

Register Management of a Complex Multi-Processor Based SoC

Dave Murray, Brian Clinton, Zoltan Sugar - Duolog





Example : OMAP3430

- Industry's first processor with advanced Superscalar ARM® Cortex[™]-A8 Core
- Industry's first processor designed in 65nm CMOS process technology adds processing performance
- IVA[™] 2+ (Image Video Audio) accelerator enables multi-standard (MPEG4, WMV9, RealVideo, H263, H264) encode/decode at D1 (720x480 pixels) 30 fps
- Integrated image signal processor (ISP) for faster, higher-quality image capture and lower system cost



Source : http://focus.ti.com/pdfs/wtbu/ti_omap3family.pdf





Multi-Processor Based SoC



SoC Integration – Evolving Pain



HW/SW inteface



Sometimes due to it's late arrival, it will completely mess up your schedule





Register and Bitfields







SoC Complexity



08

duolog 4

Traditional 'Baggage Handling'



Compressed Design Cycle



duolog

Features

- TTM has compressed the integration cycle
- Earlier deliverables required
- Parallel/Concurrent process, not Waterfall

Problems

- No refinement or milestones
- Very difficult to keep specification synchronised
- High-Level of interdependency means high cost of incremental changes and schedule slippages
- Massive translation leading to designed-in errors and high resource costs
- No central coherency meaning bugs are caught farther downstream
- Doesn't scale to increasing complexity
- This applies to each suitcase (Bitfield)
 - 45 flights of 220 people with 3 suitcases taking off at the same time = how many lost and missing ??



High-level Requirements - How do we solve?



duoloc

- Single Source solution
- Efficient modelling of bitfields, registers, systems across the different design disciplines to allow collaboration
- **Auto-generation** of multiple output views
- Standardization of data across the company
- Increased Quality through Correct-By-Construction methodology
- Promote reusability and modularisation
- Ensure fast generation and ability to regress easily
- Ensure system scalability e.g. for 10,000s of registers
- Design to be interoperable via IP-XACT
- Aim to be non-disruptive to current flows
- Easy to Use environment across all major OS.





Register and memory map management

High-level Solution





HW/SW Interface Management Solution

- Duolog have developed 'BITWISE' to manage HW/SW interface:
 - Efficient Model
 - User Friendly Design capture
 - IP and System Design flows
 - Standardized interfaces
 - Correct-by-Construction methodology
 - Flexible Generator Mechanism





'Bitwise' Data Flow







IP & System Design Flows



Modelling the System



User Friendly GUI

• 🔍 🔩 🚅 📽 🔛 🗁 🗎 🕥 • 🕞 •	Q₂ • ½ + ♀ + ♀ + ≝			🗈 📰 Bitwise
ocrates Explorer 🛛 🗖	plcp_11a bb_11a_BB_APB.ral *cor	e_11a 🛛 🖓 🗖	System Memory Map Viewer	
수 수 @ 🖻 🕏	Register Editor			
Baseband80211a	Registers Details Registers Table			
bb_11a	Registe	r (manual and		
File picp_11a	Registers Editor	Bitheld Details		
Architecture	DefaultMode			
Core 11a		Bitrieid: tU2_thresh		int_status_0x00000100
🖮 🗧 Interface	Default Register Width (number of bits): 32	Delay value in 20MHz samples		R proprieta a construction of the construction
😑 🏠 Architecture	🗷 🔜 scr ctrl	ew Description:		tx hold 0x0000010C
🖮 🗄 Instances [Unavailable]	🗷 🔜 tx_kmod_bpsk		0×00000100	netStatus 0x00000110
- frontend_11a	😥 🔜 tx_kmod_qpsk	ete		netro1 0x00000114
Example And	🗷 🖬 tx_kmod_16qam			
MAC80211a	🗷 💼 tx_kmod_64qam	P Offset: 0 DefaultMode	•	•
mac_trans_11a	🕒 📷 agc_gfit1 🛛 🗖 Do	wn	BB_APB	PB
Interface	agc_gfit2	urbh.	😨 i_bb_11a 🛛 🕞 i_core_11a	
	agc_gfit3	p-Fill width: 7	0×00000100 0×00000140	
TOP 802 11	🔟 🔟 dagc_pwr_tgt	Reset Value: 0x58 DefaultMode		
- TOP_002_11	Im mF_ctrl	er		_
	International In			+
Architecture	1119 Deserved	Access: READ_WRITE DefaultMode	Entry Point' TOP SBI	
		R Type: R 🔽 Data	Start Address: 0x00000000	
Library/Design				
Explorer	The man is the second s	W Type: W 💙 Data		
Explorei	The range data		+	
	The short thr	Reserved:	MAC_APB	
	🕀 🔜 rxp delay thr		🗵 i_mac_tra	
	🖃 🖬 rxp_long_thr		0×0000000	Memory
vigator 🛛 🔪 🗇 🗢 🗟 📄 🛱 🎽	💷 💼 rxp_efit1	Enumeration Details Table		i icilioi y
😟 🗁 EStruct	🔺 🗊 rxp_efit2	Create/Delete/Edit the Enumeration's attributes		Мар
😐 🗁 HTML	🖬 💼 rxp_efit3	Nama Description Value 🔨 Nam		Viewer
🗉 🧁 OVM_REG	🕒 👼 ph_thr1			The first fi
😑 🧁 RAL	🕒 👼 ph_thr2	Error_Co An error h 0x7h Delete		
Generator.log	🗷 📷 eq_alpha			
bb_11a_BB_APB.ral	rx_kmod_bpsk		🔜 Register Diagram 🛛	
i_core_11a_regdef.ral	Trx_kmod_qpsk		«Zoom»	
i_frontend_11a_regdef.ral	the most rx_kmod_16qam			
Tiplcp_11a_regdet.ral	TX_KMOD_64qam			
B Costernularian	Ourseiten Interferen Instanten Deventeten Devision	Desister Dualles Massau Mass	rxctrl_t02_thr [Reset : 00000000000000000000000000000000000	0000001011000]
C Vubi Last	Overview linternaces linstances Parameters Registers	Register buildies Melliory Maps Modes	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 1	
	📒 🗖 System Memory Map Log 🛛 🔡 Problems 📃 C	onsole 🔰 🕺 🍟 🗖 🗖		
	Total registers 75: 0 issues.		R0	RW
🕀 🗁 VR_AD	Dhysical Address Degister	ath Bre Becol Component Broblem	Reserved	t02_thresh
🖻 🗁 Verilog_Leaf	0x100 int status	on 802 11/i plcn 11a		
🖶 🗁 Verilog System	0x10 1 int_mask //	op_802_11/i plcp_11aplcp_11a		
🗄 🗁 Word_Doc 🛛 🕞 🗖		op_802_11/i plcp_11a plcp_11a Problem	Desister	
Templates		op_802_11/i plcp_11a plcp_11a IIODICIII	Register	
Per State Navigator		op 802 11/i plcp 11a plcp 11a LOG	Diagram	
Stats	Dw1 D cor chri h	on 002 11/1 core 115 core 115		
🖳 🗁 c_api_gen				Þ
	- 0			

IP & System Design Flows







Standardized Interface with IP-XACT™

- Use IP-XACT[™] as a standardized communication mechanism
- It is the best standard for register and memory map management
- Bitwise abstracts/decouples IP-XACT[™] for user-friendliness, completeness and simplicity
- Import/Export IP-XACT[™] 1.1, 1.2,1.4
 - Extract registers, bitfields, memory maps, etc
- Duolog has worked with TI and IP vendors to ensure alignment
- Main guideline keep it simple





Correct-By-Construction

- Comprehensive 'coherency checks' to identify incorrect data
- Coherency checks run on-demand and onthe-fly as the user is entering data
- Coherency checks can be run on data imported to the system
- Successful coherency checking must be performed before any output data is generated





Generator Framework









Case Study

Case study : Multimedia Processor





Case Study : SoC Application Processor



duologies



Soc Design Flow



Statistics

Platform

- 2.4GHz, 2Gbyte desktop machine with T3400 Intel dual core
- Windows XP professional
- Outputs

- 94 MBytes of Documentation (2 flavours)
- 7.5 Million lines of C Code (3 flavours)
- 1.26 Million lines of E code (2 flavours)
- + Millions of lines of code of custom generators developed within Bitwise
- Generation Times
 - Coherency checking time = < 10 minutes</p>
 - Generation time = average 5 minutes / generator



Benefits

Improved Quality

- Single-source specification in a single location Write once, generate all
- Perfect-by-construction methodology
- Comprehensive coherency checking

Improved Productivity

- Consolidate information across multiple users
- Promote early collaboration between engineering teams
- Eliminate manual processes
 - Release engineering talent to perform more creative tasks
- Single reference point and terminology for hardware & software teams

Reduced schedule

- Facilitate re-use methodology
- Immediate turn-around from spec to generated views
- Perfect-by-construction [™] → less debug, less re-work
- CLI to run regressions

Reduced Costs

30x+ resource savings over manual based flows







Register and memory map management

Conclusion





Conclusion

- There are key pressures requiring early and robust HW/SW integration of ever more complex systems.
- A single-source specification can be utilised to provide a correct-by-construction methodology
- Current key focus is on auto-generation of verification collateral that ensures robust SW integration
- Software view of the system could be delivered much earlier in the design cycle before HW has been fully integrated
- A good solution can ensure fewer bugs @ less cost in less time.





Requirements Coverage

	IP Libraries	Full Register Model	High-level GUI	IP-XACT Import/Export	Eclipse	Coherency Check Framework	Generator Framework	CLI
Single Source solution	0							
Efficient capture of information across the different design disciplines : Collaboration								
Auto-Generation of multiple output views to keep information synchronous and up-to- date						Ó		
Standardization of data across the company							O	
Increased Quality through rigorous specification DRC or Coherency checks	0							
Promote reusability and modularisation							O	
Ensure fast generation and ability to regress easily							\bigcirc	
Ensure system scalability e.g. for 10,000s of registers	0							
Design to be interoperable				Q				
Aim to be non-disruptive to current flows				\bigcirc				



Acknowledgements

- TI WTBU Team in Nice
 - Hedi Boufaied
 - Bertrand Blanc
 - Bob Maaraoui
 - Denis Kuntz





Thank you – Questions?





