AMI_Resolve: A case study for 56G PAM4

DesignCon IBIS Summit
February 3, 2017
Santa Clara, California

Adge Hawes, IBM
Steve Parker, GLOBALFOUNDRIES
AMI Configuration

- A textual hierarchical, data file similar to JSON
  - Simple and effective
- Users are provided choices using List or Range:
  - Choices made before simulation
  - AMI_Init processes parameters
- Corner provides limited selection:
  - Allows different settings for each operating corner
- EDA tool sends simple parameter-value pairs

- (ic_rx
  (Type String)
  (Usage In)
  (Value “ic_rx_nc.s4p”))

- (rx_rj
  (Type Float)
  (Usage In)
  (List 30e-12 45e-12 15e-12 60e-12))

- (rx_rj
  (Type Float)
  (Usage In)
  (Corner 30e-12 45e-12 15e-12))

- (ic_rx “ic_rx_nc.s4p”)
- (rx_rj 30e-12)
Limitations of Corner

- **What do the corners represent?**
  - Typ-min-max
    - Min what? Speed or delay?
  - Typ-slow-fast

- **No other corners supported**
  - Extreme best or extreme worst
  - Highest temperature but best process

- **Other variations not covered**
  - Rate, LTE mode, jitter model, etc.

<table>
<thead>
<tr>
<th></th>
<th>NC</th>
<th>WC</th>
<th>BC</th>
<th>EC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Halfrate agcmax</td>
<td>3.2</td>
<td>2.9</td>
<td>3.9</td>
<td>2.4</td>
</tr>
<tr>
<td>agcmin</td>
<td>0.4</td>
<td>0.5</td>
<td>0.1</td>
<td>0.6</td>
</tr>
<tr>
<td>Fullrate agcmax</td>
<td>3.1</td>
<td>2.8</td>
<td>3.8</td>
<td>2.3</td>
</tr>
<tr>
<td>agcmin</td>
<td>0.5</td>
<td>0.6</td>
<td>0.2</td>
<td>0.7</td>
</tr>
</tbody>
</table>
AMI_Resolve

- Parameters given new Usage type Dep:
  - Value is a placeholder
- Additional AMI call: AMI_Resolve:
  - Before AMI_Init
- Call passes all pre-selected parameters of Usage In e.g. “my_corner”
  - Also passes metrics, e.g. bit_time
- Model uses AMI_Resolve to insert values of Usage Dep parameters
- AMI_Resolve_Close cleans up
- Results presented back to EDA tool
  - proceeds with simulation using resolved parameters

- (rx_rj
  (Type Float)
  (Usage Dep)
  (Value 30e-12))

- int AMI_Resolve(
  char *AMI_Parameters_In,
  char **AMI_Parameters_Out);

- (my_corner
  (Usage In)
  (Type String)
  (List "Nom" "Worst" "Best" "Extreme"))

- int AMI_Resolve_Close(
  char *AMI_Parameters_Out);
- **Code implemented in executable DLL:**
  - Specific code in DLL resolves parameters
  - Compiled along with model
- **Advantages:**
  - Simple, minimal AMI configuration
  - Private
- **Disadvantages:**
  - Inflexible
  - Requires frequent recompilation

- **Code implemented in AMI configuration:**
  - Interpreter built into DLL code
  - AMI configuration supplies resolve instructions
- **Advantages:**
  - Flexible
  - Change in dependency does not require recompilation of the model
- **Disadvantages:**
  - More complex implementation
  - In plain view, unless scrambled
GLOBALFOUNDRIES Interpreted solution

- Scripting provided by Forth-like RPN interpreter
- (char *) and (float) stacks
- Commands for data and stack manipulation
- Scripts are short, in Value field, and can call other scripts
  - Parameters passed on stack
- RPN allows short quick calculations:
  - 3 5 9 + *
- Other AMI parameters can be written or read
- Integer, string or float data types

```
0< 0= 0> NEG NOT AND OR XOR + - * / MOD /MOD 1I 2I 1+ 1- 2* 2/ < > ?: DROP ?
DUP DUP NIP TUCK OVER SWAP ROT -ROT PICK ROLL NDROP = CMP EQ NE CAT UC LC >I <I
I. . FNEG F+ F* F/ FSWAP FDROP 1/F FDUP FOVER FPICK F= F0< F0= F1+ F0 F!
F?: INT >F F. <F FREE EXE @EXE ! @ @@ 1! 1@ N! N@ @@
bit_time corner model_name
```
AMI_Resolve examples – setup (sub-value support)

- Selection values are stored in an array
  - Labels are used to address them
  - Valid AMI data
  - Used for supplying values to other parameters
  - Usage In forces inclusion in AMI.Resolve call

- Dependent parameters have placeholder values
  - Value used if AMI.Resolve not supported
  - Not included in AMI.Resolve call

- (Tstonefile.Values
  (Usage In)
  (Type String)
  (Value
    (Extreme  "ic_rx_ec.s4p")
    (Best    "ic_rx_bc.s4p")
    (Worst   "ic_rx_wc.s4p")
    (Nom     "ic_rx_nc.s4p")))

- (baud
  (Usage Dep)
  (Type Float)
  (Value 28e09))

- (Tstonefile
  (Usage Dep)
  (Type String)
  (Value "ic_rx_nc.s4p"))
AMIResolve examples – setup (no sub-value)

- Selection values are stored in an array
  - Labels are used to address them
  - Valid AMI data
  - Used for supplying values to other parameters
  - Usage In forces inclusion in AMIResolve call

- Dependent parameters have placeholder values
  - Value used if AMIResolve not supported
  - Not included in AMIResolve call

- (Tstonefile_Values.Extreme
  (Usage In) (Type String)
  (Value "ic_rx_ec.s4p"))
- (Tstonefile_Values.Best
  (Usage In) (Type String)
  (Value "ic_rx_bc.s4p"))
- (Tstonefile_Values.Nom
  (Usage In) (Type String)
  (Value "ic_rx_nc.s4p"))
- (Tstonefile_Values.Worst
  (Usage In) (Type String)
  (Value "ic_rx_wc.s4p"))

- (baud
  (Usage Dep)
  (Type Float)
  (Value 28e09))
- (Tstonefile
  (Usage Dep)
  (Type String)
  (Value "ic_rx_nc.s4p"))
The parameter **AMI.Resolve** is executed

- Usually calls other parameter scripts

**Simple parameter calculation**

- **Words** are placed on stack, **operators** are executed, consuming/filling stack
- e.g. deriving Baud Rate from **bit_time**
- Extra math functions can be loaded

**Parameters can be multiple-use subroutines**

- Stack-based parameter passing means routines can be reused

**AMI.Resolve examples - execution**

- **(AMI.Resolve**
  (Usage In)
  (Type String)
  (Value "baud_Dep @EXE Tstonefile Select_Corner @EXE")
)

- **(baud_Dep**
  (Usage In)
  (Type String)
  (Description "Baud rate calculator")
  (Value "bit_time 1/F baud F!")
)

- **(Select_Corner**
  (Usage In)
  (Type String)
  (Value "DUP _Values. CAT TUCK my_corner @ CAT TUCK @ SWAP ! FREE FREE")
  (Description "Select Corner of parameter on stack")
)
Encrypting AMI_Resolve scripts

- Convert script from ASCII to binary
  - using ASCII85/91/94 decode
- XOR-encrypt binary data using embedded key
- Convert encrypted data to ASCII string
  - using ASCII85/91/94 encode plus “#”
- Process reversed by AMI_Resolve when ‘#’ is detected
- Only Value (not decrypted text) is presented to AMI_Init

- (Secret_routine
  (Usage In)
  (Type String)
  (Value "DUP _Values. CAT TUCK my_corner @ CAT TUCK @ SWAP ! FREE FREE")
  (Description "A proprietary routine")

- (Secret_routine
  (Usage In)
  (Type String)
  (Value "#O<`^zX>%ZCX>)XGZfA9Ab7*B`EFf-gbRchTY<VDJc_3(Mb0B")
  (Description "A proprietary routine")

- (Result_of_Secret_Routine
  (Usage Dep)
  (Type Float)
  (Value 30.45e9)
  (Description "Special value")
Conclusion

- AMI_Resolve provides a universal solution to AMI parameter selection demands
- Cross-vendor support
- GLOBALFOUNDRIES FORTH-like scripting places selection into AMI configuration file
- Scripting is fast and efficient
- Frequent model compilation is avoided
- Encryption provides protection for sensitive dependencies
Thank you!

---

QUESTIONS?
Sed ut perspiciatis unde omnis iste natus

- Sed ut perspiciatis unde omnis iste natus
- Sed ut perspiciatis unde omnis iste natus
- Sed ut perspiciatis unde omnis iste natus
- Sed ut perspiciatis unde omnis iste natus
- Sed ut perspiciatis unde omnis iste natus

Sed ut perspiciatis unde omnis iste natus

- Sed ut perspiciatis unde omnis iste natus
  - Sed ut perspiciatis unde omnis iste natus
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin condimentum tristique dui quis pretium. Etiam dignissim nibh nec velit suscipit, in malesuada nisi egestas. Nullam venenatis, arcu a
SPEAKERS

Speaker 1 Name
Title, Company
Email@address.com | website.com | @twitter

Speaker 2 Name
Title, Company
Email@address.com | website.com | @twitter