VisualSim モデル説明書

2010/11/26

ベリフィケーションテクノロジー株式会社

目次

1. 変更･改訂･確認･廃止歴

変更･改訂･確認･廃止歴			
日付	Ver	備考	担当
2010/11/	1.0	新規作成	Vtech

2. 概要

2.1 適用

本書は、TI DSP TMS320DM6437 をユーザーマニュアルをベースに VsiaulSim でモデル化を行い、ご要求のデータフローのシミュレーション環境及びモデルの説明を行う。

2.2 ツール

	ツール	備考
[1]	VisualSim	Version:VisualSim 10 Revision 30 Simulator：VisualSim Architect（モデル作成も同様）

2.3 参考文献

	ドキュメント名	バージョン	発行元
[1]	DSPデータシート：tms320dm6437.pdf	SPRS345D–NOVEMBER 2006–REVISED JUNE 2008	TI
[2]	TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide：42A4590Fd01.pdf	Literature Number: SPRU732J July 2010	TI

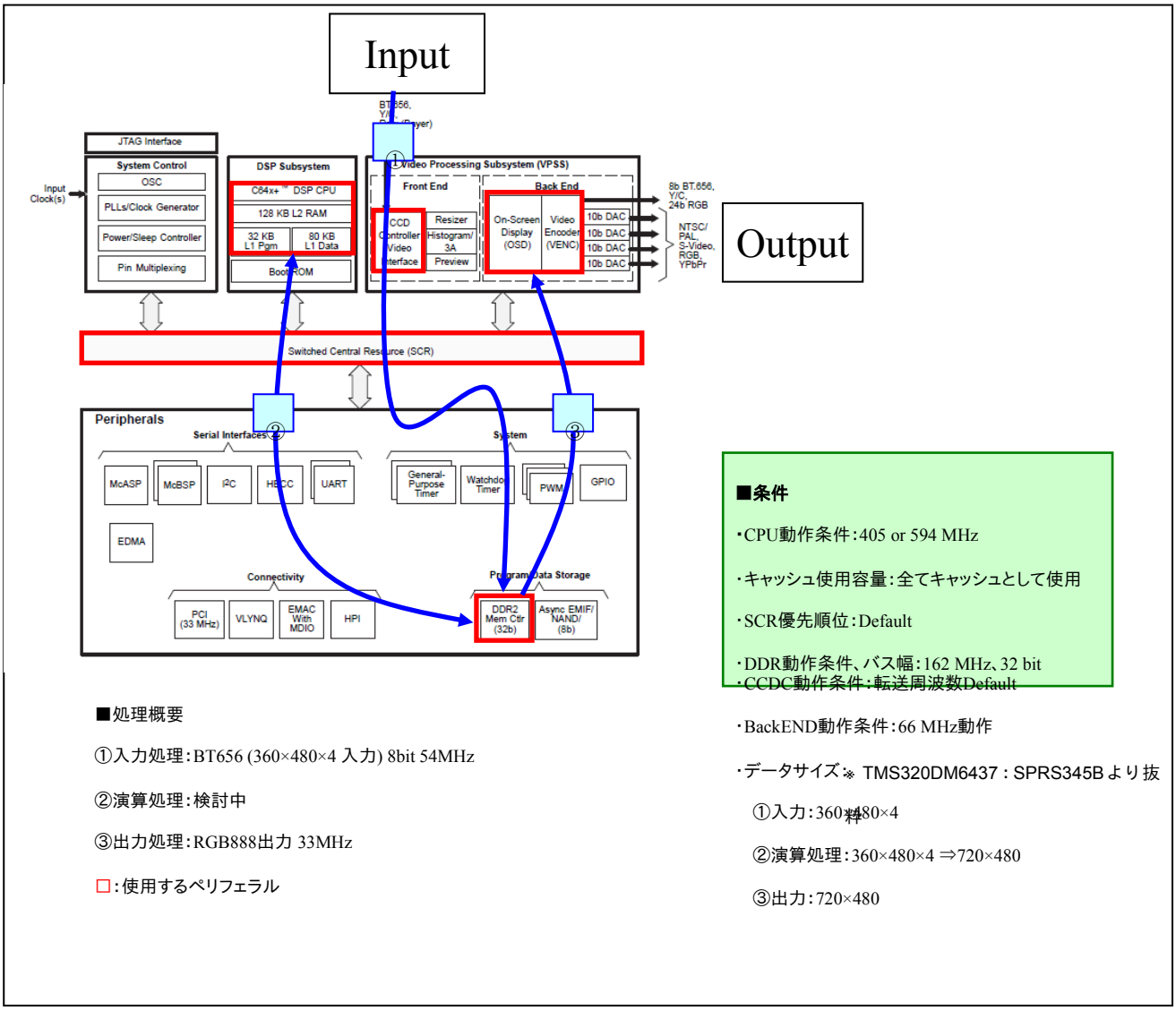
2.4 環境

	名称	備考
[1]	OS	Windows NT/2000/XP/Vista/7、Sun SPARC、Linux、Mac OS X
[2]	Java	Java SE SDK 1.6.0

3. VisualSim モデル仕様

3.1 ブロック図及びデータフロー

下記に、モデルの要求仕様を示す。



■処理概要

①入力処理：BT656 (360×480×4 入力) 8bit 54MHz

②演算処理：検討中

③出力処理：RGB888出力 33MHz

□：使用するペリフェラル

■条件

・CPU動作条件：405 or 594 MHz

・キャッシュ使用容量：全てキャッシュとして使用

・SCR優先順位：Default

・DDR動作条件、バス幅：162 MHz、32 bit

・CCDC動作条件：転送周波数Default

・BackEND動作条件：66 MHz動作

・データサイズ※ TMS320DM6437：SPRS345Bより抜

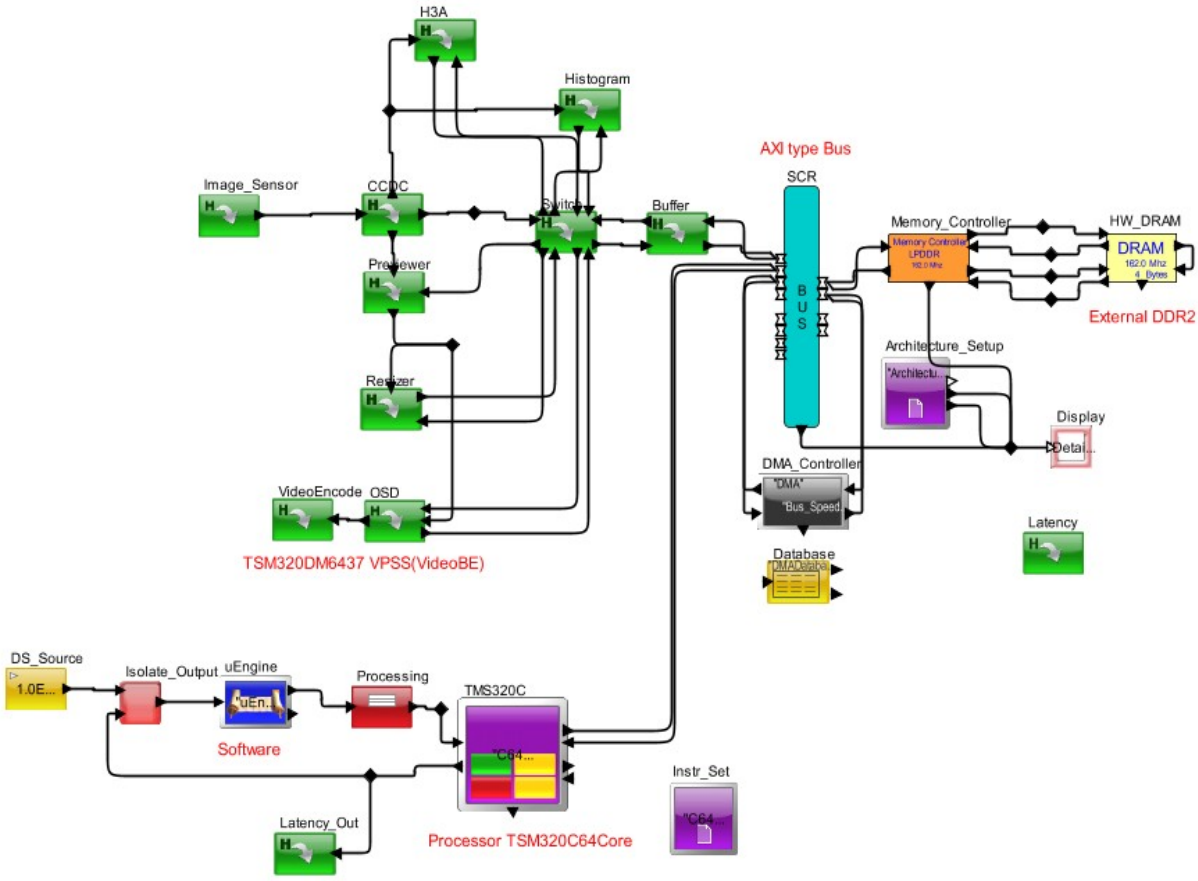
①入力：360×480×4

②演算処理：360×480×4 ⇒720×480

③出力：720×480

3.2　モデルの構成図

下図に、要求仕様と TI ユーザーマニュアルを基に作成した VsiaulSim のシステム検証モデルを示す。

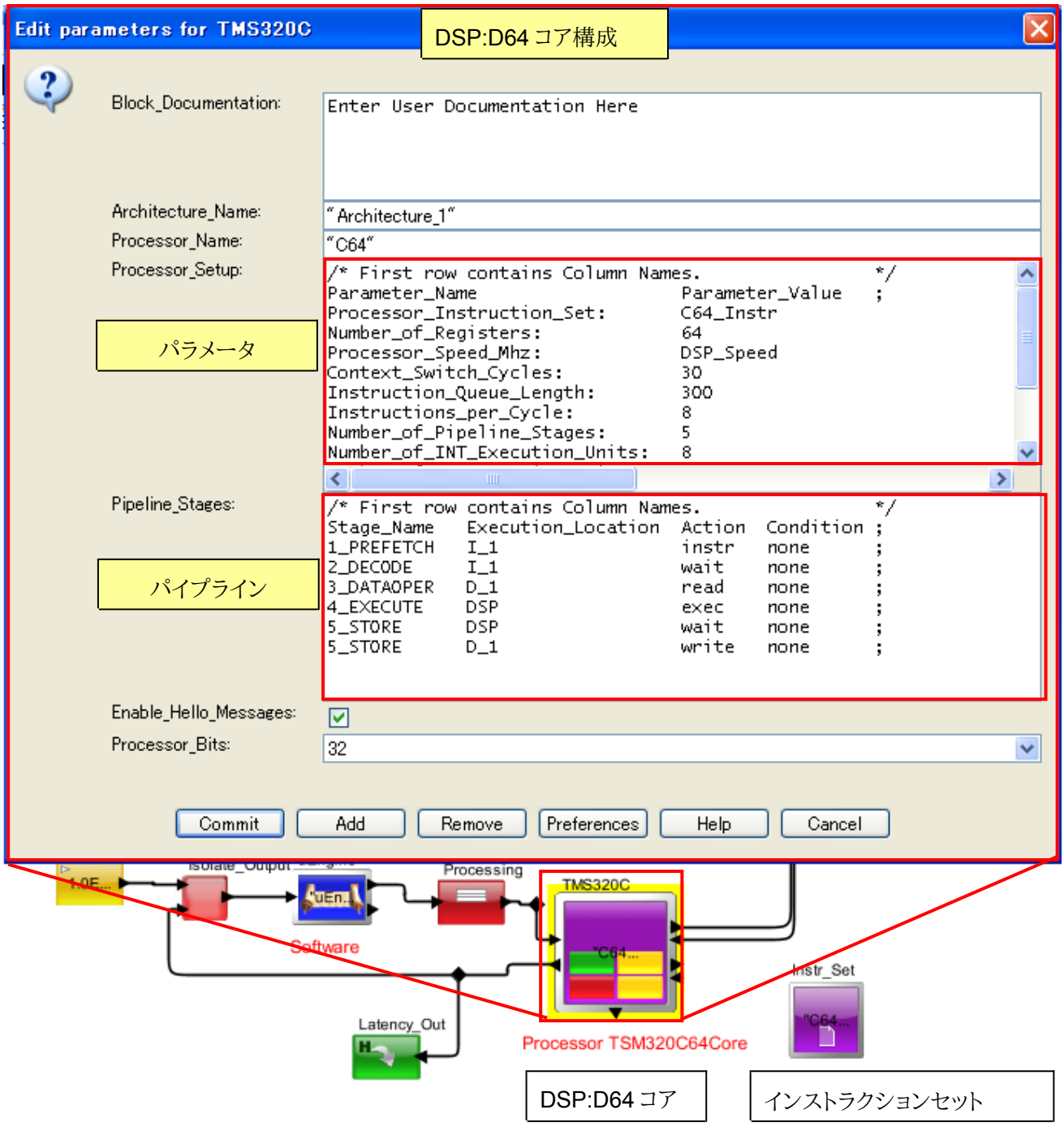


3.3 モデルの各モジュールの概要説明

下記にモデルの構成及び動作の説明をする。

3.3.1 DSP コア

DSP コアは、TMS320C の基本構成を持ったライブラリを使用。



設定されたパラーメータ及び構成及び説明は下記。赤字は説明パラメータ説明。

/* First row contains Column Names. */
Parameter_Name Parameter_Value ;
①Processor_Instruction_Set: C64_Instr :インストラクションセットを指定。
“インストラクションセットブロック”で 指定。“TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide“の命令を使用。
②Number_of_Registers: 64 :C64 コアのレジスタ数を入力
③Processor_Speed_Mhz: DSP_Speed :動作周波数。
変数としてモデルトップで変更可。 現在は、405MHz の設定。
Context_Switch_Cycles: 30 :コンテキストスイッチ切替サイクル。
Instruction_Queue_Length: 300 :入力及び実行可能なタスク数。
Instructions_per_Cycle: 8 :サイクル毎のインストラクションのサポート数
Number_of_Pipeline_Stages: 5 :パイプライン数
Number_of_INT_Execution_Units: 8 :整数演算ユニット数
Number_of_FP_Execution_Units: 0 :浮動小数点演算ユニット数
Number_of_Cache_Execution_Units: 3 :コア内蔵キャッシュ数
以下はキャッシュの詳細。I は命令キャッシュ、D はデータキャッシュ。L は 2 次キャッシュ。
動作周波数、キャッシュサイズ、キャッシュのアクセスワード数、1 ラインのワード数、ミス時に参照するメモリを指定。
I_1: {Cache_Speed_Mhz=DSP_Speed, Size_KBytes=32.0, Words_per_Cache_Access=8, Words_per_Cache_Line=128, Cache_Miss_Name=L2}
D_1: {Cache_Speed_Mhz=DSP_Speed, Size_KBytes=80.0, Words_per_Cache_Access=8, Words_per_Cache_Line=128, Cache_Miss_Name=L2}
L2: {Cache_Speed_Mhz=DSP_Speed, Size_KBytes=128.0, Words_per_Cache_Access=8, Words_per_Cache_Line=128, Cache_Miss_Name=DRAM}
＊DRAM は外部メモリ名。

3.3.2 DSP コア周辺

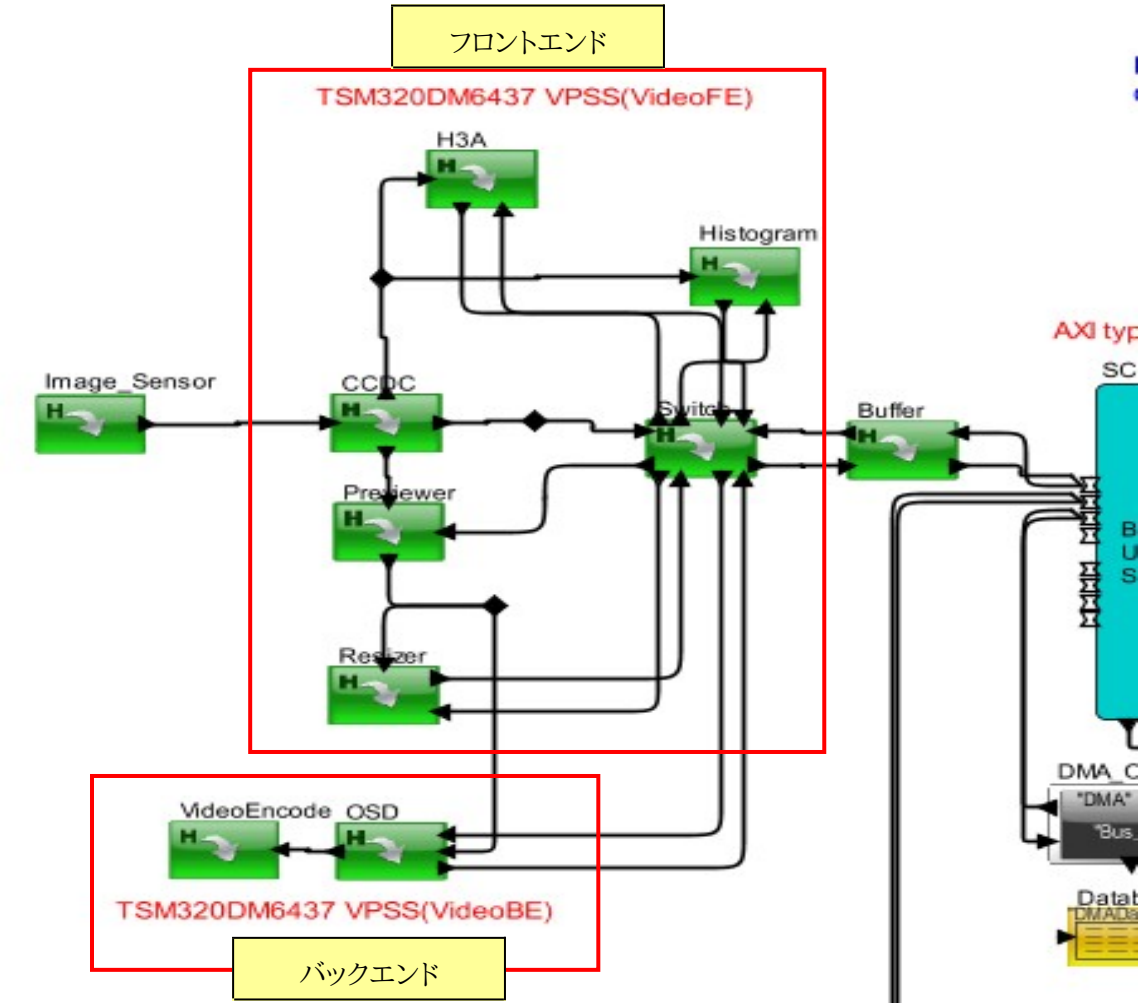
Video フロントエンド及びバックエンドの一部を実装。ブロック名は下記。

1 フロントエンドブロック：CCDC、H3A、Histogram、Previewer、Resizer

2 バックエンドブロック　：OSD、Video Encoder

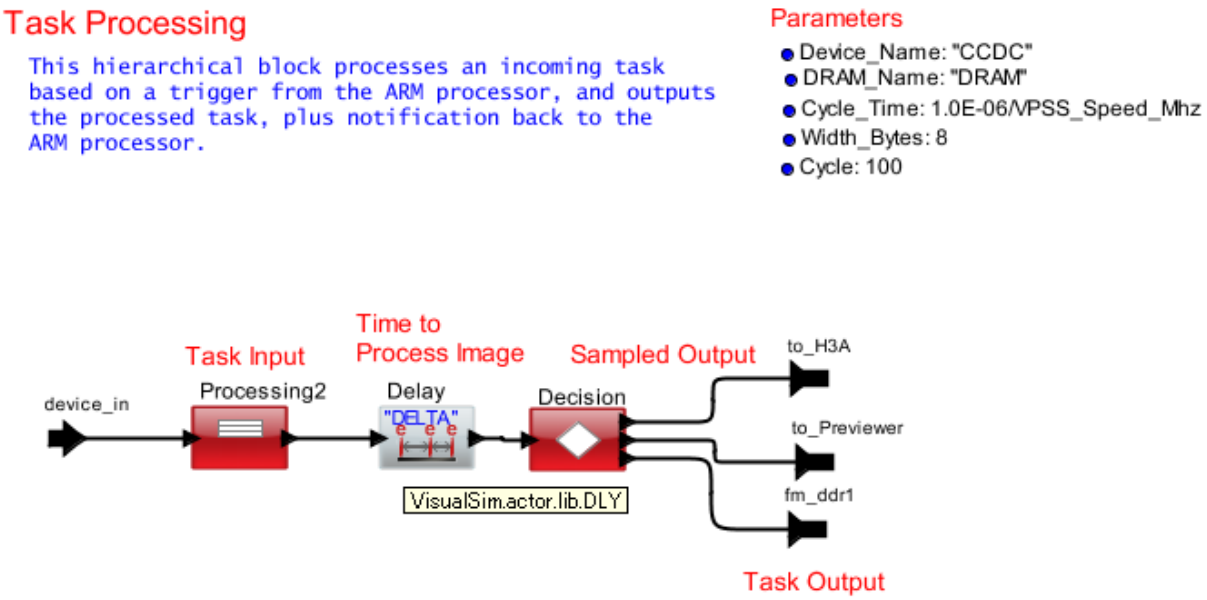
3 その他　　：Image Sensor、Switch、Buffer

下記は、その構成図。



1-1 CCDC

下記に CCDC 内部の構成図を示す。イメージセンサーより、入力されたデータを DDR 及び関係各部へ送る。現在は、要求仕様の通り DDR へ直接ライトするパス設定となっている。



・“Prossing”ブロックは、動作「ライト or リード」の定義、後工程で必要なパラメータ設定などを行う。

設定例：

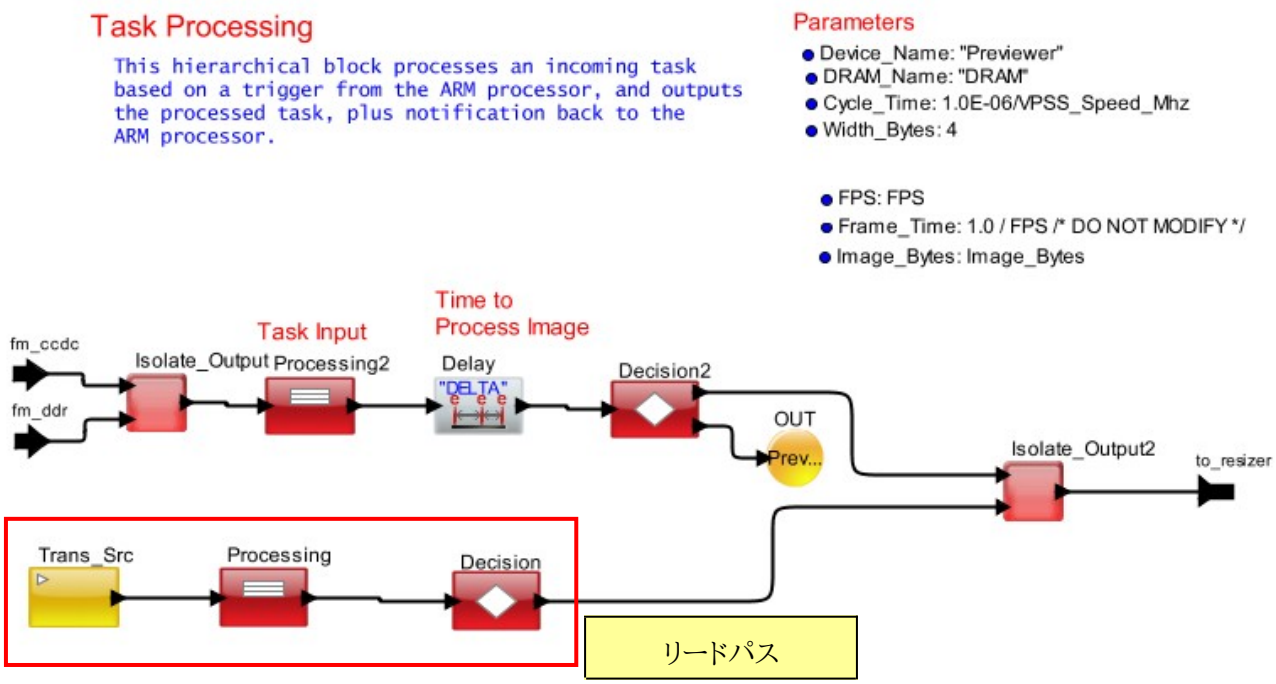
```
/* Template to enter multiple RegEx lines*/
input.DELTA       = Cycle_Time*Cycle
input.A_Task_Name = Device_Name
input.A_Task_Flag = true
input.A_Source = Device_Name
input.A_Destination = DRAM_Name
input.A_Hop = "SCR"
input.A_Command = "Write"
```

・“Delay”ブロックは、このブロックでの処理時間（サイクル数など）の設定を行う。
現在は、上記図の“Cycle”で 100 サイクルの設定。

・“Decision”ブロックは、各設定や演算処理などを加え、結果を各ポートへ出力する。
現在は、入力を DDR へ出力。（fm_ddr1）

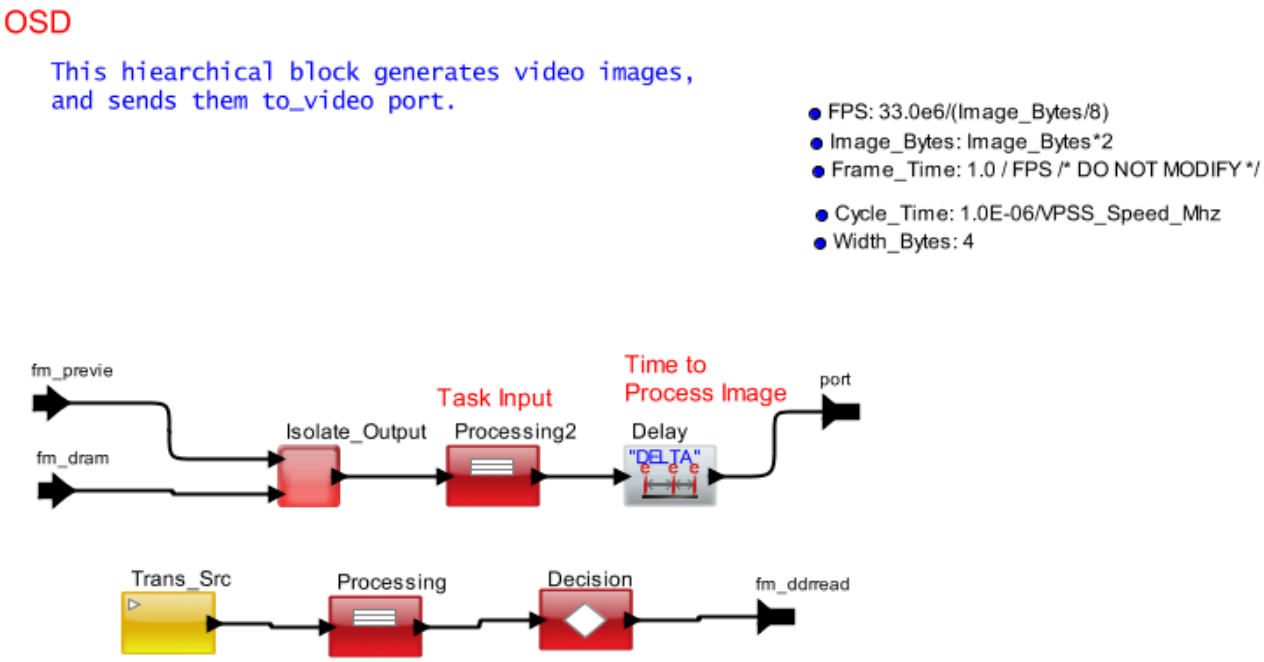
1-2 H3A、Histogram、Previewer、Resizer

下記に H3A、Histogram、Previewer、Resizer 内部の構成図を示す。CCDC より、入力されたデータを処理して DDR へライトするパスと CCDC から直接 DDR へライトしたデータをリードして DDR へライトするパスの 2 通りを実装。現在は、仕様通りのパスでこれらのブロックは無効。（下図は、代表して Previewer を記載）



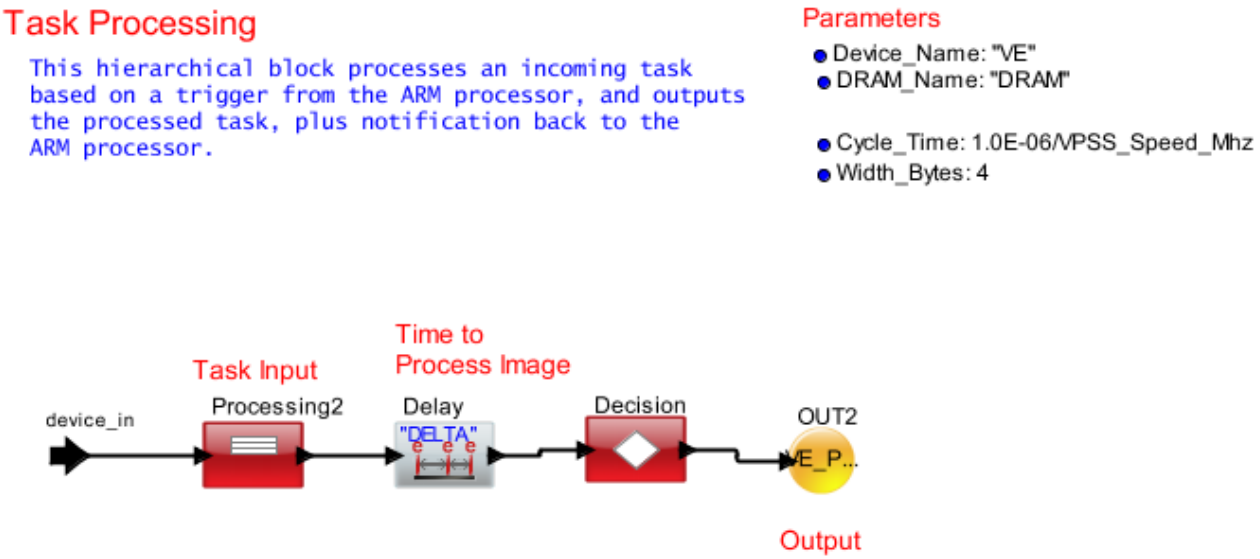
・“Prossing”ブロックは、動作「ライト or リード」の定義、後工程で必要なパラメータ設定などを行う。
・“Delay”ブロックは、このブロックでの処理時間（サイクル数など）の設定を行う。
現在は、上記図の“Cycle”で 100 サイクルの設定。
・“Decision”ブロックは、各設定や演算処理などを加え、結果を各ポートへ出力する。
現在は、入力を DDR へ出力と処理時間の計算を出力。（黄色丸の OUT）
・“Trans_Src”は、リード動作のトランザクションを発生。

下記に OSD 内部の構成図を示す。Previewer より、入力を出力するパスと DDR より、データをリードして出力するパスの 2 通りを実装。現在は、DDR よりリードするパスが有効。



・“Prossing”ブロックは、動作「ライト or リード」の定義、後工程で必要なパラメータ設定などを行う。
・“Delay”ブロックは、このブロックでの処理時間（サイクル数など）の設定を行う。現在は、上記図の“Cycle”で 100 サイクルの設定。
・“Decision”ブロックは、各設定や演算処理などを加え、DDR へリードアクセス。
・“Trans_Src”は、リード動作のトランザクションを発生。

下記に Video Encoder 内部の構成図を示す。OSD より、来たデータを出力。
Video Encoder 処理時間の追加とここまでの処理時間の計算を行う。



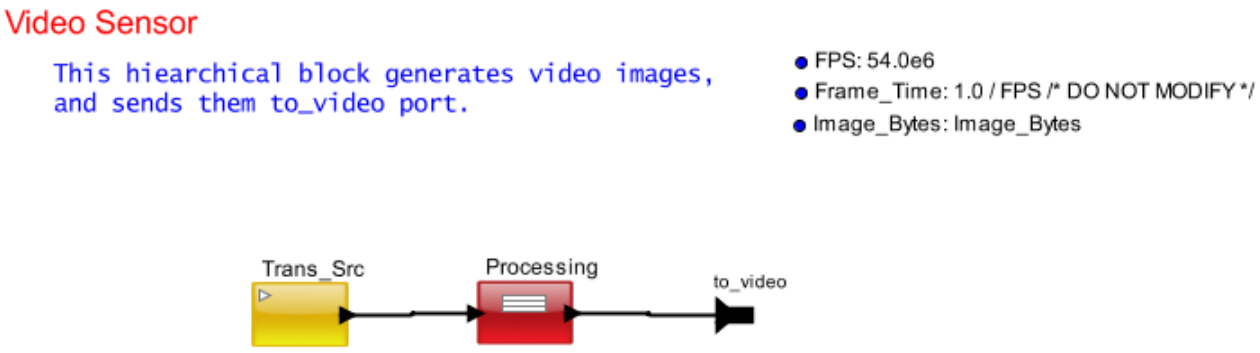
・“Prossing”ブロックは、動作「ライト or リード」の定義、後工程で必要なパラメータ設定などを行う。ここでは、処理時間追加のみ。

```
/* Template to enter multiple RegEx lines*/
    input.DELTA       = Cycle_Time*100.0
    input.A_Task_Name = Device_Name
    input.A_Task_Flag = true
```

・“Delay”ブロックは、このブロックでの処理時間（サイクル数など）の設定を行う。
現在は、上記図の“Cycle”で 100 サイクルの設定。

・“Decision”ブロックは、トータルの処理時間を計算し出力。（OUT2 から出力）

下記に Image Sensor 内部の構成図を示す。イメージセンサのデータを入力する。

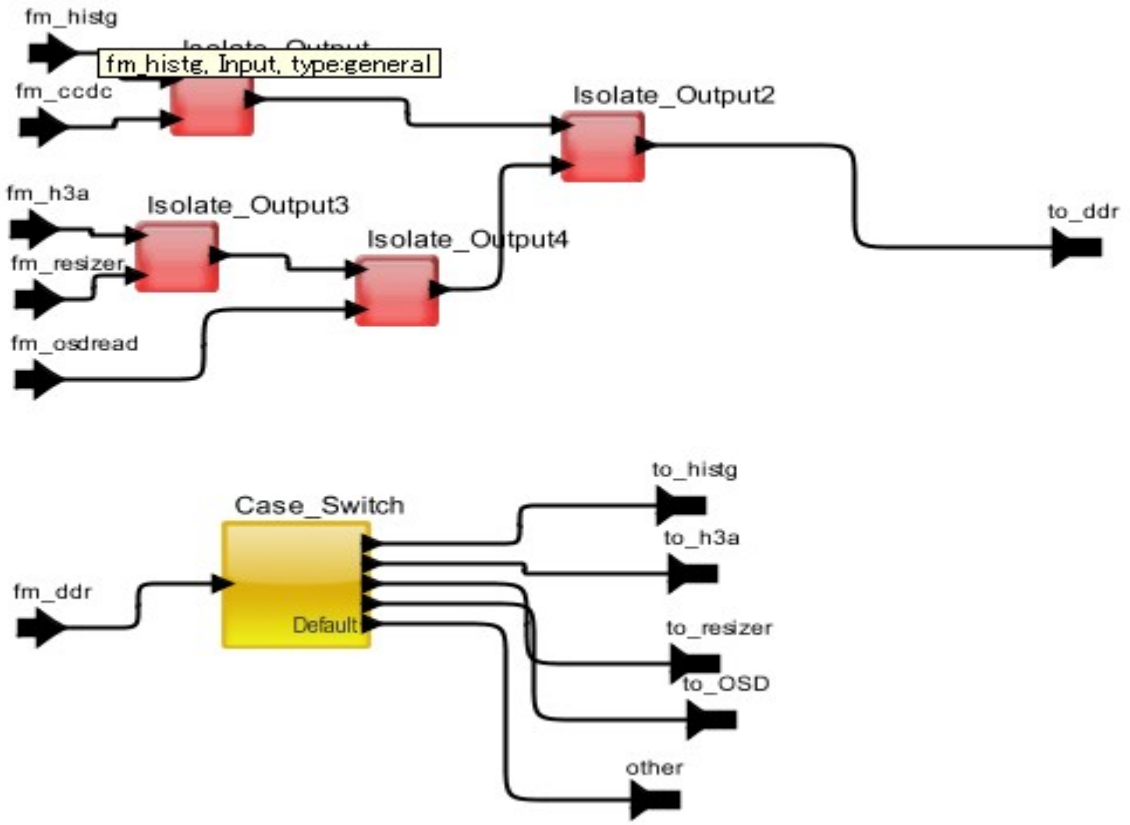


・“Prossing”ブロックは、動作「ライト or リード」の定義、後工程で必要なパラメータ設定などを行う。ここでは、入力データの設定。

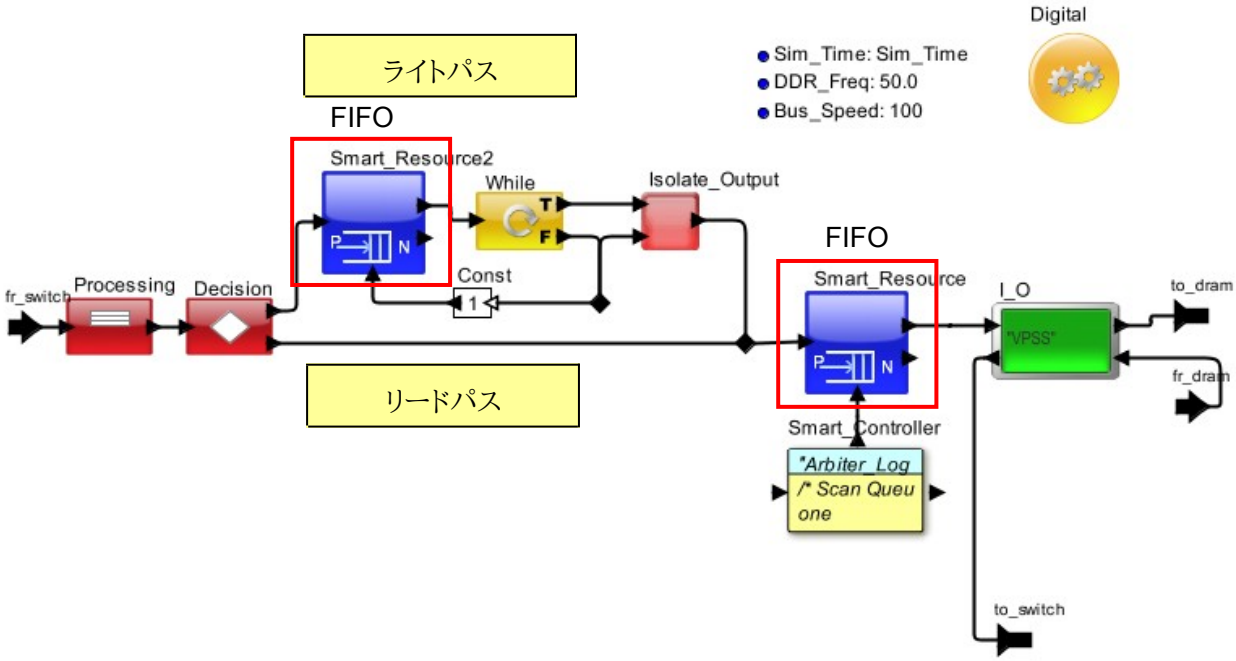
```
/* Template to enter multiple RegEx lines*/
  input.A_Task_Name  = "Image_" + input.ID
  input.A_Bytes      = Image_Bytes
  input.A_Bytes_Remaining = Image_Bytes
  input.Sample_Count = 0
  input.Last_Frag = false
  input.A_Task_Flag = true
  input.EndT = TNow + Frame_Time
  input.My_Time_Stamp = TNow
```

・“Trans_Src”は、データ入力トランザクションを発生。

下記に Switch 内部の構成図を示す。各ブロックのデータを一つにする。
各ブロックからの入力は、来た順に DDR へ出力、DDR からの信号は、行き先により振り分ける。



下記に Buffer 内部の構成図を示す。Bus->メモリ（DDR）のデータを一旦保存し、各ブロックからのアクセスが破綻しないように調整する。 また、バス幅合ったデータ転送を制御する。動作は、ライト及びリード双方に対応。



3.4 DSP サンプルプログラム

提供頂いたアセンブラコードのプログラムより、インストラクションセットを抜き出し、VisualSimで実行可能なフォーマットへ変換。 “uEngine”ブロック（下図参照）で指定。

・“DS_Source”ブロックは、DSP を動作されるイベントの発生。
現在は、初期動作のイベント発生。その後は、DSP が1つのソフトを終了後
次を実行する。

・“Prossing”ブロックは、DSP へ入力するための設定を行う。

```
MyDS  = newRecordToken("Processor_DS")
InDS  = newToken(input)
InDS = merge(InDS, MyDS)
InDS.A_Hop = InDS.Processor_Name
InDS.A_Destination = InDS.A_Hop
InDS.Processor_Name =  InDS.Processor_Name
InDS.A_Source = "Src"
InDS.A_Task_Flag = false
InDS.A_Instruction = InDS.A_Instruction
InDS.Event_Name = "C64"
InDS.TIME = TNow
InDS.A_Variables = 3000
InDS.A_Priority = 10
InDS.Last_Frag = true
```

・“Latency_Out”は、プログラムの実行時間を計算。ブロック構成は下記。
計算は、下記“Decision”ブロックで実行。

・プログラムファイル名：f_vpss_pas.asm、f_vpbe_pas.asm、f_ccdc_pas.asm
・インストラクションセット：下記は f_ccdc_pas.asm の場合例（命令の部分のみ）

```
   "LDW","STW","MVK","STW","LDW","OR","STW","MVC","SET","MVC","ADDK","S
TW",
   "STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","ST
W",
   "STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","ST
W",
   "STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","ST
W",
   "STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","STW","ST
W",
   "STW","STW","STW","MVC","STW","MVC","STW","LDW","LDW","CALLP","LDW
","STW"
   ,"LDW","MVC","LDW","MVC","LDW","LDW","LDW","LDW","LDW","LDW","LDW
","LDW",
   "LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW"
,"LDW","
   LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW",
"LDW","L
   DW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","LDW","
LDW","LD
   W","LDW","LDW","LDW","LDW","LDW","LDW","LDW","ADDK","LDW","STW","S
TW","ST
   W","STW","STW","STW","STW","STW","STW","STW","STW","MVK","STW","STW"
,"STW
   ","STW","STW","STW","STW","MVK","STW","STW","STW","MVK","STW","STW","
STW"
   ,"MVK","STW","LDW","LDW","AND","STW","STW","MVK","STW","STW","MVK",
"STW",
   "STW","STW","STW","MVK","STW","STW","STW","CALLP","CALLP","LDW"
```

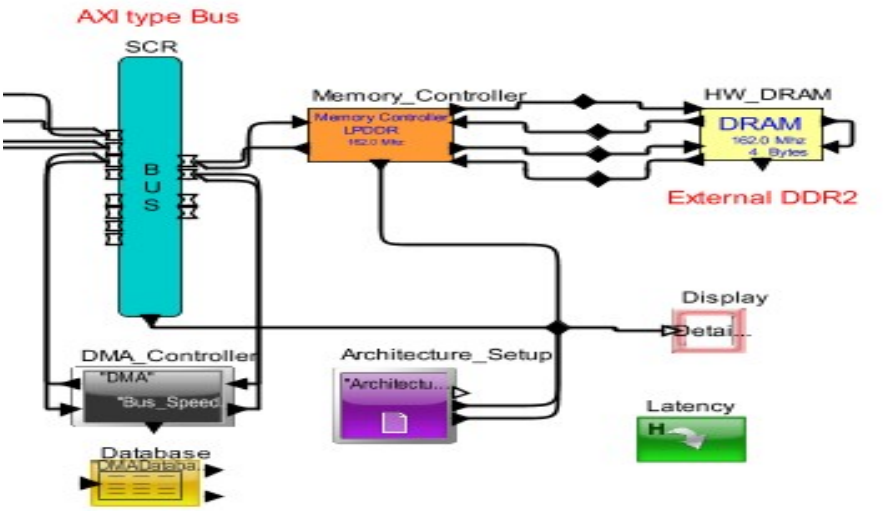
上記を含んだプログラムファイルを“uEngine”ブロックで指定。

ファイルは、作業ディレクトリに保存。連続してプログラムを実行する場合は、
下記の{instruction_set}に上記のようなプログラムを挿入し、プログラムを続ける。
（ファイルはテキスト形式）
例：vpbe―> vpss の場合

```
{Processor_Name="C64_",Processor_ID=1,A_Task_Name=" vpbe ",
        A_Task_ID=1L,A_Priority=0,A_Instruction={instruction_set}}
{Processor_Name="C64_",Processor_ID=1,A_Task_Name=" vpss ",
        A_Task_ID=1L,A_Priority=0,A_Instruction={instruction_set}}
```

3.5 外部ブロック

DSPの以外のブロックの構成を下記に示す。



3

Edit parameters for Database

Database ブロック

Block_Documentation:
```
*.xml, *.csv files abs or rel (./) path
  -- *.csv real columns set to number
Input_Fields == Lookup_Fields (num, type)
Output_Expr: match, match_last, match_all
  -- match_all.field not allowed
```

Linking_Name: "DMADa

Data_Structure_Text:

A_Task		A_IDX	A_Task_Source	Burst_Word
OSD		0	DRAM	64
Histogram	Load	0	DRAM	64
Resizer	Load	0	DRAM	64
Previewer	Load	0	DRAM	64
H3A	Load	0	DRAM	64

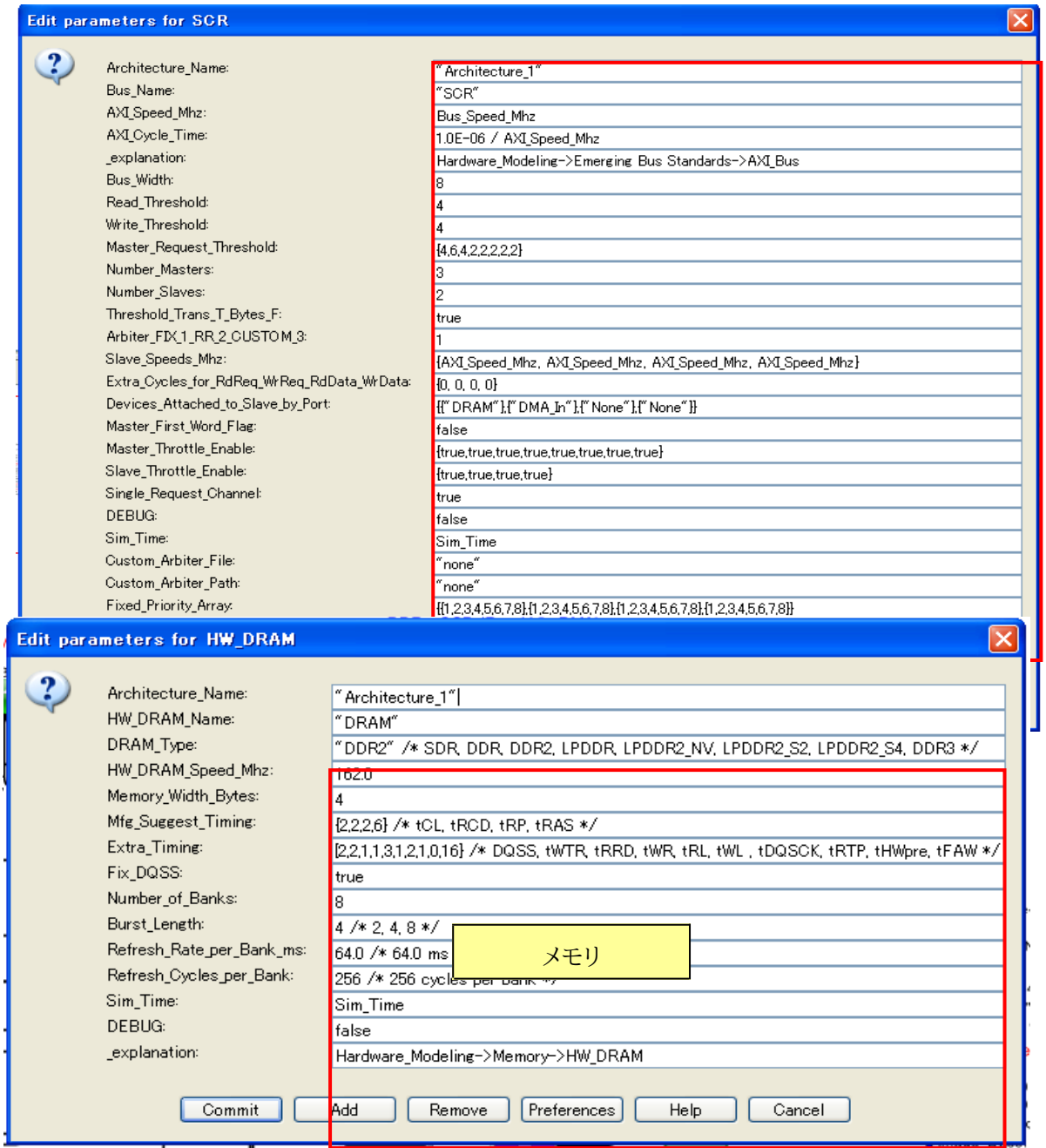
Input_Fields: "ID"
Lookup_Fields: "ID"
Output_Expression: "output = match" /* FORMAT output = match.fieldb */
Mode: Read

Commit | Add | Remove | Preferences | Help | Cancel

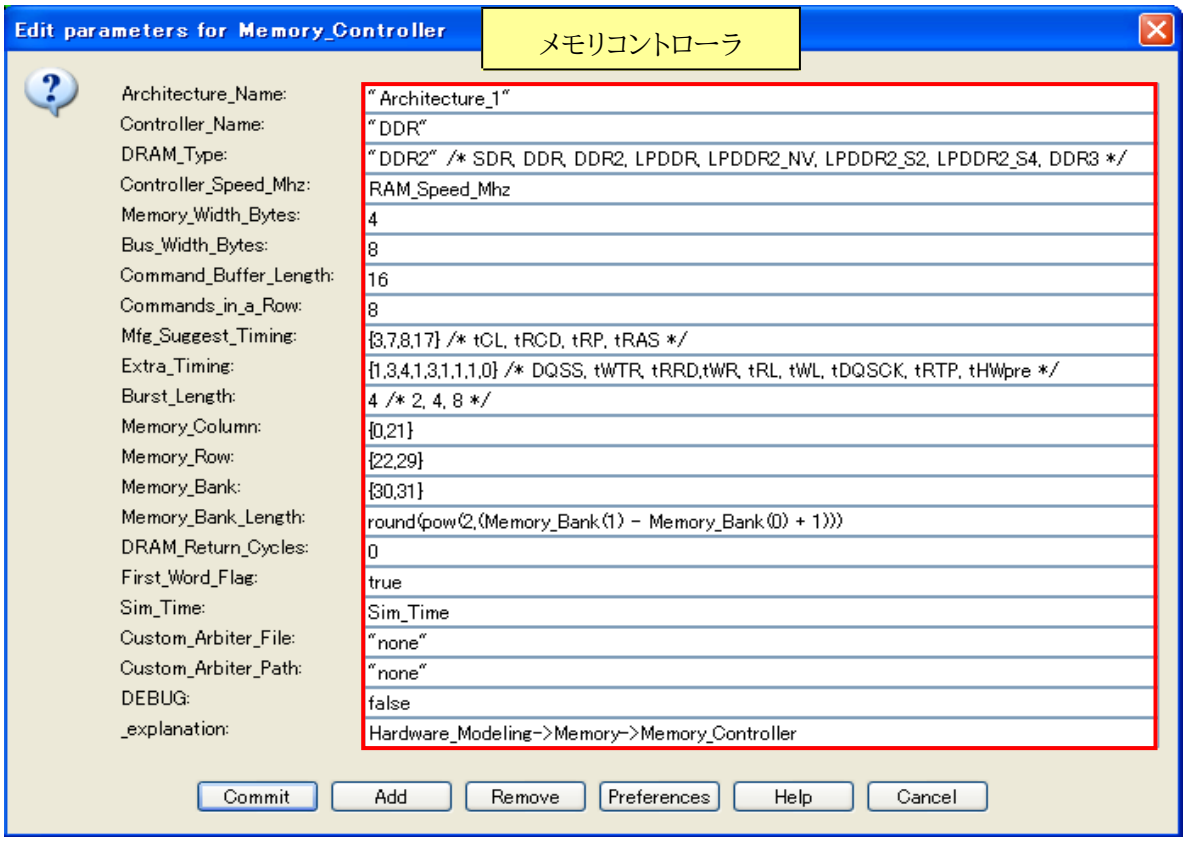
A_Task_Name	A_Instruction	A_IDX	A_Task_Source	Burst_Word_Size	A_Task_Address	A_Command	A_Bytes	A_Priority	A_Destination
OSD	Load	0	DRAM	64	1	Read	(Image_Bytes)	1	DMA ;↓
Histogram	Load	0	DRAM	64	2	Read	(Image_Bytes)	1	DMA ;↓
Resizer	Load	0	DRAM	64	3	Read	(Image_Bytes)	1	DMA ;↓
Previewer	Load	0	DRAM	64	4	Read	(Image_Bytes)	1	DMA ;↓
H3A	Load	0	DRAM	64	4	Read	(Image_Bytes)	1	DMA ;↓

3.5.2 Bus

Busは、“AXI”ブロックでチャネル方式のバスを使用。
Busの設定は、下記で行う。



3.5.



メモリは、“HW_DRAM”ブロックのメモリと“Memory_Controller”ブロックのスペックはバス幅接続、周波数以外は、仮のスペックでの設定。

4. パラメータ

入力データ量及び各モジュールの動作周波数を変数として設定。仕様により、TOP より変更可能。
各モジュールの設定も同様、変数で変更可能。
尚、各モジュール変数は、各モジュール内で定義。数値の変更は、TOP より変更可能。

- Sim_Time: 3e-3
- FPS: 200
- Architecture_Name: "Architecture_1"

Processor
- DSP_Speed: 405.0
- DSP_Name: "C64"
- Mem_Name: "DRAM"

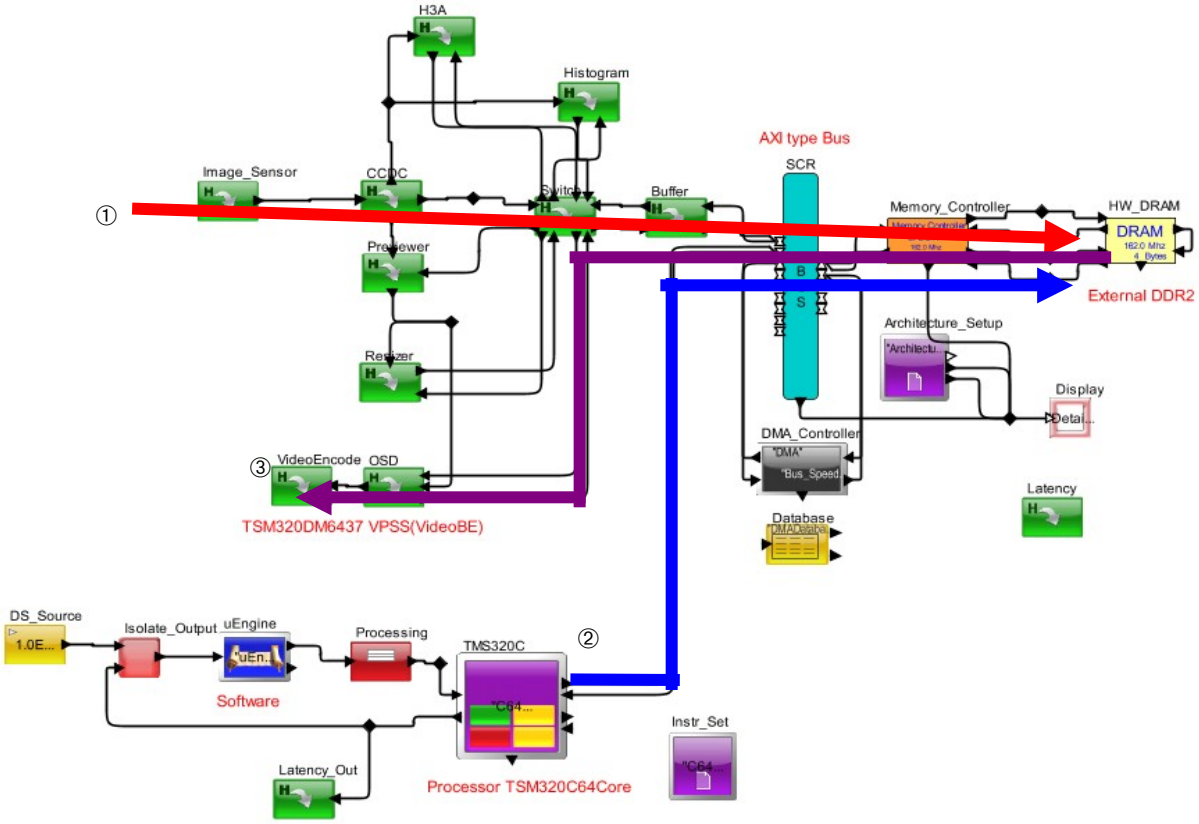
Image Characteristics
- data_bits: 8
- X_Pixels: 360
- Y_Pixels: 480
- Pixel_Bits: 2*data_bits
- Image_Bytes: ((X_Pixels*Y_Pixels*Pixel_Bits)/8) YUV:422 -> Y_pix_rasio:4,U,V_pix_rasio:2
- Raw_Enabled: false
- VPSS_Speed_Mhz 135

Switch Resource
- Bus_Speed_Mhz 405.0

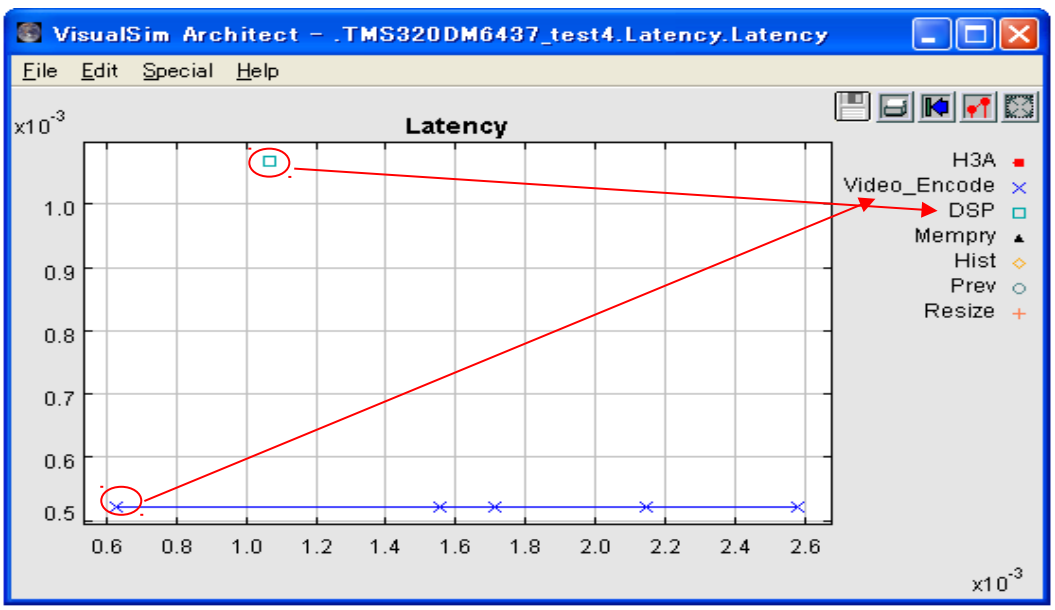
DDR2 Memory
- RAM_Speed_Mhz 162.0
- RAM_Access_Time: "Read 1000.0/RAM_Speed_Mhz*2,Prefetch 2.5,Refresh 1000.0/RAM_Speed_Mhz*2,Write 1000.0/RAM_Speed_Mhz*2"

5. シミュレーション

本モデルの現在の設定でシミュレーションをした結果を下記に示す。
シミュレーションのパスは下図の通り。各パスのアクセスは、シミュレーション開始よりそれぞれ
非同期に動作。



下記は、上記モデルのパスが非同期に動作した場合の vpbe（1 回実行）のシミュレーション結果。
グラフは、VideoEncoder と DSP のレイテンシの結果である。（sim 実行時間：0～3ms の結果）



下記は、シミュレーション結果のテキストデータ。“Architecture_Setup”が 3ms まで実行した時、設定の周波数で動作させた実時間での表示。（Min、Mean、Max）

```
DISPLAY AT TIME                  ------ 3.0000000000 ms ------↓
{BLOCK                           = ".TMS320DM6437_test4.Architecture_Setup", ↓
C64_Context_Switch_Time_Pct_Max = 0.0025514403292, ↓
C64_Context_Switch_Time_Pct_Mean         = 0.0025514403292, ↓
C64_Context_Switch_Time_Pct_Min = 0.0025514403292, ↓
C64_Context_Switch_Time_Pct_StDev        = 0.0, ↓
C64_D_1_Hit_Ratio_Max            = 100.0, ↓
C64_D_1_Hit_Ratio_Mean           = 1.2362637362637, ↓
C64_D_1_Hit_Ratio_Min            = 0.0, ↓
C64_D_1_Hit_Ratio_StDev          = 11.0497975366417, ↓
C64_D_1_KB_per_Thread_Max        = 0.0, ↓
C64_D_1_KB_per_Thread_Mean       = 0.0, ↓
C64_D_1_KB_per_Thread_Min        = 0.0, ↓
C64_D_1_KB_per_Thread_StDev      = 0.0, ↓
C64_I_1_Hit_Ratio_Max            = 100.0, ↓
C64_I_1_Hit_Ratio_Mean           = 21.6483516483516, ↓
C64_I_1_Hit_Ratio_Min            = 0.0, ↓
C64_I_1_Hit_Ratio_StDev          = 41.144712083666, ↓
C64_I_1_KB_per_Thread_Max        = 0.0, ↓
C64_I_1_KB_per_Thread_Mean       = 0.0, ↓
C64_I_1_KB_per_Thread_Min        = 0.0, ↓
C64_I_1_KB_per_Thread_StDev      = 0.0, ↓
C64_L2_Hit_Ratio_Max             = 100.0, ↓
C64_L2_Hit_Ratio_Mean            = 0.3571428571429, ↓
C64_L2_Hit_Ratio_Min             = 0.0, ↓
C64_L2_Hit_Ratio_StDev           = 5.6824318207238, ↓
C64_L2_KB_per_Thread_Max         = 0.0, ↓
C64_L2_KB_per_Thread_Mean        = 0.0, ↓
C64_L2_KB_per_Thread_Min         = 0.0, ↓
C64_L2_KB_per_Thread_StDev       = 0.0, ↓
C64_Stall_Time_Pct_Max           = 35.5739917695473, ↓
C64_Stall_Time_Pct_Mean          = 35.5739917695473, ↓
C64_Stall_Time_Pct_Min           = 35.5739917695473, ↓
C64_Stall_Time_Pct_StDev         = 0.0, ↓
C64_Task_Delay_Max               = 0.001069042435, ↓
C64_Task_Delay_Mean              = 8.9308096295879E-4, ↓
C64_Task_Delay_Min               = 1.340914E-5, ↓
C64_Task_Delay_StDev             = 3.1367743537874E-4, ↓
DELTA                            = 0.0, ↓
DMA_IO_per_sec_Max               = 3.558E6, ↓
DMA_IO_per_sec_Mean              = 3.558E6, ↓
DMA_IO_per_sec_Min               = 3.558E6, ↓
DMA_IO_per_sec_StDev             = 0.0, ↓
DMA_Throughput_MBs_Max           = 28.464, ↓
DMA_Throughput_MBs_Mean          = 28.464, ↓
DMA_Throughput_MBs_Min           = 28.464, ↓
DMA_Throughput_MBs_StDev         = 0.0, ↓
DS_NAME                          = "Architecture_Stats", ↓
ID                               = 1, ↓
INDEX                            = 0, ↓
TIME                             = 0.003}↓
↓
```

```
DISPLAY AT TIME                    ------ 3.0000000000 ms ------↓
{BLOCK                             = ".TMS320DM6437_test4.Architecture_Setup", ↓
C64_D_1_Utilization_Pct_Max        = 0.0368724279835, ↓
C64_D_1_Utilization_Pct_Mean       = 0.0368724279835, ↓
C64_D_1_Utilization_Pct_Min        = 0.0368724279835, ↓
C64_D_1_Utilization_Pct_StDev      = 0.0, ↓
C64_INT_1_Utilization_Pct_Max      = 0.0261728395062, ↓
C64_INT_1_Utilization_Pct_Mean     = 0.0261728395062, ↓
C64_INT_1_Utilization_Pct_Min      = 0.0261728395062, ↓
C64_INT_1_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_2_Utilization_Pct_Max      = 0.021316872428, ↓
C64_INT_2_Utilization_Pct_Mean     = 0.021316872428, ↓
C64_INT_2_Utilization_Pct_Min      = 0.021316872428, ↓
C64_INT_2_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_3_Utilization_Pct_Max      = 0.0012345679012, ↓
C64_INT_3_Utilization_Pct_Mean     = 0.0012345679012, ↓
C64_INT_3_Utilization_Pct_Min      = 0.0012345679012, ↓
C64_INT_3_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_4_Utilization_Pct_Max      = 0.0, ↓
C64_INT_4_Utilization_Pct_Mean     = 0.0, ↓
C64_INT_4_Utilization_Pct_Min      = 0.0, ↓
C64_INT_4_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_5_Utilization_Pct_Max      = 0.022633744856, ↓
C64_INT_5_Utilization_Pct_Mean     = 0.022633744856, ↓
C64_INT_5_Utilization_Pct_Min      = 0.022633744856, ↓
C64_INT_5_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_6_Utilization_Pct_Max      = 0.0184362139918, ↓
C64_INT_6_Utilization_Pct_Mean     = 0.0184362139918, ↓
C64_INT_6_Utilization_Pct_Min      = 0.0184362139918, ↓
C64_INT_6_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_7_Utilization_Pct_Max      = 0.0316872427984, ↓
C64_INT_7_Utilization_Pct_Mean     = 0.0316872427984, ↓
C64_INT_7_Utilization_Pct_Min      = 0.0316872427984, ↓
C64_INT_7_Utilization_Pct_StDev    = 0.0, ↓
C64_INT_8_Utilization_Pct_Max      = 0.0275720164609, ↓
C64_INT_8_Utilization_Pct_Mean     = 0.0275720164609, ↓
C64_INT_8_Utilization_Pct_Min      = 0.0275720164609, ↓
C64_INT_8_Utilization_Pct_StDev    = 0.0, ↓
C64_I_1_Utilization_Pct_Max        = 0.019012345679, ↓
C64_I_1_Utilization_Pct_Mean       = 0.019012345679, ↓
C64_I_1_Utilization_Pct_Min        = 0.019012345679, ↓
C64_I_1_Utilization_Pct_StDev      = 0.0, ↓
C64_L2_Utilization_Pct_Max         = 0.0059259259259, ↓
C64_L2_Utilization_Pct_Mean        = 0.0059259259259, ↓
C64_L2_Utilization_Pct_Min         = 0.0059259259259, ↓
C64_L2_Utilization_Pct_StDev       = 0.0, ↓
C64_PROC_Utilization_Pct_Max       = 0.016277302944, ↓
C64_PROC_Utilization_Pct_Mean      = 0.016277302944, ↓
C64_PROC_Utilization_Pct_Min       = 0.016277302944, ↓
C64_PROC_Utilization_Pct_StDev     = 0.0, ↓
C64_Pipeline_Utilization_Pct_Max           = 0.019012345679, ↓
C64_Pipeline_Utilization_Pct_Mean          = 0.019012345679, ↓
C64_Pipeline_Utilization_Pct_Min           = 0.019012345679, ↓
C64_Pipeline_Utilization_Pct_StDev         = 0.0, ↓
C64_Register_Rd_Utilization_Pct_Max        = 8.2304526748971E-5, ↓
C64_Register_Rd_Utilization_Pct_Mean       = 8.2304526748971E-5, ↓
C64_Register_Rd_Utilization_Pct_Min        = 8.2304526748971E-5, ↓
C64_Register_Rd_Utilization_Pct_StDev      = 0.0, ↓
C64_Register_Wr_Utilization_Pct_Max        = 6.5843621399177E-4, ↓
C64_Register_Wr_Utilization_Pct_Mean       = 6.5843621399177E-4, ↓
C64_Register_Wr_Utilization_Pct_Min        = 6.5843621399177E-4, ↓
C64_Register_Wr_Utilization_Pct_StDev      = 0.0, ↓
DELTA                              = 0.0, ↓
DS_NAME                            = "Architecture_Stats", ↓
ID                                 = 1, ↓
INDEX                              = 0, ↓
TIME                               = 0.003}↓
↓
↓
```

6. メリット

1.VisualSimは、前述のモデルのようにクロックサイクルベースで正確な時間をシミュレーション
することが可能です。
2.さらに、他のESLツールのように詳細な設計が決まらない段階より、容易にモデル化を行いシス
テム性能の検討が可能です。
3.複数のデバイス、モジュール、IF、バスなどが複雑に絡み動作（非同期/同期共）する大規模な大きなシス
テムのHW/SWに要求される性能を決定する場合の検討に長けています。
よって、設計初期のシステムの構成や個々のバス、デバイス、モジュールのHW/SWが満たす必要
があるスペックやHW/SWの切り分けなど、性能とコストの最適なトレードオフを定量的かつ精度
良く決めることが可能です。
4. 新規システム設計や大幅な仕様の変更なども容易にモデル変更や作成が行えるため、短期間で
製品開発を行うことを可能にします。
5.多くのライブラリやモデル作成の例などを使い、短期間で容易に検討を始めることが可能です。
また、C/C++、JAVA、SystemC、MATLABやVisualSim内で使用するRegEXなどの多くの言語をサポー
トしていますので、システム検討の進捗に合わせ、複雑なファンクションを持った記述のモデル
に切り替えれば、詳細な検討及び結果の精度を高めることが可能です。
6.柔軟なモデル作成が可能ですので、電気的なシステムだけでなくメカニカルなモデルやセンサー
などもモデル化するれば、それらの電子制御のシステム的な性能や構成の検討が可能です。
7.多くのケースで容易に性能の検証が行えますので、インプリ時に把握困難な性能ネックの動作
モードやクリティカルな経路をHW/SWを含んだシステムとして事前に把握可能なため、RTLやSW
の設計/検証時に対策/確認が容易になり、設計品質の向上/検証作業の効率化が可能です。

尚、本ツールは、上記の多くのメリットを出すため、ライブラリの抽象度を高くしています。
また、システム設計の初期段階で使用するため、詳細設計後のRTLやSWの記述の構造的な影響や
抽象度の高さによる誤差が最終製品との差異になります。
さらに、合成などにより後の設計工程へ自動的にモデルを継承することが出来ませんので、これら
の特性をご理解頂いた上でご使用頂ければ、設計/検証の効率化、品質の向上、システムのコスト
ダウンに大きな力を発揮することと思います。

今後、システム検証に必要なモデル化やツールのサポートをさせて頂きますので、ご質問や不明点
などございましたら、ご遠慮なくお問い合わせ頂けますようお願い致します。