STIL言語テストベクターフォーマット（簡略版）

クレデンスD10は、被試験デバイス（DUT）に適用するテストベクターをSTIL（「スタイル」と発音）言語で定義しています。STILファイル（同じ物理ファイル内で定義できるため、ブロックとも呼ばれる）には、テストピン、電圧仕様、タイミング仕様、テストパターンが記述されています。

- 信号ブロック
  - ピン信号の定義 - 注：各ピンに対して1つだけ定義することができます。
- タイミングブロックと波形表
  - 信号のタイミングセットと波形のフォーマットを定義します。
- DCLevelsブロック
  - 被試験デバイスに印加され、またそこから期待されるDC信号レベル（VIL/VIH, VOL/VOH）を定義する。
- パターンブロック
  - テストベクターを定義します。これは、一定の時間周期でタイムスライスされます（イベント駆動ではありません）。各ベクター内のデータは、その固定時間帯における個々のピンの動作を指定します。

これらの個々のパラメータセットは、テストベクターのセットを実行するために必要なパラメータの完全なセットを定義するために、他のSTILフォーマットブロックによって参照されるようにリンクされています。

- 信号グループブロック
  - バスや制御信号などの各種端子をまとめる。
- パターンバースト・ブロック
  - 1つまたは複数のパターンブロックを一連のパターンにリンクさせる
- パターンエグゼックブロック
  - バーストブロックを必要なDCレベルやタイミングセットでつなぎます。

IDD、VOL、VOHの測定に使用するテストベクター位置。

デバイスピンを所定の状態（VOL/VOH/Tri-State）で測定するためには、テストパターンを一定の位置で停止し、ATE測定器を使用して適切なパラメータを測定する必要があります。これを容易にするため、テストベクター内でこれらの測定を行うべき場所には、プログラムラベルとそれに付随する説明のテキストコ メントが表示されます。(このドキュメントの最後にある例を参照してください）。

注：キープアライブクロックを必要とするダイナミックデバイスについては、Micross Engineeringにお問い合わせください。

ディーン・クラックネル
英国ハンツ州アルトンの
オメガパーク、オリエル

・コート。GU34 2YT Tel :
+44 (0)1420 594180

信号ブロック

シンタックス

```
信号        ( SigName ( In | Out | InOut | Supply | Pseudo ) ( ;|
            ( { ( ScanIn | ScanOut ) ;} ) ) )*
( WFCMap {
                        ( from_wfc -> to_wfc( to_wfc )+ ;* )
                        ( from_wfc1 from_wfc2 -> to_wfc ; )*。
                        } )
            )*)*
}
```

SigName = 信号名を表す文字列 In = 信号を入力として定義する
Out = 信号を出力として定義する
InOut = 双方向の信号として定義する
Supply = 電源またはグラウンド信号を定義する
Pseudo = 信号がデバイスピンでない場合に使用
ScanIn = 信号をスキャン入力として定義
ScanOut = 信号をスキャン出力として定義

例

```
信号
            "pc4" InOut;
            irq InOut;
            scan0 In { ScanIn; }.
}
```

SignalGroupsブロック

SignalGroupsブロックは、0個以上の信号グループへの名前付き参照を作成するために使用されます。以下の例に示すように、Diamond SeriesではSignalGroupsブロックのサブセットがサポートされています。

シンタックス

```
SignalGroups { (シグナルグループ)
 ( GroupName = '( SigName | GroupName ) ( + ( SigName | GroupName ) )*' ;|)
                        ( WFCMap {
                        ( from_wfc -> to_wfc( to_wfc )+ ;*)
                        ( from_wfc1 from_wfc2 -> to_wfc ; )*。
                        } )
          )*)*
}
```

GroupName = 信号グループ名を表す文字列。SigName = シグナル名を表す文字列。

WFCMap = ブロックは、WaveformCharacter の値を他の WaveformCharacter にマッピングすることができる。

FROM_WFC = 別の WaveformCharacter にマップされる WaveformCharacter。FROM_WFC1 = 別の WaveformCharacter にマップされる最初の WaveformCharacter。

FROM_WFC2 = 別のWaveformCharacterにマップされる2番目のWaveformCharacter。

TO_WFC = WavefromCharacter の代用として使用する WaveformCharacter またはそのリスト。

例

```
SignalGroups { (シグナルグループ)
          "ポルタ" = ` "pa7" + "pa6" + "pa5" + "pa4" + "pa3" + "pa2" + "pa1"
          + "pa0" ` ;
          "portb" = ` "pb7" + "pb6" + "pb5" + "pb4" + "pb3" + "pb2" + "pb1"
          + "pb0" ` ;
          all = `porta + portb';
          A = `pa0+pa1+pa2+pa3'
          {.
                    WFCMap {
                              z->x; //シングルWFCマッピング
                              01->x;//two-WFC マッピング (Ⓐの存在が必要)
                    }// WFCMapの終了
          }// end A
}
```

Timing BlockとWaveformTable Blocks

タイミングブロックは、タイミングエッジの配置と、Vector文の信号に波形文字を適用して参照される周期的な波形のフォーマットを定義します。各ベクターは一定の周期で発生し、その周期内での立ち上がりエッジと立ち下がりエッジの配置は、ピン毎またはピングループ毎に定義することが可能です。

Timingブロックは1つ以上の**WaveformTable**ブロック（Wfts）を含むことができ、各ブロックは Periodと**Waveformsの**コレクションから構成されています。各波形はイベントのセットで構成され、それぞれがエッジの配置を持つ。

シンタックス

```
Timing TimingName { (タイミングネーム)
        { WaveformTable TableName { }
                周期 TimeExpr ;
                波形
                        (SigName {
                                (WFC)* {
                                        ( TimeExpr (Event ( | Event)* ); )*。
                                } )*
                        } )*
                }
        }
}
```

TimingName = Timingブロックの名前を表す文字列です。TableName = WaveformTableブロックの名前を表す文字列。Period = 周期を定義する。
SigName = 信号名を表す文字列。
WFC = パターンデータに使用する波形文字を定義します。波形文字は、[0-9]、[a-z]、[A-Z]の英数字1文字である必要があります。
TimeExpr = ' (integer | float) (engineering_prefix) (unit) '
Event = =.

- D｜ForceDown-VIL値を強制的に下げます。
- U｜ForceUp（フォースアップ）- VIHの値を強制的に変更します。
- Z｜ForceOff：ドライバモードがHiZのとき、ドライバをOFFにします。ドライバモードがVIHHのとき，強制的にVIHHにする。Pin PMUがconnectByPattern()のとき，Pin PMUの電圧を強制的に変化させる。
- N｜ForceUnknown（フォースアンノウン） - Uフォーマットと同じです。
- L｜CompareLow-VOLより小さい値で比較する。
- H｜CompareHigh-VOHより大きな値で比較する。
- X｜CompareUnknown - 気にしない
- T｜CompareOff - VOLとVOHの間の値を比較します。

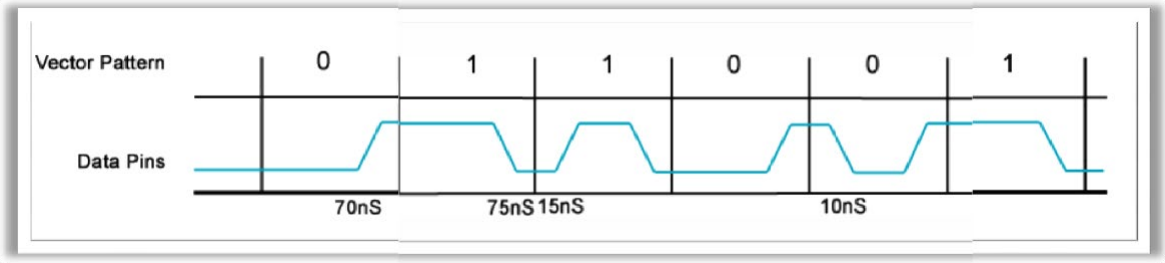
例

```
タイミング "Param_Time_Set"{。
        WaveformTable "T1" {...
                周期'100nS'; 波
                形{.
                        // 機能的導通タイミング
                        ..."Control_pins" { 0 { '45nS' D;
                        }}。
                        「Control_pins" { 1 { '50nS' U; }}。
                        "Data_pins" { 0 { '10ns' D; '70nS' U; }}。
                        「Data_Pins" { 1 { '15nS' U; '75nS' D; }}。
                        「Clock_pins" { 1 { '15ns' U; '75nS' D; }}。
                        「Clock_pins" { 0 { '15ns' D; }}。
                }
        }
}
```

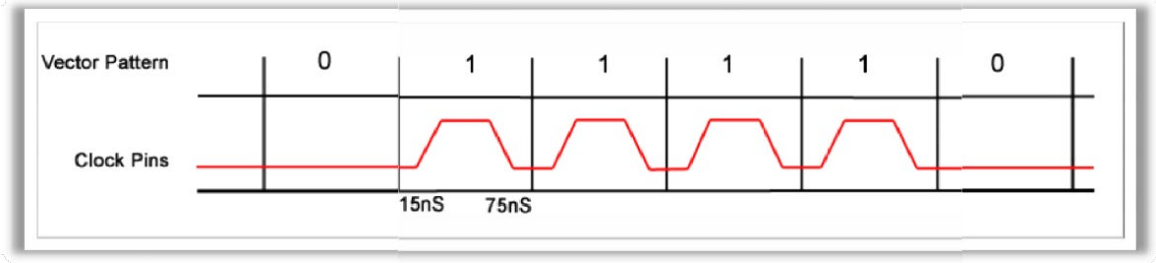
は、次のような波形を生じさせるだろう。

1: ピングループ "Control_pins" は、グループ内の各ピンのベクターパターンが "1" の時、50nS で High になり、前のベクターが既に High であった場合はエッジは発生しません。そのピンのパターンに "0 "がある場合、ピンは45nSでローになり、前のベクトルがすでにローであった場合はエッジは発生しません。この時間形式は、Non-return To Zero (NRZ) と呼ばれ、ピンは次の遷移まで指定された状態に留まります。

2: ピングループ "Data_pins "は各ベクター内で状態を変化させます。Surround by ComplimentやXOR形式のバリエーションで、アドレスバスやデータバスでセットアップやホールドのタイミングを取るのに有効です。



3: ピングループ "Clock_pins "は、パターンが "1 "の時にパルスを発生し、"0 "の時にはLowのままです - このフォーマットは、Return To Zero (RZ) として知られているものです。



DCLevelsブロック

注：このブロックは、お客様が提供された情報をもとにMicrossが作成します。

DCLevels ブロックは、各テストパターンの信号に適用される DC レベルを定義します。

DCLevels ブロックは、DC パラメータに名前を付け、一連の信号に対する特定の DC データを定義します。DCLevels ブロックの各ステートメントには、1 つの DC パラメータの特性が指定されます。各 DC パラメータは、1 つの DCLevels ブロックで、特定の信号に対して 1 回のみ指定することができる。同じブロック内の信号は一意でなければならず、信号の重複は許されない。API は DCLevels ブロックの信号を参照して DCLevels を設定することができる。

シンタックス

```
(DCLevels (DCLevelsName) { // DCLevelsブロック
        (SignalName {)
                (VIH (dc_expr)+;)
                (VIL
                (dc_expr)+;)(VIHH
                (dc_expr)+ ;
                )(VIHH(dc_expr))
                。(VOH
                (dc_expr)+;)
                (VOL (dc_expr)+;)
        })*
})*
```

DCLevelsName = DCLevelsブロックの名前を表す文字列です。SignalName = 信号名または信号グループ名を表す文字列。
VIH = 駆動高電圧を指定します。VIL = ドライブ低電圧を指定します。
VIHH = オルタネート駆動電圧を指定します。VOH = コンペア高電圧を指定します。
VOL = コンペア低電圧を指定します。
dc_expr = ' (integer | float) (engineering_prefix) (unit) ' 例

```
DCLevels dc_func {
        インス
                VIH「vih1」
                、VIL「vil1
                」。
        }
}
```

パターンブロック

パターンブロックは、パターンベクターデータを定義します。構文

パターン *PatName* {
(*WaveformTableDeclaration* | *FunctionalVector*)*。
}
PatName=ident
WaveformTableDeclaration = (W | WaveformTable) *ident*
FunctionalVector = (*ParallelVector* ／*LoopVector* ／*MatchVector* ／*Macro* ／*Procedure*)
Vector = (F | V) { *SigName* = (*WFC*)+ ;
}.*条件* = C { *SigName* = (*WFC*)+ ; }.*WFC*
= *(文字|数字*|#|%*)*
}

例

パターン例

```
W _default_WFT_;
        label:v{ all = 00000000 00000000 00000000 00000000; }.
        v { all = 11111111 111111 111111 111111; }.
ループ          10 {//マルチベクトルループとして実装 V {
        ALL = 11111111 111111 11111111; }.V { ALL =
        HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
        HHHHHHHHHHHHHHHH} V { ALL = 00000000 00000000
        00000000 00000000; }.v { all = llllllll llllllll
        llllllll; }.
}
ループ5                                 {//繰り返しベクトルとして実装V { ALL
        = 11111111 111111 111111 111111; }.
}
停止する。
}
```

パターンバースト
・ブロック

注：このブロックは、お客様が提供された情報をもとにMicrossが作成します。

PatternBurst ブロックは、テスト中に実行されるパターンのシーケンスを定義します。構文
PatternBurst *BurstName* { パターンバースト
パットリスト
(*PatName* ;)*)
}
BurstName = パターンバースト名を表す文字列です。
PatName = パターンブロックの名前を表す文字列。

例

```
PatternBurst "_burst_" {
    PatList {.
        "first_pattern "です。
    }
}
```

パターンエグゼックブロック

注：このブロックは、お客様が提供された情報をもとにMicrossが作成します。

PatternExecブロックは、テスター上でパターンを実行するために必要なすべての部品を定義する「接着剤」です。このブロックで定義されるのは

- Spec 変数の解決に使用する Category 名を指定します。
- は，仕様変数のどの値（Min，Typ，Max，Meas）を適用するかを示すセレクタ名である。
- PatternExecを実行する前に設定するDCLevelsブロック。
- PatternExecを実行するときに使用するAPGSteeringブロック。
- WaveformTableの参照を見つけるためのTimingブロック。
- と、使用するPatternBurstを指定します。

参照されている Timing または DCLevels ブロックに複数のカテゴリを持つ spec 変数が含まれている場合、PatternExec ブロックに 1 つ以上の Category 文を指定する必要があります。Timing または DCLevels ブロックが複数の値（すなわち、Min、Typ、または Max 値）を含む spec 変数を参照する場合、変数は、Selector ブロックによってどの値を適用するかを解決するか、または参照の変数名を修飾する（たとえば `var.Min' など）ことによって、明確な方法で指定する必要があります。指定されたTimingブロックは、参照されるすべてのPatternブロックにおいて参照されるすべてのWaveformTable名を解決しなければならない。

PatternExecブロックの各項目はオプションです。Spec Category、SelectorとTiming、DC Levelsのバインドに使用することができます。パターンバーストが含まれている場合は、パターンバースト内の各パターンを実行します。

シンタックス

```
パターンチェック (PAT_EXEC_NAME)
            ( カテゴリ CATEGORY_NAME ;
            )* ( セレクタ SELECTOR_NAME ;
            )* ( セレクタ SELECTOR_NAME ;
            )* ( セレクタ SELECTOR_NAME ;
            )
            ( APGSteering APG_STEERING_NAME; )
            ( DCLevels (DC_LEVELS_NAME;) )
            ( Timing TIMING_NAME ; )
            ( PatternBurst PAT_BURST_NAME ; )
}
```

CATEGORY_NAME = カテゴリーブロック名を選択します。PatternExecブロックは複数のカテゴリーを持つことができますが、各カテゴリー内の変数は一意でなければなりません。
SELECTOR_NAME = セレクターブロック名を選択します。PatternExecブロックは複数のセレクタを持つことができますが、各セレクタの変数は一意でなければなりません。
APG_STEERING_NAME = DUTピンをAPGリソースに割り当てるAPGSteeringブロック名を選択します。
DC_LEVELS_NAME = DCLevels ブロックを選択します。PatternExecブロックには、0個または1個のDCLevels クロックを含めることができます。
TIMING_NAME = タイミング・ブロックを選択します。PatternExecブロックは、0個または1個のTimingブロックを含むことができます。PAT_BURST_NAME = 実行するPatternBurstブロックを選択します。PatternExecブロックは0個または1個のPatternBurstブロックを含むことができる。

例

```
PatternExec "FuncExec" { (パターンエグゼック)
          DCLevels "DCLevels"、
          Timing "Timing"、
          PatternBurst "FuncBurst";
}
```

簡単なSTILの例

```
STIL 1.0;
信号
          DIR In;
          OE_ In;
          A0 In; A1 In; A2 In; A3 In;
          A4 In; A5 In; A6 In; A7 In;
          B0 Out; B1 Out; B2 Out; B3 Out;
          B4 Out; B5 Out; B6 Out; B7 Out;
}
SignalGroups { (シグナルグループ)
          abus='a7 + a6 + a5 + a4 + a3 + a2 + a1 + a0';
          bbus='b7 + b6 + b5 + b4 + b3 + b2 + b1 + b0';
          all ='dir + oe_ + abus + bbus' となる。
}
タイミング "basic_timing"{。
          WaveformTable "1"{。
                    周期'500ns'; 波
                    形{.
                              DIR { 01 { '0ns' D/U; }}。
                              OE_ { 01 { '0ns' U; '200ns' D/U; '300ns' U; }}
                              。ABUS { 01 { '10ns' D/U;}}。
                              BBUS { HLZ { '0ns' Z; '0ns' X; '260ns' H/L/T; '280ns' X;}} 。
                              }
                    }// end WaveformTable one
}// end タイミング "basic_timing"

PatternBurst "pat1_burst" {。
          PatList { "pattern_1";
}// end PatternBurst "pat1_burst"

パターンエグゼック
          Timing "basic_timing";
          PatternBurst "pat1_burst";
}//end PatternExec

パターン " pattern_1" {.
          W「1」。
LAB000です。V { ALL=0000000000LLL; }.// ここで全ての VOL を測定
LAB001: V { ALL=0010000000HLLLLL; }.// ここでB7のVOHを測定
          v { all=00010000001hllll; }.v
          { all=00001000001lhlll; }.v {
          all=0000010000lllhll;  }.v  {
          all=0000001000llhlll;} 。 v  {
          all=0000000100lllhll;} 。 v  {
          all=0000000010llllhl; }.
LAB008です。V { all=00000000011lllllh; }.// ここでB0 VOHを測定する
停止する。
}// 終了 パターン " pattern_1 "
```