Conditional Expressions in IBIS-AMI
IBIS Summit, DesignCon 2010, Santa Clara, California
The Need for Conditional Expressions

- AMI Configuration data supplies one set of data or user choices
- User can select from a list, range or increment
  - (fileparam
    (Usage In)
    (Type String)
    (Format List “File1.dat” “File2.dat” “File3.dat”)
  )
- Different configuration setups may be required for corners, data rates, etc
  - Many parameters to tweak
  - User must code by hand, or
  - EDA vendors provide proprietary wrapper
- SerDes vendor must supply these parameters in additional data
- Conditional Expressions gives AMI Configuration a “pre-process” facility
- IBM’s HSSCDR simulator uses conditional expressions for rate- and parameter-dependent values
Current facilities in AMI Configuration

- Why Can’t we use Corner Format?
  - Three corners may not be enough
    - Extreme slow, extreme fast, ideal
  - Other combinations may be required
    - E.g. Best process, worst voltage,
    - Alternative s-parameters for different supply voltages
    - Gain factors based on user registers

- Why can’t the DLL calculate internally? Why does the EDA tool have to know?
  - No good for "simulator directives" (Usage Info) e.g.
    - SJ, RJ, Tx_DCD …
    - On-chip s-parms

- Why not get the EDA tool to do it?
  - May not be required for all DLLs
  - May not cover DLL user’s needs
  - Easily accommodated within API.
Types of preprocessing that might be needed

- **Substitution**
  - Use of parameter (short) string as part of filename
    - (Tx_IC "ic_tx_${CORNER}.s4p"
      where ${CORNER} = ("nc" | "bc" | "wc" | "ec" | "0")

- **"Case" or "Switch" statements**
  - Selection of one value based on an index value
    - (Tx_DCD "(${CORNER}=='EC' ? 1.05 :
        ($CORNER=='WC' ? 0.93 :
        ($CORNER=='BC' ? 0.20 :
        0.5 )))" )

- **Unit conversion**
  - Changing a parameter that expects "% UI" into one that expects absolute time
    - (rj "0.321*$BAUD/10e9" ) converts 321fs into %UI.
Types of preprocessing that might be needed (contd.)

- **Threshold**
  - Selection of parameter based on threshold values of another (number) parameter
    - $(\text{fileparm } "($BAUD<8.75e9 \ ? \ 'low\_rate\_file' \ : \\
                      ($BAUD<11.4e9 \ ? \ 'mid\_rate\_file' \ : \\
                      'high\_rate\_file' ))\)"

- **Piecewise Linear Approximation**
  - Calculation of value based on linear interpolation between measured values
    - $(\text{txlev } "($TXPOW<0 \ ? \ 1 \ : \\
                      ($TXPOW<21 \ ? \ ($TXPOW-0)/(21-0)*(230-0)+0 \\
                      ($TXPOW<33 \ ? \ ($TXPOW-21)/(33-21)*(358-230)+230 \\
                      ($TXPOW<47 \ ? \ ($TXPOW-33)/(47-33)*(506-358)+358 \\
                      ($TXPOW<60 \ ? \ ($TXPOW-47)/(60-47)*(640-506)+506 \\
                      640 ))))"

- **Any combinations of the above**
  - $\text{BAUD}$ may be derived by DLL from "bit_time" in AMI_Init call
  - Whitespace can be removed to avoid newlines in strings.
How will it work?

- Parameter value(s) contain evaluation **string**, written in language of choice
  - Optional prefix can denote language type (Rj "EVAL:0.321*$BAUD/10e9")
  - May use curly brackets to signify CE: (Rj "{0.321*$BAUD/10e9}")

- Expressions are dependent on other parameters, which will be automatically entered as variables by the preprocessor (with $ prefix)

- Some parameters may be processed ahead of all others (e.g. initialization)
  - (Init "EVAL:$pi=3.14159")

- EDA tool calls **AMI_Init** with special “preprocess” flag to tell DLL to resolve parameters
  - **AMI_memory_handle** NULL,
  - No impulse response (*impulse_matrix* NULL), or
  - Negative number of aggressors

- DLL tool resolves parameters and returns them in **AMI_parameters_out**

- EDA tool now uses resolved parameters to start simulation

- Space reserved for **AMI_parameters_out** must be freed in **AMI_Close**.
What Language to use?

- DLL must implement resolution of conditional expressions
- Language used is the choice of DLL developer
- EDA tool has no interest in content of evaluation strings
  - DLL and Configuration file must agree
- Can be proprietary, public domain or open-source
  - Open-source should be dynamically linked to protect IP
- Can be home grown
  - Preferably should support strings
  - Costly to develop and maintain.
Possible Interpretative Languages

- C Inline Evaluator (many sources)
  - (+) Full functions
  - (-) May not process strings

- Forth
  - (+) Integer, Float and String support
  - (+) Extensive Scientific Function library
  - (+) Many implementations available in public domain
    - MinForth
    - Pforth
    - FICL
  - (-) RPN notation

- Perl
  - (+) supports anything
  - (+) easily testable on command line
  - (-) bulky

- Others?
Example AMI file

```
( Init
  ( Usage In )
  ( Type String )
  ( Description "Interpreter Initialization")
  ( Value "FORTH: : THRESH DUP SA+ DUP C@ 0= IF DROP FDROP EXIT THEN >FLOAT0 FOVER F< WHILE 2 +SA
    REPEAT FDROP ;"
  )
)

( corner
  ( Usage Info )
  ( List "0" "nc" "bc" "wc" "ec" )
  ( Labels "Ideal" "Nominal" "Best" "Worst" "Extreme" )
  ( Type String )
  ( Description "Corner selected by the user" )
)

( baud
  ( Usage In )
  ( Type Float )
  ( Description "Baud rate" )
  ( List 15.1e9 6.4e9 10.1e9 )
)

( rotlins
  ( Array ( Usage Info ) ( Type Boolean ) ( Value True ) )
  ( Usage Info )
  ( Description "Rotator Linearity file")
  ( 0
    ( Type String )
    ( Value "pr_wc_3g75.dat" )
  )
  ( 1
    ( Type String )
    ( Value "7.5e9" )
  )
  ( 2
    ( Type String )
    ( Value "pr_wc_6g4.dat" )
  )
)

( rotlin
  ( Usage In )
  ( Type String )
  ( Description "Rotator Linearity File" )
  ( Value "FORTH: $rotlins f$baud THRESH COUNT TYPE" )
)```
Example AMI file (results)

- **Sent to the DLL:**
  - ( Init "FORTH: : THRESH DUP SA+ DUP C@ 0= IF DROP FDROP EXIT THEN >FLOAT0 FOVER F<
    WHILE 2 +SA REPEAT FDROP ;"
  )

  ( corner "nc"
  ( baud 15.1e9 ) | $baud COUNT >FLT FCONSTANT f$baud
  ( rotlins "pr_wc_3g75.dat" "7.5e9" "pr_wc_6g4.dat"
  )
  ( rotlin "FORTH: $rotlins f$baud THRESH COUNT TYPE"
  )

- **Returned from DLL:**
  - ( corner "nc"
  ( baud 15.1e9
  ( rotlins "pr_wc_3g75.dat" "7.5e9" "pr_wc_6g4.dat"
  )
  ( rotlin "pr_wc_6g4.dat"
  )
Summary

- Conditional Preprocessing necessary for some models
- Best handled by the DLL, not EDA tool
- Several useful functions identified
- Simple hooks into IBIS-AMI API
- Choice of interpretive languages
- Remove the need for EDA Vendor wrappers for models.