register Slice Optimization for AMBA 3 AXI Bus

Jianfeng Liu, SungJoo Yoo, Byeong Min, KyuMyung Choi, SooKwan Eo
CAE Team, System LSI
Samsung Electronics Co LTD
Oct. 03, 2007
# Table of Contents

- Introduction
- Problem Statement
- Our Solution
- A Case Study
- Conclusions
Targeting the AXI Bus Timing Closure

- Timing closure is always a headache in modern SoC design
- For high frequency bus design, Register Slice is an elegant solution
Principles of Register Slice Insertion

- The 5 AXI channels of Register Slice operate independently with each other.
- The AXI traffic signals of each AXI channel of the Register Slice can be categorized into two groups:
  - Ready signal
  - Valid and Payload signals
The 3 Modes of the Register Slice

For each AXI channel

- The Register Slice has 3 modes of operation, which provides timing isolation for different AXI traffic signals
- The AXI traffic signals are multiplexed to the different processing units according to the specified mode parameter
- The area of the fully isolation mode processing unit is about twice of that of the forward mode, and the bypass mode has almost no area penalty

<table>
<thead>
<tr>
<th>AXI Traffic Signals</th>
<th>3 Modes of Register Slice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid &amp; PAYLOAD</td>
<td>BYPASS</td>
</tr>
<tr>
<td></td>
<td>X</td>
</tr>
<tr>
<td>READY</td>
<td>X</td>
</tr>
</tbody>
</table>
Problem Statement

- Introduction
- Problem Statement
- Our Solution
- A Case Study
- Conclusions
Problems in Register Slice Insertion

- Each AXI channel comprises of more than a hundred of signals, which makes the STA timing report analysis a nightmare.
- The design of AXI bus PL300 comprises of many kinds of combinational paths, which make the usage of register slice correlated with each other.
- There are many positions for inserting Register Slice in a complex SoC design. Roughly speaking, if there are $P$ possible Register Slices, the possible Register Slice configuration is $5 \times 3^P$, which is definitely unfeasible for human exploration for a reasonable complex SoC design, e.g. $P>10$. 

Confidential
The Traditional Design Flow

- Difficult in making decisions
  - The STA report is difficult to read
  - It’s difficult to predict the influence of Register Slice insertion
- Time consuming: normally a couple of iterations are needed to obtain a satisfactory solution.
- Optimality: there is no clue that the solution is an optimal one, and that how far it is from the optimal solution

---

**Design Spec.**

- **AMBA Designer**
  - 3 h
- **Design Compiler**
  - 28 h
  - (Syn., Scan Ins, Net. Merging)
- **STA**
  - 3 h

**Timing Closure**

- Yes
- No

**Determine Register Slice Configuration Using PrimeTime STA Report**

N (2~5) times

2 days?
Register Slices Tend to Be Over-Used in Practice

- Tight Time-To-Market pressure drives the timing closure engineers hasty to make an over-designed solution
- In one of our example SoC design, Register Slice accounts for
  - 40% of the total bus gate counts
  - 4 cycles latency in memory access for most IPs
Problem Statement

- Introduction
- Problem Statement
- Our Solution
- A Case Study
- Conclusions
The New Design Flow

- AXI-aware path grouping and visualization have been proposed to aid STA report analysis.
- An accurate RS insertion gain estimator has been introduced to predict the effect of RS insertion to timing slack.
- Normally, no iteration is needed to determine the register slice configurations.
- The optimality of register slice configuration is guaranteed by the accurate RS insertion gain estimator and the optimization procedure.
Design Flow Comparison

<table>
<thead>
<tr>
<th></th>
<th>Traditional RS ins. Flow</th>
<th>New RS ins. Flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficulty in determining RS configurations</td>
<td>Very Difficult</td>
<td>Easy</td>
</tr>
<tr>
<td>Design Cycle Time</td>
<td>10+ days (N=3)</td>
<td>3~4 days</td>
</tr>
<tr>
<td>Quality of Solution</td>
<td>Far from Optimal</td>
<td>Near-Optimal</td>
</tr>
</tbody>
</table>

Confidential 12
AXI-aware Timing Analysis (I)

AXI is rich in combinational paths

The Signal Traffics of Address Channel

ARM Developers’ Conference & Design Pavilion 2007
### AXI-aware Timing Analysis (II)

A table has been made to visualize the Worst Negative Slack of ALL the combinational path groups related to AXI bus.

| AR   | M0   | M1   | M3   | M4   | M5   | M6   | M7   | M8   | M9   | M10  | M11  | S0   | S1   | S2   | S3   | FF   |
|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| M0   | -1.1 | -1.8 | -1.2 | -1.6 | -1.8 | -1.5 | 0.06 | 0.07 | 2.79 | 2.76 | 2.92 | -0.3 | 1.16 | 1.27 | 1.4  | 2.2  | 1.79 |
| M1   | -1   | -1.7 | -1.6 | -1.5 | -1.7 | -0.5 | 0.07 | 0.19 | 2.87 | 2.83 | 3.01 | -0.2 | 1.1  | 1.34 | 1.52 | 2.26 | 1.86 |
| M2   | -1.1 | -1.7 | -1.2 | -1.6 | -1.8 | -1.5 | -0   | 0.08 | 2.88 | 2.72 | 2.9  | -0.2 | 1.12 | 1.19 | 1.32 | 2.33 | 1.79 |
| M3   | -0.9 | -1.6 | -1.1 | -1.5 | -1.7 | -1.3 | 0.18 | 0.25 | 2.96 | 2.88 | 3.09 | -0.1 | 1.49 | 1.39 | 1.5  | 2.41 | 1.99 |
| M4   | -1.1 | -1.8 | -1.3 | -1.6 | -1.9 | -0.6 | 0    | 0.04 | 2.72 | 2.75 | 2.86 | -0.4 | 1.23 | 1.16 | 1.46 | 2.2  | 1.76 |
| M5   | -1   | -1.8 | -1.3 | -1.7 | -1.8 | -1.5 | 0.0  | 0.07 | 2.76 | 2.74 | 2.89 | -0.4 | 1.35 | 1.24 | 1.43 | 2.24 | 1.79 |
| M6   | -1   | -1.7 | -1.2 | -1.6 | -1.8 | -1.5 | 0.05 | 0.08 | 2.77 | 2.79 | 2.9  | -0.3 | 1.44 | 1.34 | 1.41 | 2.17 | 1.81 |
| M7   | -1   | -1.7 | -1.2 | -1.6 | -1.8 | -1.5 | -0   | 0.11 | 2.81 | 2.75 | 2.93 | -0.3 | 1.18 | 1.27 | 1.46 | 2.2  | 1.82 |
| M8   | -3.2 | -3   | -3.4 | -3.8 | -3.7 | -2.1 | -2.1 | 0.57 | 0.53 | 0.71 | -2.5 | -0.9 | -1   | -0.8 | 0.06 | -0.4 |
| M9   | -3.4 | -4.1 | -3.5 | -3.9 | -4.2 | -3.8 | -2.3 | -2.2 | 0.49 | 0.36 | 0.61 | -2.6 | -1   | -1.1 | -1   | 0.15 | -0.5 |
| M10  | -2.8 | -3.5 | -3   | -3.4 | -3.6 | -3.3 | -1.7 | -1.7 | 0.98 | 0.93 | 1.13 | -2.1 | -0.4 | -0.6 | -0.4 | 0.67 | 0.04 |
| M11  | -1.1 | -1.7 | -1.2 | -1.6 | -1.8 | -1.5 | 0.02 | 0.06 | 2.76 | 2.74 | 2.9  | -0.3 | 1.43 | 1.35 | 1.36 | 2.23 | 1.8  |
| S0   | 0.95 | 0.22 | 0.48 | 0.46 | -0.2 | -0.1 | 1.39 | 1.44 | 4.54 | 4.54 | 4.94 | 1.52 | 3.18 | 3.3  | 3.16 | 3.56 | 1.8  |
| S1   | 0.6  | -0.2 | 0.54 | -0.1 | 0.01 | 0.16 | 1.72 | 1.72 | 4.27 | 4.23 | 4.41 | 1.16 | 3.3  | 3.3  | 3.16 | 3.56 | 1.8  |
| S2   | 0.39 | -0.4 | 0.07 | -0.3 | -0.2 | 0    | 1.62 | 1.61 | 4.32 | 4.3  | 4.45 | 1.28 | 3.3  | 3.3  | 3.16 | 3.56 | 1.8  |
| S3   | 0.92 | 0.18 | 0.66 | 0.29 | 0.54 | 0.65 | 2.22 | 2.34 | 4.52 | 4.94 | 4.65 | 1.43 | 3.3  | 3.3  | 3.16 | 3.56 | 1.8  |
| FF   | -0   | -0.7 | -0.2 | -0.6 | -0.9 | -0.8 | 1.01 | 1.11 | 3.8  | 3.68 | 3.93 | 0.7  | 2.15 | 2.19 | 2.32 | 3.26 | X   |
The Practical Issue in Register Slice

- There are some logic controls paths which consume some time delay
- As the analysis is based on the signal groups (Valid and payload signals), the critical timing path can switch from one to another after Register Slice insertion
Register Slice Insertion Gain Estimator

- Inputs are the STA analysis reports of the 4 kinds of RS configurations
  - All the RS are in bypass mode
  - All the RS are in fully isolation mode
  - All the RS at the slave interface of PL300 are in forward isolation mode
  - All the RS at the master interface of PL300 are in forward isolation mode

- The register slice gain estimator uses the STA results of the above 4 RS configurations to build an interpolation model to predict the slack value for any register slice solutions

- The reason why the 4 RS configurations are used for building the interpolation model is a tradeoff between complexity and accuracy
Four RS Configurations Needed

(a) Bypass Mode at All Ports
(b) Full Mode at All Ports
(d) Forward Mode at Slave Interfaces
(d) Forward Mode at Master Interface
The Estimator Is Accurate

- The mismatch between the timing result of the synthesized netlist and that obtained by the RS gain estimator is a Gaussian like function.
- The variance is pretty small for a practical SoC design.
The objective is to minimize the area overhead and traffic bandwidth penalty of register slice usage under the constraint of meeting timing closure.
Problem Formulation of RS Optimization (II)

- Mathematically, the RS optimization problem is

\[
\min_{\mathbf{x} \in \mathbf{C}} \mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \quad \text{s.t.} \quad t(\mathbf{x}) \geq V_{th}
\]

- \( \mathbf{x} \) is a vector of the mode configuration of the 5 AXI channels of \( \mathbf{P} \) register slices, \( x_{p,q} \in \{0,1,2\} \) represents respectively the fully isolation mode, forward isolation mode and bypass mode of the \( q \)-th channel of the \( p \)-th register slice. \( q = \{0,1,2,3,4\} \) denotes the 5 AXI channels, i.e. AR, R, AW, W, and B.

- \( \mathbf{C} = \{0,1,2\}^5 \) is a 5\( \mathbf{P} \)-dimensional space, and each dimension has 3 integers.

- \( \mathbf{f}(\mathbf{x}) \) is the objective function for optimization, which has 2 objective functions \( f_1(\mathbf{x}) \) and \( f_2(\mathbf{x}) \) for area and traffic penalty respectively.

- \( t(\mathbf{x}) \) is the worst timing slack of the timing related with AXI bus design, and \( V_{th} \) is the minimum allowed slack value according to the design specification.
Problem Formulation of RS Optimization (III)

- The multiple objective function $f(x)$ can be transformed into a scalar objective function. Here, $f(x)$ is transformed into a positively weighted convex sum of the objectives:

$$\min_{x \in C} f(x) = \sum_{i=1}^{2} \alpha_i f_i(x) \quad \alpha_i \geq 0, \ i = 1, 2, \quad s.t. \quad t(x) \geq V_{th}$$

- Let $a(x_{p,q})$ is the area of the q-th channel of the p-th register slice at mode $x_{p,q}$, then $f_1(x) = \sum_{p,q} a(x_{p,q})$ is the area overhead of the register slice insertion $x$.

- Let $b(x_{p,q})$ is the penalty factor to overall traffic delay due to the insertion of q-th channel of p-th register slice at mode $x_{p,q}$, then $f_2(x) = \sum_{p,q} b(x_{p,q})$ is the traffic penalty of the register slice insertion $x$.

- The optimization problem can be decoupled for the 5 AXI channels

$$\min_{x \in C_q} f_q(x) = \sum_{i=1}^{2} \alpha_{i,q} f_{i,q}(x) \quad \alpha_{i,q} \geq 0, \ i = 1, 2 \quad q = 1, 2, 3, 4, 5$$

$$s.t. \quad t_q(x) \geq V_{th}$$
The Exterior Penalty method is used to convert the constrained optimization problem into an unconstrained problem:

\[
\min_{x \in C_q} \left( \sum_{i=1}^{2} \alpha_{i,q} f_{i,q}(x) \right) + \theta_q \sum_{m=1}^{M_q} \xi \left( t_{q,m}(x) - V_{th} \right), \quad \alpha_{i,q} \geq 0 \quad i = 1, 2, \quad q = 1, 2, 3, 4, 5
\]

- \( t_{q,m}(x) \) is the worst timing slack in the m-th timing path group of the q-th channel
- \( \xi(x) = \begin{cases} 
1, & x < 0 \\
0, & x \geq 0 
\end{cases} \)
- \( \theta_q > 0 \) is the weighting factor to balance the influence of the original objective function and the constraint penalty
The Iterative Optimization Solver

- Denote the objective function
  \[ g(x) = \left( \sum_{i=1}^{2} \alpha_{i,q} f_{i,q}(x) \right) + \theta_q \sum_{m=1}^{M_q} \xi(t_{q,m}(x) - V_{th}) \]

- Optimality Criteria:
  \[ \forall x \in C_k, \quad g(x) \geq g(x_k) \]
  Where \( C_k \) is the subset of \( C_q \) whose elements differ with \( x_k \) at only one position

- The search direction is the maximum of
  \[ x_{k+1} = \left\{ x \mid \max_{x \in C_k} \frac{g(x_k) - g(x)}{|x - x_k|} \right\} \]

Initialization
\[ k=0, x_0 = [2,2,...,2] \]

Optimal Sol.? No

k=k+1

Calculate the Search direction

Synthesize the RS solution \( x_k \)
The Optimization Engine for RS Optimization

- The optimization engine runs very fast (less than a minute)
- The optimization engine minimize both the area overhead and traffic latency penalty of register slice insertion under the constraint of meeting a certain timing closure criterion
- The optimization engine can give a series of optimization problem solutions, each for a different timing closure target, which provides a solid base for making a tradeoff with other timing closure techniques, e.g. changing to the fast library cells (normally more power consumption)
Problem Statement

- Introduction
- Problem Statement
- Our Solution
- A SoC design Case Study
- Conclusions
RS Optimization for an Example SoC

- The bus design of the example SoC comprises two AXI bus,
  - Bus A: 64 bit PL300
  - Bus B: 32 bit PL300
- RSC is the number of elements in configuration vector \( x \) whose values are 1 (forward mode) or 0 (full mode)
- Design cycle is the time consumed by the process of achieving timing closure for the AXI bus related paths

<table>
<thead>
<tr>
<th></th>
<th>Bus A</th>
<th>Bus B</th>
<th>Bus A + Bus B</th>
<th>Design Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td>80 RSC</td>
<td>15 RSC</td>
<td>95 RSC</td>
<td>14 days</td>
</tr>
<tr>
<td>Proposed</td>
<td>22 RSC</td>
<td>8 RSC</td>
<td>30 RSC</td>
<td>5 days</td>
</tr>
<tr>
<td>Reduction</td>
<td>73%</td>
<td>47%</td>
<td>68%</td>
<td>64%</td>
</tr>
</tbody>
</table>
Problem Statement

- Introduction
- Problem Statement
- Our Solution
- A SoC design Case Study
- Conclusions
Conclusions

- A new design flow of timing closure for AXI bus based design has been proposed
- The proposed Register Slice optimization flow/tool:
  - Eases the effort for the AXI bus timing closure
  - Avoids the iterations of AXI bus timing closure, and reduces more than of the design cycle of register slice insertion
  - Provides a near-optimal solution for minimizing both the area overhead and the traffic penalty of the Register Slice usage to meet timing closure
  - Has been fully automated in PERL scripts, and has a fast run speed
  - Enable system tradeoff between different timing closure measures
- The result from its applications in a real SoC design is very encouraging