参考資料

Application Note

MMA エラー注入

Manojna Manojna

概要

故障注入と例外処理は、安全システムにとって重要です。この資料は、MMA 故障を逐次的に注入する方法を説明します。

目次

商標

1 はじめに

マトリクス乗算アクセラレータ (MMA) は、C7x プロセッサ アーキテクチャのスカラーおよびベクトル機能を拡張する密結合行列乗算コプロセッサです。ストリーミング エンジン (SE) を介してデータを受け取る MMA は、大量の乗算累算 (MAC) 演算、行列最適化されたストレージ、および行列最適化されたデータ転送を提供します。これにより、ビジョン (CNN、SfM、フィルタリングなど)、音声 (xNN)、オーディオ (畳み込み)、レーダー (FFT)、および制御 (強化学習、状態更新) に基づくアプリケーションで支配的な高密度線形代数演算を効率的に実行できます。

この資料では、故障注入のシーケンシャル プロセスについて説明し、特に MMA (マトリックス乗算器アクセラレータ) 診断に注目します。

この資料は、これらのテストを順に実施する方法について説明し、故障への対処と回復について詳細に理解することを検証します。

この文書では、エラー注入のシーケンスが実行された後にコードの通常のフローを実行するための例外の回復については説明していません。統合例については、SDK 11.2 をご覧ください。

2 命令

2.1 範囲

この資料では、MMA のシーケンシャル故障インジェクション テストのみに焦点を当て、これらの故障をトリガして実行する方法の詳細を説明します。

このドキュメントでは、連続テストの完了後に c7x をリセットする手順については説明していません。テスト後のリカバリ、システム リセット、およびシステム状態の復元は本書の対象外であり、システム設計およびフォールト トレランスに従って別途実施する必要があります。

2.2 HWA 命令定義

ハードウェア アクセラレータ (HWA) は、C7x CPU と緊密に結合されています。HWA は C7x CPU からコマンドを受け取り、特定の計算タスクを実行し、その結果を C7x CPU に返します。

- HWAOPEN：以降の HWA 動作で使用するオペランドの種類と計算内容を HWA に伝えるため、コンピュート テンプレートを HWA に送信します。
- HWALDA、HWALDB、HWALDC、HWALDAB、HWALDBC：これらの各種 HWALD 命令は、GRF、LRFL、SE0、SE1 レジスタから値を読み取り、それらのレジスタを HWA に送信して、HWA のオペランド マトリクスを初期化するために使用されます。
- HWAOP：HWAOP 命令は、HWA に対して、プログラムされた計算を続行するように指示します。
- HWAXFER：HWAXFER 命令は、HWA ロジックから計算された結果を内部バッファに転送します。
- HWARCVS：HWARCVS 命令は、HWA 内部バッファに格納された値を、C7x CPU 汎用レジスタに転送します。
- HWACLOSE：HWACLOSE は HWA 操作を終了します。HWA 内のすべての中間値は破棄されます。

詳細については、MMA の仕様書を参照します。詳細については、*C71x DSP CPU、命令セット、マトリクス乗算アクセラレータ*を参照します。

2.3 故障注入手順

故障注入手順

MMA の診断を評価するためには、発生する動作を監視しながら、システムに故障を意図的に注入する必要があります。HWA_STATUS エラー コード フィールドに 0 以外の値が含まれている場合、HWARCV_(0) 命令を実行すると、c7x 例外がトリガされます。

例外が発生した後、IERR レジスタおよび IESR レジスタを読み取って、故障の内容を特定する必要があります:

- IERR: 特定のフラグで内部例外の原因を示します。
- IESR: 故障タイプとサブタイプの詳細を示します。

2.3.1 ブロック図

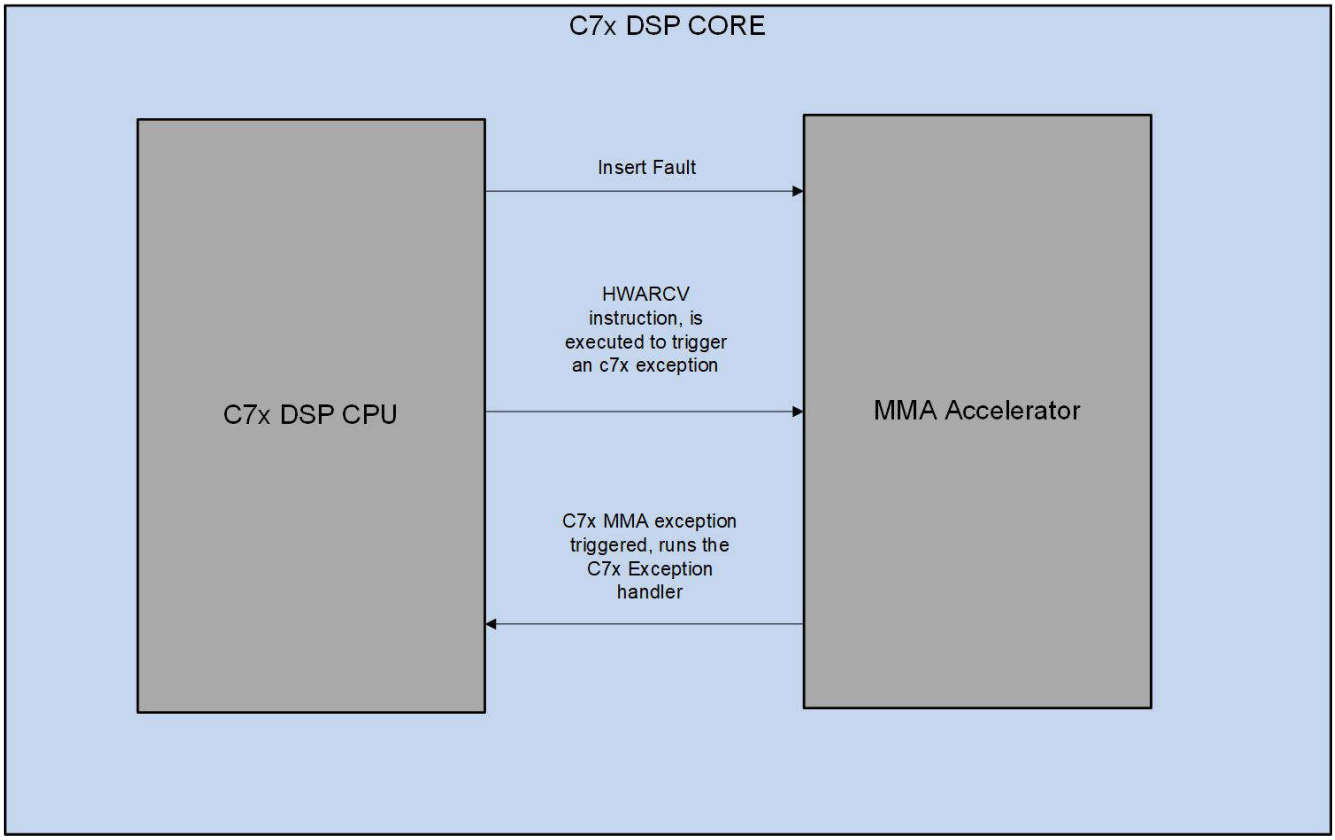

図 2-1. C7x-MMA エラー注入のブロック図

2.3.2 故障注入：ステップ

2.3.2.1 アンダーフロー エラー

アンダーフロー エラーは、転送バッファの読み取りが試行されたが、FIFO に有効なデータがなかった場合に発生します (プログラミング エラー)。以下にアンダーフロー エラーを注入する手順を示します。

1. HWAOPEN 命令を使用して、HWA_CONFIG および HWA_OFFSET を書き込みます。
2. HWAXFER を実行して、HWA_STATUS の内容を内部バッファに転送します。
3. HWARCV_(0) 命令を実行して、HWA 内部バッファに格納された値を C7x CPU の汎用レジスタに転送します。
4. c7x CPU への内部バッファに有効なデータがなくなるまで、ステップ 3 を繰り返します。このデータは例外をトリガーします。

```
void underflow_exception(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg;
  mma_config_reg = __gen_HWA_CONFIG_REG_v1();
  __HWA_OFFSET_REG offset_reg;
  offset_reg = __gen_HWA_OFFSET_REG();
  __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
  __HWAADV();
  __HWAADV();
  __HWAXFER(__MMA_XFER_SRC_HWA_STATUS);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWARCV(0);
  __HWARCV(0);
}
```

2.3.2.2 オーバーフロー エラー

オーバーフロー エラーは、転送バッファの読み取りを試みた際に、以前の HWA 命令によって FIFO がオーバーフローした場合に発生します (プログラミング エラー)。表示されている手順に従って、エラーを注入します。

1. HWAOPEN 命令を使用して、HWA_CONFIG および HWA_OFFSET を書き込みます。
2. HWAXFER を実行して、HWA_STATUS の内容を内部バッファに転送します。
3. バッファがオーバーフローするまで、手順 2 を繰り返します。
4. HWARCV 命令が実行され、エラーが観測されます。

```
void overflow_exception(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg;
  mma_config_reg = __gen_HWA_CONFIG_REG_v1();
  __HWA_OFFSET_REG offset_reg;
  offset_reg = __gen_HWA_OFFSET_REG();
  __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
  int loop_begin = 1;
  int loop_end  = 30;
  for(;loop_begin<=loop_end;loop_begin++)
  {
    __HWAXFER(__MMA_XFER_SRC_HWA_STATUS);
    __HWAADV();
    __HWAADV();
    __HWAADV();
    __HWAADV();
  }
  __HWARCV(0);
}
```

2.3.2.3 オフセット パリティ エラー

オフセット パリティ エラーは、HWA_OFFSET レジスタでパリティの不一致が検出されたときに発生します。
HWA_OFFSET パリティ エラーを注入するには、次の手順を実行します。

1. HWAOPEN 命令を使用して、パリティ計算をイネーブルにし、パリティ チェックをイネーブルにした状態 (PARITYCTRL = NORMAL) に HWA_CONFIG と HWA_OFFSET を書き込みます。
2. ベクトル レジスタの値は、テスト対象のフィールドのソフトウェアによって破損します。
3. HWAOPEN 命令は再び使用され、パリティ計算を無効にし、パリティ チェックを有効 (PARITYCTRL = PNCMCK) に設定した状態で、その C7x ベクタ レジスタの破損データを使用して HWA_CONFIG および HWA_OFFSET に書き込みを行います。このモードでは、最初に計算された保存されたパリティ値が、影響を受けるレジスタで使用されて計算されたパリティと比較してチェックされます。
4. HWAXFER を実行して、HWA_CONFIG/HWA_OFFSET などのコンテンツを転送し、バッファを転送します。
5. HWARCV 命令が実行され、エラーが観測されます。

```
 __HWA_OFFSET_REG get_corrupted_offset(void)
{
__HWA_OFFSET_REG mma_offset_reg = __gen_HWA_OFFSET_REG();
mma_offset_reg.A_LUT_VAL_1 = 0x01;
return mma_offset_reg; //return the corrupted offset
}
#pragma FUNC_CANNOT_INLINE(offset_parity_test)
#pragma FUNCTION_OPTIONS(offset_parity_test,"--opt_level=0")
void offset_parity_test(void)
{

__HWA_CONFIG_REG_v1 mma_config_reg;
mma_config_reg = __gen_HWA_CONFIG_REG_v1();
mma_config_reg.A_ALUTEN = __MMA_A_CONFIG_LUT2;
mma_config_reg.PARITYCTRL = __MMA_NORMAL;
__HWA_OFFSET_REG offset_reg;
offset_reg = __gen_HWA_OFFSET_REG();
__HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
__HWAADV();
__HWAADV();
__HWAADV();
__HWAADV();
__HWA_CONFIG_REG_v1 mma_config_1;
mma_config_1 = __gen_HWA_CONFIG_REG_v1();
mma_config_1.A_ALUTEN = __MMA_A_CONFIG_LUT2;
//Parity computation disabled and parity checking enabled.
mma_config_1.PARITYCTRL = __MMA_PNCM_CK;
__HWA_OFFSET_REG offset_reg2 = get_corrupted_offset(); //This gets the corrupted offset
__HWAADV();
__HWAADV();
__HWAADV();
__HWAADV();
__HWAOPEN(mma_config_1,offset_reg2,__MMA_OPEN_FSM_RESET);
__HWAADV();
__HWAADV();
__HWAADV();
__HWAADV();
__HWAXFER(__MMA_XFER_SRC_HWA_OFFSET);
__HWAADV();
__HWAADV();
__HWAADV();
__HWARCV(0);
__asm(" NOP");
__asm(" NOP");


}
```

2.3.2.4 設定パリティ エラー

HWA_CONFIG レジスタでパリティの不一致が検出されると、構成パリティ エラーが発生します。HWA_CONFIG パリティ エラーを注入する手順を以下の手順に示します。すべての FSM 構成 (A、B、C、X FSM 構成) はパリティによって保護されます。

1. HWAOPEN 命令を使用して、パリティ計算をイネーブルにし、パリティ チェックをイネーブルにした状態 (PARITYCTRL = NORMAL) に HWA_CONFIG と HWA_OFFSET を書き込みます。
2. ベクトル レジスタの値は、テスト対象のフィールドのソフトウェアによって破損します。
3. HWAOPEN 命令は再び使用され、パリティ計算を無効にし、パリティ チェックを有効 (PARITYCTRL = PNCMCK) に設定した状態で、その C7x ベクタ レジスタの破損データを使用して HWA_CONFIG および HWA_OFFSET に書き込みを行います。このモードでは、最初に計算された保存されたパリティ値が、影響を受けるレジスタで使用されて計算されたパリティと比較してチェックされます。
4. HWAXFER を実行して、HWA_CONFIG/HWA_OFFSET などのコンテンツを転送し、バッファを転送します。
5. HWARCV 命令が実行され、エラーが観測されます

2.3.2.4.1 FSM 構成パリティ エラー

HWA_CONFIG レジスタでパリティの不一致が検出された場合、A FSM 構成パリティ エラーが発生します。A FSM Config パリティ エラーを注入する手順は、次のとおりです。

1. HWAOPEN 命令は、パリティ計算およびパリティ チェックを有効 (PARITYCTRL = NORMAL) にした状態で HWA_CONFIG および HWA_OFFSET に書き込むために使用されます。また、Get_MMAconfig() で指定されたパラメータを使用して HWA_CONFIG を設定します。
2. ベクトル レジスタの値は、テストすべき特定のフィールド (A FSM) のソフトウェアによって破損します。
3. HWAOPEN 命令は再び使用され、パリティ計算を無効にし、パリティ チェックを有効 (PARITYCTRL = PNCMCK) に設定した状態で、その C7x ベクタ レジスタの破損データを使用して HWA_CONFIG および HWA_OFFSET に書き込みを行います。このモードでは、最初に計算された保存されたパリティ値が、影響を受けるレジスタで使用されて計算されたパリティと比較してチェックされます。
4. HWALDA/HWAXFER を実行してエラーを注入します。このエラーは HWA_STATUS レジスタで更新されます。
5. HWARCV 命令が実行され、エラーが観測されます

```
#pragma FUNCTION_OPTIONS(Get_AFSM_Corrupt_config,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_AFSM_Corrupt_config(void)
{
  __HWA_CONFIG_REG_v1 corrupted_config = __gen_HWA_CONFIG_REG_v1();
  corrupted_config.A_ATYPE = __MMA_A_CONFIG_ATYPE_UINT32;
  corrupted_config.A_ALUTEN = __MMA_A_LUTEN_LAST;
  return corrupted_config;
}
#pragma FUNCTION_OPTIONS(Get_MMAconfig,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_MMAconfig(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg;
  mma_config_reg = __gen_HWA_CONFIG_REG_v1();
  mma_config_reg.A_ATYPE = __MMA_A_CONFIG_ATYPE_UINT16;
  mma_config_reg.A_ALUTEN = __MMA_A_LUTEN_LAST;
  mma_config_reg.B_BSWPER = 1000;
  mma_config_reg.B_BTYPE = __MMA_B_CONFIG_SIZE16;
  mma_config_reg.B_BSTART = 1;
  mma_config_reg.C_ATYPE = __MMA_C_CONFIG_ATYPE_SA;
  mma_config_reg.C_BTYPE = __MMA_C_CONFIG_BTYPE_UINT16;
  mma_config_reg.PARITYCTRL = __MMA_NORMAL;
  mma_config_reg.C_BSTART = 1; /* Initial B bank selection for reading B matrix data for the matrix
computations */
  mma_config_reg.C_CRSTART = 1; /* Initial C bank selection for reading operands */
  mma_config_reg.C_CWSTART = 1; /* Initial C bank selection for writing computation results */
  mma_config_reg.C_CLSTART = 1;
  return mma_config_reg;
}
#pragma FUNCTION_OPTIONS(MMA_AFSM_configParityError_injection,"--opt_level=off")
void MMA_AFSM_configParityError_injection(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg = Get_MMAconfig();
  __HWA_OFFSET_REG offset_reg;
```

```
  offset_reg = __gen_HWA_OFFSET_REG();
  __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __asm(" HWALDA .L2 VB1");
  __HWAXFER(__MMA_XFER_SRC_C);
  __HWA_CONFIG_REG_v1 mma_config_corrupt = Get_AFSM_Corrupt_config();
  mma_config_corrupt.PARITYCTRL = __MMA_PNCM_CK;
  __HWA_OFFSET_REG offset_reg2 = __gen_HWA_OFFSET_REG();
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWAOPEN(mma_config_corrupt,offset_reg2,__MMA_OPEN_FSM_RESET);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __asm(" HWALDA .L2 VB0"); //Load to A matrix
  __HWAXFER(__MMA_XFER_SRC_HWA_CONFIG);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWARCV(0);
  __asm(" NOP");
  __asm(" NOP");
}
```

2.3.2.4.2 B FSM 構成パリティ エラー

HWA_CONFIG レジスタでパリティの不一致が検出された場合、B FSM 構成パリティ エラーが発生します。B FSM Config パリティ エラーを注入する手順は、次のとおりです。

1. HWAOPEN 命令は、パリティ計算およびパリティ チェックを有効 (PARITYCTRL = NORMAL) にした状態で HWA_CONFIG および HWA_OFFSET に書き込むために使用されます。また、Get_MMAconfig() で指定されたパラメータを使用して HWA_CONFIG を設定します。
2. ベクトル レジスタの値は、テストすべき特定のフィールド (B FSM) のソフトウェアによって破損します。
3. HWAOPEN 命令は再び使用され、パリティ計算を無効にし、パリティ チェックを有効 (PARITYCTRL = PNCMCK) に設定した状態で、その C7x ベクタ レジスタの破損データを使用して HWA_CONFIG および HWA_OFFSET に書き込みを行います。このモードでは、最初に計算された保存されたパリティ値が、影響を受けるレジスタで使用されて計算されたパリティと比較してチェックされます。
4. HWALDB/HWAXFER を実行してエラーを注入します。このエラーは HWA_STATUS レジスタで更新されます。
5. HWARCV 命令が実行され、エラーが観測されます

```
#pragma FUNCTION_OPTIONS(Get_BFSM_Corrupt_config,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_BFSM_Corrupt_config(void)
{
  __HWA_CONFIG_REG_v1 corrupted_config = __gen_HWA_CONFIG_REG_v1();
  corrupted_config.B_BSWPER = 100;
  corrupted_config.B_BTYPE = __MMA_B_CONFIG_SIZE32;
  return corrupted_config;
}
#pragma FUNCTION_OPTIONS(Get_MMAconfig,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_MMAconfig(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg;
  mma_config_reg = __gen_HWA_CONFIG_REG_v1();
  mma_config_reg.A_ATYPE = __MMA_A_CONFIG_ATYPE_UINT16;
  mma_config_reg.A_ALUTEN = __MMA_A_LUTEN_LAST;
  mma_config_reg.B_BSWPER = 1000;
  mma_config_reg.B_BTYPE = __MMA_B_CONFIG_SIZE16;
  mma_config_reg.B_BSTART = 1;
  mma_config_reg.C_ATYPE = __MMA_C_CONFIG_ATYPE_SA;
  mma_config_reg.C_BTYPE = __MMA_C_CONFIG_BTYPE_UINT16;
  mma_config_reg.PARITYCTRL = __MMA_NORMAL;
  mma_config_reg.C_BSTART = 1; /* Initial B bank selection for reading B matrix data for the matrix
computations */
  mma_config_reg.C_CRSTART = 1; /* Initial C bank selection for reading operands */
  mma_config_reg.C_CWSTART = 1; /* Initial C bank selection for writing computation results */
  mma_config_reg.C_CLSTART = 1;
  return mma_config_reg;
```

```
}
#pragma FUNCTION_OPTIONS(MMA_BFSM_configParityError_injection,"--opt_level=off")
void MMA_BFSM_configParityError_injection(void)
{
__HWA_CONFIG_REG_v1 mma_config_reg = Get_MMAconfig();
  __HWA_OFFSET_REG offset_reg;
  offset_reg = __gen_HWA_OFFSET_REG();
  __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __asm(" HWALDB .L2 VB1");
  __HWAXFER(__MMA_XFER_SRC_C);
  __HWA_CONFIG_REG_v1 mma_config_corrupt = Get_BFSM_Corrupt_config();
  mma_config_corrupt.PARITYCTRL = __MMA_PNCM_CK;
  __HWA_OFFSET_REG offset_reg2 = __gen_HWA_OFFSET_REG();
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWAOPEN(mma_config_corrupt,offset_reg2,__MMA_OPEN_FSM_RESET);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __asm(" HWALDB .L2 VB0"); //Load to B matrix
  __HWAXFER(__MMA_XFER_SRC_HWA_CONFIG);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWARCV(0);
  __asm(" NOP");
  __asm(" NOP");

}
```

2.3.2.4.3 C FSM 構成パリティ エラー

HWA_CONFIG レジスタでパリティの不一致が検出されると、C FSM 構成パリティ エラーが発生します。C FSM Config パリティ エラーを注入する手順を以下の手順に示します。

1. HWAOPEN 命令は、パリティ計算およびパリティ チェックを有効 (PARITYCTRL = NORMAL) にした状態で HWA_CONFIG および HWA_OFFSET に書き込みを行うために使用されます。また、Get_MMAconfig() で指定されたパラメータを用いて HWA_CONFIG を設定します。
2. ベクタ レジスタ内の値は、テスト対象の特定フィールド (C FSM) においてソフトウェアによって破損されます。
3. HWAOPEN 命令は再び使用され、パリティ計算を無効にし、パリティ チェックを有効 (PARITYCTRL = PNCMCK) に設定した状態で、その C7x ベクタ レジスタの破損データを使用して HWA_CONFIG および HWA_OFFSET に書き込みを行います。このモードでは、最初に計算された保存されたパリティ値が、影響を受けるレジスタで使用されて計算されたパリティと比較してチェックされます。
4. HWALDC/HWAXFER を実行してエラーを注入します。このエラーは HWA_STATUS レジスタで更新されます。
5. HWARCV 命令が実行され、エラーが観測されます。

```
#pragma FUNCTION_OPTIONS(Get_CFSM_Corrupt_config,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_CFSM_Corrupt_config(void)
{
  __HWA_CONFIG_REG_v1 corrupted_config = __gen_HWA_CONFIG_REG_v1();
  corrupted_config.C_ATYPE = __MMA_C_CONFIG_ATYPE_UA;
  corrupted_config.C_BTYPE = __MMA_C_CONFIG_BTYPE_UINT16;
  return corrupted_config;
}
#pragma FUNCTION_OPTIONS(Get_MMAconfig,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_MMAconfig(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg;
  mma_config_reg = __gen_HWA_CONFIG_REG_v1();
  mma_config_reg.A_ATYPE = __MMA_A_CONFIG_ATYPE_UINT16;
  mma_config_reg.A_ALUTEN = __MMA_A_LUTEN_LAST;
  mma_config_reg.B_BSWPER = 1000;
  mma_config_reg.B_BTYPE = __MMA_B_CONFIG_SIZE16;
  mma_config_reg.B_BSTART = 1;
  mma_config_reg.C_ATYPE = __MMA_C_CONFIG_ATYPE_SA;
  mma_config_reg.C_BTYPE = __MMA_C_CONFIG_BTYPE_UINT16;
  mma_config_reg.PARITYCTRL = __MMA_NORMAL;
```

```
  mma_config_reg.C_BSTART = 1; /* Initial B bank selection for reading B matrix data for the matrix
computations */
  mma_config_reg.C_CRSTART = 1; /* Initial C bank selection for reading operands */
  mma_config_reg.C_CWSTART = 1; /* Initial C bank selection for writing computation results */
  mma_config_reg.C_CLSTART = 1;
  return mma_config_reg;
}

#pragma FUNCTION_OPTIONS(MMA_CFSM_configParityError_injection,"--opt_level=off")
void MMA_CFSM_configParityError_injection(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg = Get_MMAconfig();
  __HWA_OFFSET_REG offset_reg;
  offset_reg = __gen_HWA_OFFSET_REG();
  __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
 __HWAADV();
  __HWAADV();
  __HWAADV();
  __asm(" HWALDC .L2 VB1");
  __HWAXFER(__MMA_XFER_SRC_C);
  __HWA_CONFIG_REG_v1 mma_config_corrupt = Get_CFSM_Corrupt_config();
  mma_config_corrupt.PARITYCTRL = __MMA_PNCM_CK;
  __HWA_OFFSET_REG offset_reg2 = __gen_HWA_OFFSET_REG();
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWAOPEN(mma_config_corrupt,offset_reg2,__MMA_OPEN_FSM_RESET);
   __HWAADV();
  __HWAADV();
  __HWAADV();
  __asm(" HWALDC .L2 VB0"); //Load to C Matrix
  __HWAXFER(__MMA_XFER_SRC_HWA_CONFIG);
 __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWARCV(0);
  __asm(" NOP");
  __asm(" NOP");
}
```

2.3.2.4.4 X FSM 構成パリティエラー

HWA_CONFIG レジスタでパリティの不一致が検出された場合、X FSM 構成パリティ エラーが発生します。X FSM 構成パリティ エラーを注入する手順を、次の手順に示します。

1. HWAOPEN 命令は、パリティ計算およびパリティ チェックを有効 (PARITYCTRL = NORMAL) にした状態で HWA_CONFIG および HWA_OFFSET に書き込むために使用されます。また、Get_MMAconfig() で指定されたパラメータを使用して HWA_CONFIG を設定します。
2. ベクトル レジスタの値は、テストすべき特定のフィールド (X FSM) のソフトウェアによって破損します。
3. HWAOPEN 命令は再び使用され、パリティ計算を無効にし、パリティ チェックを有効 (PARITYCTRL = PNCMCK) に設定した状態で、その C7x ベクタ レジスタの破損データを使用して HWA_CONFIG および HWA_OFFSET に書き込みを行います。このモードでは、最初に計算された保存されたパリティ値が、影響を受けるレジスタで使用されて計算されたパリティと比較してチェックされます。
4. HWAXFER を実行してエラーを注入します。このエラーは HWA_STATUS レジスタで更新されます。
5. HWARCV 命令が実行され、エラーが観測されます。

```
#pragma FUNCTION_OPTIONS(Get_XFSM_Corrupt_config,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_XFSM_Corrupt_config(void)
{
  __HWA_CONFIG_REG_v1 corrupted_config = __gen_HWA_CONFIG_REG_v1();
  corrupted_config.X_ReLU = 1;
  return corrupted_config;
}
#pragma FUNCTION_OPTIONS(Get_MMAconfig,"--opt_level=off")
__HWA_CONFIG_REG_v1 Get_MMAconfig(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg;
  mma_config_reg = __gen_HWA_CONFIG_REG_v1();
  mma_config_reg.A_ATYPE = __MMA_A_CONFIG_ATYPE_UINT16;
  mma_config_reg.A_ALUTEN = __MMA_A_LUTEN_LAST;
```

```
  mma_config_reg.B_BSWPER = 1000;
  mma_config_reg.B_BTYPE = __MMA_B_CONFIG_SIZE16;
  mma_config_reg.B_BSTART = 1;
  mma_config_reg.C_ATYPE = __MMA_C_CONFIG_ATYPE_SA;
  mma_config_reg.C_BTYPE = __MMA_C_CONFIG_BTYPE_UINT16;
  mma_config_reg.PARITYCTRL = __MMA_NORMAL;
  mma_config_reg.C_BSTART = 1; /* Initial B bank selection for reading B matrix data for the matrix
computations */
  mma_config_reg.C_CRSTART = 1; /* Initial C bank selection for reading operands */
  mma_config_reg.C_CWSTART = 1; /* Initial C bank selection for writing computation results */
  mma_config_reg.C_CLSTART = 1;
  return mma_config_reg;
}
#pragma FUNCTION_OPTIONS(MMA_XFSM_configParityError_injection,"--opt_level=off")
void MMA_XFSM_configParityError_injection(void)
{
  __HWA_CONFIG_REG_v1 mma_config_reg = Get_MMAconfig();
  __HWA_OFFSET_REG offset_reg;
  offset_reg = __gen_HWA_OFFSET_REG();
  __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET); //Initialises Config,offset
 __HWAXFER(__MMA_XFER_SRC_HWA_CONFIG);
  __HWA_CONFIG_REG_v1 mma_config_corrupt = Get_XFSM_Corrupt_config(); //Receives X FSM corrupt
configuration
  mma_config_corrupt.PARITYCTRL = __MMA_PNCM_CK; //Parity checking enabled calculation disabled
  __HWA_OFFSET_REG offset_reg2 = __gen_HWA_OFFSET_REG();
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWAOPEN(mma_config_corrupt,offset_reg2,__MMA_OPEN_FSM_RESET);
 __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWAXFER(__MMA_XFER_SRC_HWA_CONFIG);
  __HWAADV();
  __HWAADV();
  __HWAADV();
  __HWARCV(0);
  __asm(" NOP");
  __asm(" NOP");
}
```

2.3.2.5 C 読み取りエラー

C 読み取りエラーは、同じ C マトリクス バンクに対して複数の読み取りアクセスが同時に行われ、競合が発生した場合に起こります。以下の手順に従って、このエラーを注入します。

1. HWAOPEN 命令は、動作を MULPLUS として構成することで、HWA_CONFIG と HWA_OFFSET を書き込むために使用されます。
2. HWAOP と HWAXFER 演算を並列実行します。
3. C マトリクス バンクへの短時間かつ頻繁な読み取りおよび書き込みにより、エラーが発生します。
4. HWARCV 命令が実行され、エラーが観測されます。

```
void c_read_error(void)
{
    int switch_period=0;
    int repeat_op_count = 0;
    for( ;switch_period<5;switch_period++)
    {
        __HWA_CONFIG_REG_v1 config_reg = __gen_HWA_CONFIG_REG_v1();
        config_reg.C_OPERATION0 = __MMA_C_CONFIG_MULPLUS;
        config_reg.C_OPERATION1 = __MMA_C_CONFIG_MULPLUS;
        config_reg.C_CRSWPER = switch_period; //different switching period
        config_reg.C_CWSWPER  = switch_period+10000;
        __HWA_OFFSET_REG offset_reg;
        offset_reg = __gen_HWA_OFFSET_REG(); //This generates the offset register.
        __HWAOPEN(config_reg,offset_reg,__MMA_OPEN_FSM_RESET); //This opens theHWAOPEN
        repeat_op_count = 0;
        for(;repeat_op_count<5;repeat_op_count++)
        {
        __asm(" HWAOP.S1 SA0");
        __asm(" ||HWAXFER .L1 CMAT");
        __HWAADV();
        __HWAADV();
        __HWAADV();
        __HWAADV();
        }
    }
  __HWARCV(0); //calls the error
  __asm(" NOP");
__asm(" NOP");
}
```

2.3.2.6 C 書き込みエラー

C 書き込みエラーは、同じ C マトリックス バンクへの同時書き込みアクセスが複数ある場合に発生し、競合が発生します。以下の手順に従って、このエラーを注入します。

1. HWAOPEN 命令は、動作を MULPLUS として構成することで、HWA_CONFIG と HWA_OFFSET を書き込むために使用されます。
2. HWAOP と HWALDC 演算を並列実行します。
3. C マトリクス バンクへの短時間かつ頻繁な読み取りおよび書き込みにより、エラーが発生します。
4. HWARCV 命令が実行され、エラーが観測されます。

```
void c_write_error(void)
{
   int switch_period=0;
    int repeat_op_count = 0;
    for( ;switch_period<2;switch_period++)
    {
        __HWA_CONFIG_REG_v1 config_reg = __gen_HWA_CONFIG_REG_v1();
        config_reg.C_OPERATION0 = __MMA_C_CONFIG_MULPLUS;
        config_reg.C_OPERATION1 = __MMA_C_CONFIG_MULPLUS;
        config_reg.C_CRSWPER = switch_period; //different switching period
        config_reg.C_CWSWPER  = switch_period+10000;
        __HWA_OFFSET_REG offset_reg;
        offset_reg = __gen_HWA_OFFSET_REG(); //This generates the offset register.
        __HWAOPEN(config_reg,offset_reg,__MMA_OPEN_FSM_RESET); //This opens theHWAOPEN
        repeat_op_count = 0;
        for(;repeat_op_count<10;repeat_op_count++)
        {
        __asm(" HWALDC .L2 VB0"); //Earlier SE0++
        __asm(" ||HWAOP.S1 SA0"); //This should run them in parallel
        __HWAADV();
        __HWAADV();
        __HWAADV();
        __HWAADV();
        }
    }
    __HWARCV(0);  //Calls the error
    __asm(" NOP");
    __asm(" NOP");
}
```

3 フロー図

この状態遷移図は、MMA の故障注入テストが順次実行される流れを示しています。この画像は、故障を導入して故障から回復し、次の一連の故障を注入するための体系的な手順を概説しています。

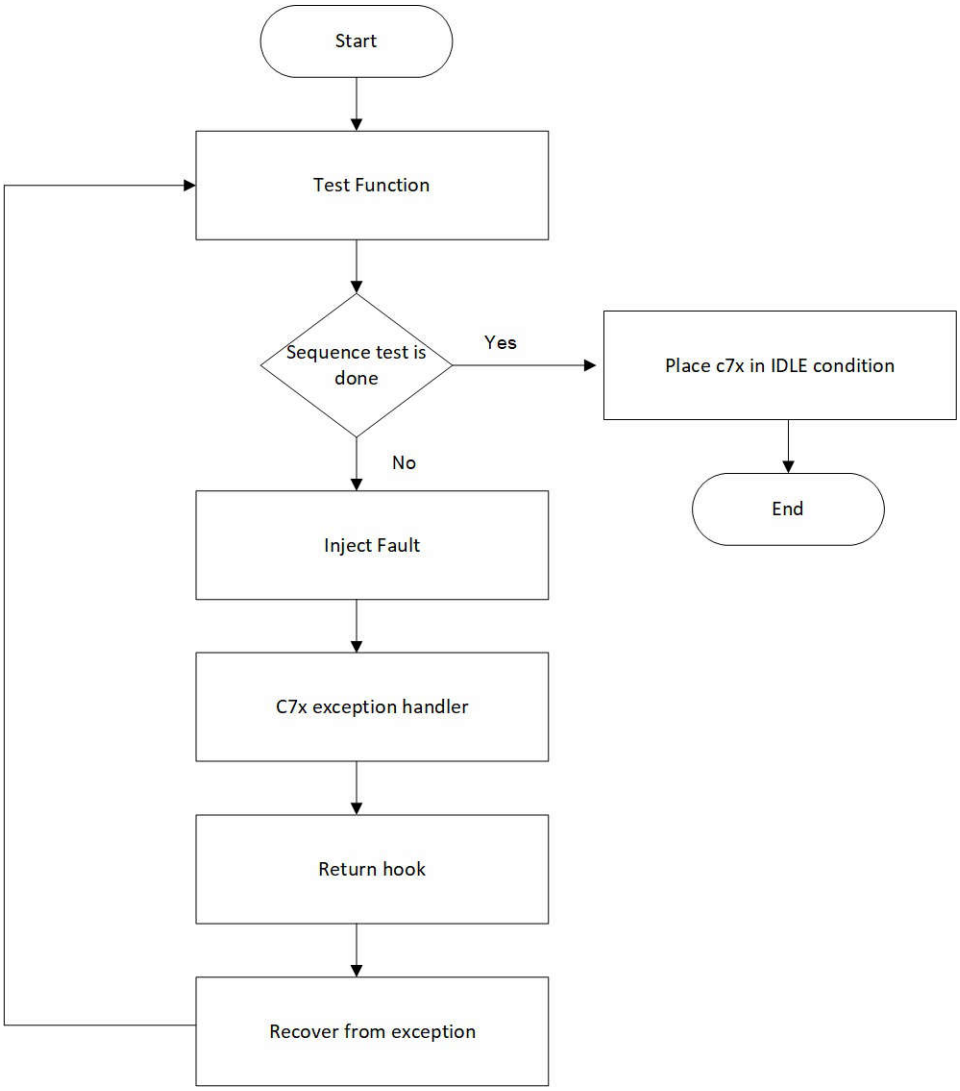

図 3-1. テスト シーケンス フロー図

3.1 コード変更

実装やコードの変更を開始する前に、いくつかの前提条件に従う必要があります。

> リターン フックを登録する必要があります。リターン フックが登録されていない場合、リターン フック関数は例外ハンドラの後に実行されません。

3.1.1 リターン フックの定義

登録されたリターン フック関数には、例外ハンドラから復帰するために以下のアセンブリ コードを含める必要があります。ただし、これはあくまでシーケンス テスト用です。通常のフローでは、例外のシーケンスが生成されるため、これを実行しないでください。

```
void reg_return_hook (void)
{
    __asm(" addkpc.d1 $PCR_OFFSET(test_fun), a0"); //Transfers the //test function
//address to a0.
    __asm(" ldd.d1 *ECSP[1], a1");//Status is saved in a1
    __asm(" rete.s1 a0,a1"); //rete, returns from the exception //to the beginning of the test
function, as //a0 contains the test //function address
}
```

3.1.2 MMA クリア関数

この関数は MMA の内部状態をクリアし、次のテストを実行できるよう準備します。

```
void clear_mma(void)
{
     __HWA_CONFIG_REG_v1 mma_config_reg;
     mma_config_reg = __gen_HWA_CONFIG_REG_v1();
     mma_config_reg.PARITYCTRL = __MMA_NORMAL;
     __HWA_OFFSET_REG offset_reg;
     offset_reg = __gen_HWA_OFFSET_REG();
     __HWAOPEN(mma_config_reg,offset_reg,__MMA_OPEN_FSM_RESET);
     __HWAADV();
     __HWAADV();
     __HWAADV();
     __HWAADV();
     __HWACLOSE(0); //Clearing the first error code
}
```

3.1.3 シーケンス テスト

次の関数は、MMA エラー注入テストのシーケンスがあることを検証します。次のコードは、エンド実装がユーザーによって完全に依存しているサンプルです。

```
volatile unit8_t track_exception = 0;
void test_fun(void)
{
    switch(track_exception)   /*track exception is a global variable ,        initialized to zero*/
    {
        case 0:
            clear_mma(); /*this cleares the previous mma error code, definition is given above*/
            track_exception = track_exception+1;//increment track //exception
            overflow_exception();//generate exception
            break;
        case 1:
            clear_mma();
            track_exception = track_exception+1;
            appLogPrintf("Under flow exception\n"); /*Only for debug, remove*/
            underflow_exception(); // generate exception
            break;
        case 2:
            clear_mma();
            track_exception = track_exception+1;
            appLogPrintf("offset parity exception\n"); /*only for debug remove*/
            offset_parity_test();//generate excpetion
            break;
        case 3:
```

```
            clear_mma();
            appLogPrintf("config parity exception\n"); /*Only for debug*/
            track_exception = track_exception+1;
            config_parity_test();
            break;
        case 4:
            clear_mma();
            track_exception = track_exception+1;
            appLogPrintf("offset parity exception\n");/*Only for debug*/
            offset_parity_test();
            break;
        case 5:
            appLogPrintf("Signal ESM\n");
            appLogPrintf("Waiting for ESM event\n");
            while(1) /*it needs to wait, if not the program doesnt go , and it would generate
multiple exceptions*/
            {

            }
            break;
        default:
    //Handle any unlikely scenario
            break;
}
    return;
}
```

4 まとめ

本書は、c7x DSP コア上での MMA 故障注入の手順および例外処理に焦点を当てています。

この文書では、エラー注入のシーケンスが実行された後にコードの通常のフローを実行するための例外の回復については説明していません。

5 参考資料

- テキサス インスツルメンツ、*C71x DSP CPU、命令セット、行列乗算アクセラレータ*、ユーザー ガイド。
- テキサス インスツルメンツ、*J721S2、TDA4AL、TDA4VL、TDA4VE、AM68A テクニカル リファレンス マニュアル*、テクニカル リファレンス マニュアル。

重要なお知らせと免責事項

テキサス・インスツルメンツは、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、テキサス・インスツルメンツ製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した テキサス・インスツルメンツ製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている テキサス・インスツルメンツ製品を使用するアプリケーションの開発の目的でのみ、テキサス・インスツルメンツはその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。テキサス・インスツルメンツや第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、テキサス・インスツルメンツおよびその代理人を完全に補償するものとし、テキサス・インスツルメンツは一切の責任を拒否します。

テキサス・インスツルメンツの製品は、テキサス・インスツルメンツの販売条件、または ti.com やかかる テキサス・インスツルメンツ製品の関連資料などのいずれかを通じて提供する適用可能な条項の下で提供されています。テキサス・インスツルメンツがこれらのリソースを提供することは、適用されるテキサス・インスツルメンツの保証または他の保証の放棄の拡大や変更を意味するものではありません。

お客様がいかなる追加条項または代替条項を提案した場合でも、テキサス・インスツルメンツはそれらに異議を唱え、拒否します。

郵送先住所:Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

重要なお知らせと免責事項

TI は、技術データと信頼性データ (データシートを含みます)、設計リソース (リファレンス デザインを含みます)、アプリケーションや設計に関する各種アドバイス、Web ツール、安全性情報、その他のリソースを、欠陥が存在する可能性のある「現状のまま」提供しており、商品性および特定目的に対する適合性の黙示保証、第三者の知的財産権の非侵害保証を含むいかなる保証も、明示的または黙示的にかかわらず拒否します。

これらのリソースは、TI 製品を使用する設計の経験を積んだ開発者への提供を意図したものです。(1) お客様のアプリケーションに適した TI 製品の選定、(2) お客様のアプリケーションの設計、検証、試験、(3) お客様のアプリケーションに該当する各種規格や、その他のあらゆる安全性、セキュリティ、規制、または他の要件への確実な適合に関する責任を、お客様のみが単独で負うものとします。

上記の各種リソースは、予告なく変更される可能性があります。これらのリソースは、リソースで説明されている TI 製品を使用するアプリケーションの開発の目的でのみ、TI はその使用をお客様に許諾します。これらのリソースに関して、他の目的で複製することや掲載することは禁止されています。TI や第三者の知的財産権のライセンスが付与されている訳ではありません。お客様は、これらのリソースを自身で使用した結果発生するあらゆる申し立て、損害、費用、損失、責任について、TI およびその代理人を完全に補償するものとし、TI は一切の責任を拒否します。

TI の製品は、TI の販売条件 、 TI の総合的な品質ガイドライン 、 ti.com または TI 製品などに関連して提供される他の適用条件に従い提供されます。TI がこれらのリソースを提供することは、適用される TI の保証または他の保証の放棄の拡大や変更を意味するものではありません。 TI がカスタム、またはカスタマー仕様として明示的に指定していない限り、TI の製品は標準的なカタログに掲載される汎用機器です。

お客様がいかなる追加条項または代替条項を提案する場合も、TI はそれらに異議を唱え、拒否します。

最終更新日：2025 年 10 月