SystemCセミナー【TLM実践コース】 目次

第1章　TLモデリングの記述スタイル

1. システム開発の現状
2. 仕様および実装上の問題
3. アルゴリズム設計の導入
4. TL設計の導入
5. 通信インタフェースの共通化
6.
7.
8. 概要　コーディングスタイル（トランザクション）
9. 概要　コーディングスタイル（通信シーケンス）
10. 概要　コーディングスタイル（モデル構成）
11. 概要　LTコーディングスタイル
12. 概要　ATコーディングスタイル
13. データの粒度の違い
14. 通信レベルの変更に対応する
15. 通信インタフェースの交換

第2章　OSCI TLM-2.0　LTコーディングスタイル

1. 共通TLMインタフェース
2.
3. ソケット（種類）
4. シンプルソケット　概要
5. ソケット(LT/AT共通)
6. ソケット（標準ソケット）
7. イニシエータの記述（シンプルソケット）
8. ターゲットの記述（シンプルソケット）
9. 標準ソケット
10. インタフェース（LT/AT共通インタフェース）
11. インタフェース（Blocking Transport/LTのみ）
12. インタフェース（Debug Transport/LT/AT共通）
13. インタフェース（Direct Memory/LT/AT共通）
14. インタフェース（Direct Memory/LT/AT共通）
15.
16. 汎用ペイロード(LT/AT共通)
17. 汎用ペイロード（アトリビュートの種類）
18. 汎用ペイロード
    （アトリビュート・アクセス用のメンバ関数）
19. 汎用ペイロード
    （アトリビュート・アクセス用のメンバ関数）
20. 汎用ペイロード（データポインタ）
21. 汎用ペイロード（データ長）
22. 汎用ペイロード
    （データ転送例 : unsigned char -> unsigned char）
23. 汎用ペイロード
    （データ転送例 : unsigned int -> unsigned char）
24. 汎用ペイロード（バイトイネーブル）
25. 完全なターゲットモデルのアドレッシング
26. 共用体(union)の利用
27. 共用体によるレジスタアドレス表現
28. 汎用ペイロード（ストリーミング幅）
29. 汎用ペイロード（ストリーミング幅）（2）
30. 汎用ペイロード（応答ステータス）
31. 汎用ペイロード（応答ステータス）
32.
33. ユーティリティ（エンディアン変換）
34. ユーティリティ（エンディアン変換）
35. ユーティリティ（エンディアン変換）

第3章　LTモデル作成手順

1. ターゲットで使用するアルゴリズム（バブルソート）
2. TLM入門で紹介したC++モデル
3. bsortの記述(C++)
4. tbの記述(C++)
5. TLM2.0通信部分のクラスを作成（target_base.h）
6. TLM2.0通信部分のクラスを作成（initiator_base.h）
7. 通信基本クラスの記述方針
8. bsortのTLM化
9. tbの記述のTLM化
10. ターゲットの自律動作
11. 状態保持レジスタの追加
12. tbの変更
13. bsortの変更
14. 遅延の追加
15. プロトコル（通信シーケンス　同期LT）
16. タイミングアノテーションLT
17. 遅延の設定
18. 機能の遅延
19. 遅延動作の通信シーケンス
20. 実行結果
21. ターゲットの自律動作と遅延の関係
22. 自律動作させない遅延記述
23. ターゲット兼イニシエータ作成とアドレスの再検討
24. シンプルバスの導入
25. SimpleBusLT
26. アドレスの変更
27. bsortにイニシエータを追加
28. メモリ（ターゲット）
29. バスへの接続
30. 3つのモデルの通信シーケンス
31. 実行結果
32. 結果の波形表示

SystemCセミナー【TLM実践コース】 目次

第4章　LT高速化のノウハウ

1. テンポラルデカップリングの採用
2. プロトコル（通信シーケンス　テンポラルデカップリング LT）
3. ユーティリティ（tlm_quantum_keeper/LT）
4. イニシエータの変更
5. イニシエータ／ターゲットの変更
6. 実行結果
7. テンポラルデカップリングの精度（概念）
8. 結果が異なる事例(同一メモリアクセス)
9. 結果が同一の事例(同一メモリアクセス)
10. 結果が異なる事例(同時刻イベント)
11. wait(SC_ZERO_TIME)を挿入してこの問題を回避
12. テンポラルデカップリング まとめ
13. ダイレクトメモリーアクセス(DMI)の採用
14. DMIポインタの取得
15. DMI記述子（アトリビュート・アクセス用のメンバ関数）
16. DMI記述子（アトリビュート・アクセス用のメンバ関数）
17. DMIアクセスを利用するときの注意点
18. イニシエータの変更
19. イニシエータの変更
20. ターゲットの変更

第5章　TLMの実践

1. TLM環境でやりたいこと
2. TLM環境に必要なもの
3. 汎用プロセッサモデル（仮想CPU：hdlab社製）
4. OVP
5. OVPCPUモデル
6. サンプル. SystemC_TLM2.0
7. 仮想CPU環境
8. OpenRiscへの移植
9. ARMコアに移植する場合

演習