Metadata based padring and pad multiplexing generation for microcontroller design

Augusto Ken Morita[1,2], Rubens Takiguti[1] and Wilhelmus A. M.Van Noije[2]

[1]Freescale Semiconductor, Campinas, Brazil
[2]LSI/PSI – Escola Politécnica, Universidade de São Paulo Sao Paulo, Brazil
e-mail: augusto.morita@freescale.com, rubens.takiguti@freescale.com and noije@lsi.usp.br

ABSTRACT

Defining padring and its pad interconnections is one of the most time consuming steps of a microcontroller SoC design. Also, multiple device features plus testability in limited pinout packages entails more functions shared per pin. As such, the number of padring connections and complexity exponentially grows at each new design. This paper describes a methodology to configure pad connections and select correct pad cells. The methodology works out technology and function definitions for each pin using new and previous designs data stored in a metadata database in order to generate the padring RTL code for the SoC. This methodology was proven with Freescale microcontrollers, among them to check the padring connection of the commercial microcontroller MC9S08MP16 [1], and was used to implement both MC9S08QB8 [2] and MC9S08SC4 [3] microcontrollers, which resulted in reduced design time in order of 10 folds, along with high quality generated RTLs.
.

Index Terms: Pad; Connection; Padring; Pad multiplexing

I. INTRODUCTION

In the design of a microcontroller SoC (System on Chip), pads definition and their interconnections are one of the most time consuming and error prone step in the integration RTL (Register Transfer Logic) process [4][5]. To achieve the specification of a design, the number of variables related to functionality that a specific pin will present reveals an extremely complex number of connections that needs to be made [6].

A fast and reliable design of padring code speeds up the integration cycle, reduces the cost of new SoCs [4] and decreases time to market of new designs [7]. There are two relevant aspects in padrings; one is the physical placement and organization [4][5][8] and the other is the logical perspective, which includes the functional and test pin multiplexing [8].

The regular approach to make a padring in microcontroller is to use manual data input and a set of partial scripts and spreadsheet to help the generation of RTL code. But that is technology-dependent and minimally takes advantage by reuse of previous designs data [5][6]. There are several software in the market to help the generation of padring but they are not well integrated with the diversity of pad multiplex codes and they are primarily used for physical padring organization and placement generation [6].

The padring RTL is normally based on similar designs to reduce the need of re-encoding entire RTL and uses portions of previous design data, changing some variable names and making small modifications manually, with the help of short script and spreadsheet. This kind of reuse is naturally subject to mistakes. The effort and risk increase with higher number of pads in each microcontroller design and complex functionalities multiplexed in these pads.

The physical padring planning and placement are not addressed by this work. This paper only addresses the generation of RTL code of pad cells and their interconnection with other blocks of a microcontroller design.

In section II, a background of pads and their connections are presented. In section III, a complete set of blocks that compose a microcontroller padring is presented. Section IV presents how the information is organized in this methodology and how it is translated to metadata structure. In section V, the results obtained using this methodology are shown. Finally, in section VI some conclusions are drawn.

II. BACKGROUND

Pad is a physical cell that serves as interface from the external world, giving access to the internal circuit. There are many kinds of pads such as: power, corner, spacer, high-voltage, analog and digital input/output, isolation and so on. Each pad cell has a specific function to perform in the device.

Each pad has many signals that control the behavior of the associated pad cell like direction of signal, pull up, pull down, whether the signal will carry power, digital or analog, input or, output buffers and many other characteristics that are specific to each type of pad cell.

Each type of pad cell has a set of control signals that vary in number and function. These signals control the behavior inside the circuit of the pad cell. The correct choice of pad cell is important to achieve the specified function in the design.

IO pads compose most of the pad cells in a microcontroller design. Because of the limited pin number in a device, these pads need to share multiple functions [10] including test functions [11] and they are required to be connected to many peripheral blocks that make up a microcontroller SoC.

For example, a pad can be used as an analog signal input for a specific IP (Intellectual Property) like AD converters; or as input/output of a generic port; or as clock input for timers; and even it can be used for test functionality. So the pad cell is constructed such that it can perform many functions in a shared mode.

For each function to which a pad can be assigned, there is a given set of control signals required to be configured in the inputs of that pad cell. Also there are output signals from that cell (pad) that need to be routed to the correct module or internal connection.

Exemplifying in numbers, these connections can vary from 10,000 in small SoCs to 100,000 or more in complex SoCs. In Figure 1 a basic illustration of a microcontroller padring is shown.

III. BASIC BLOCKS OF A MICROCONTROLLER PADRING

In this section, some basics of microcontroller padring are presented. Padring is composed with some few blocks and their interconnections.

Following, for better understanding of this methodology each block is detailed and illustrated.

A. PAD cell

PAD is the physical cell that allows communication to the external world. The main propose of a pad is to interface the external world signal to internal circuit or to output a specific internal signal.

There are many types of pad cells, each one with specific circuit that handles the function needed (power, high voltage, analog IO, digital IO, corner, spacer, differential, etc).

Each pad has its own set of inputs and outputs. The inputs are the signals that control the behavior of the pad, configuring the circuit functionality embedded in each type of pad, also including the internal peripheral signal to be delivered by pad to the external world.

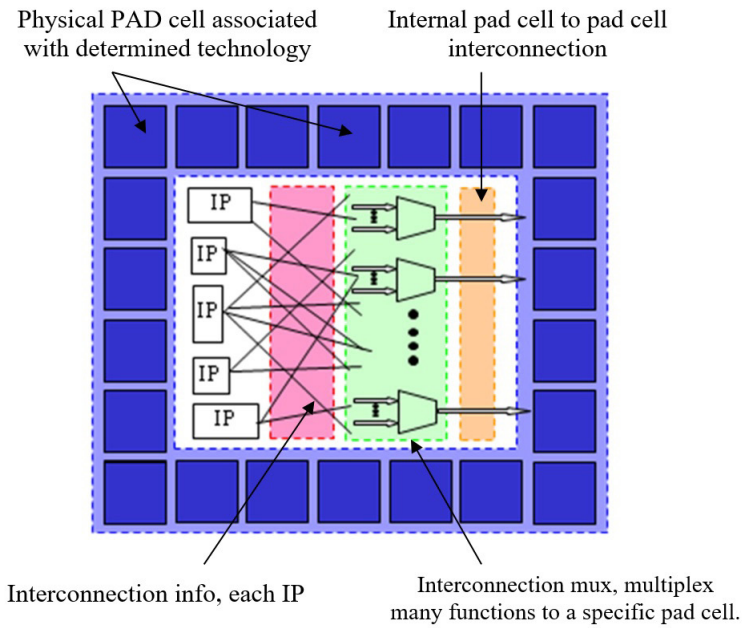


Figure 1. Microcontroller padring.

The outputs from pad cells are the result of a signal that comes from the external world to internal circuits or peripheral. Figure 2 shows a diagram of pad block and how it is connected inside the device.

For each type of pad, the number of input/output signals varies. There are cases with no input or output signals like corner pad cells. Also there are others with many inputs and outputs like general IO pad cells.

B. Mux PAD

Mux Pad block is a module composed with many multiplexers that has the function of routing the correct signal to the pad cell. Usually mux block interconnection is unique for each SoC design because of specific requirements of each microcontroller design.

IO pads usually share many functions and should be connected to specific block or circuit depending how we want to use the pad. For each IO pad there are many configuration possibilities for input and output connections to achieve the desired functionality.

To comply with this flexibility, all related pad signals for configuration and input/output should be multiplexed and be available to the pad when the pin is configured for a specific function. Figure 3 illustrates the block diagram of multiplexer PAD and their connections.

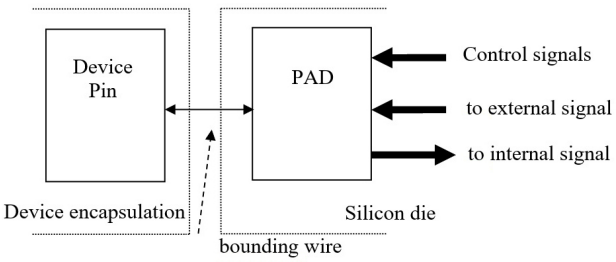


Figure 2. PAD macro block.

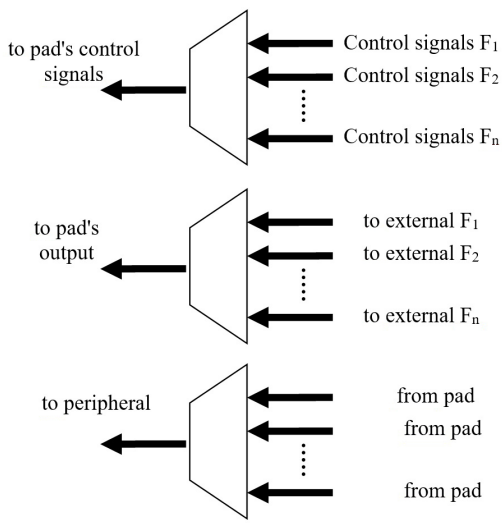


Figure 3. Mux PAD block.

B. Control Port

Control Port block is a circuit composed by registers that store the values which define how the pad cell is configured, as shown in Figure 4. Inside this block the information of which function the pad will perform in a given moment is stored, generating and driving the correct control signals to the mux pad block. It is also responsible for defining the direction of pin and how the internal buffers and multiplexers inside the pad cell will behave as required.

The communication with this block is done using the CPU, which has the control to write the information needed in each register. Usually such information is written during the initialization of a microcontroller program. However, if a pad needs to perform more than one function during program execution, these configurations can be set dynamically.

This block is also responsible for working as generic IO port, storing the data to be output by the pad cell, or capturing the external device data and presenting to CPU. Each instance of this block usually references a set of port pins in the microcontroller (PTA[n:0], PTB[i:0] and so on).

D. General microontroller padring

All these three blocks presented so far compose the padring of a microcontroller, as shown in Figure 5.

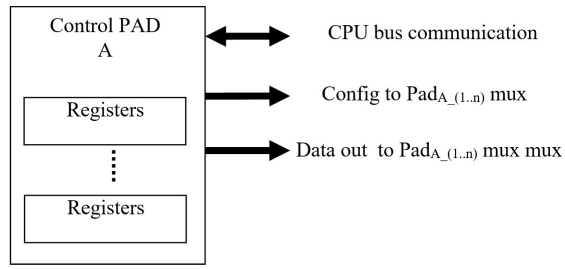


Figure 4. Control Port A block diagram.

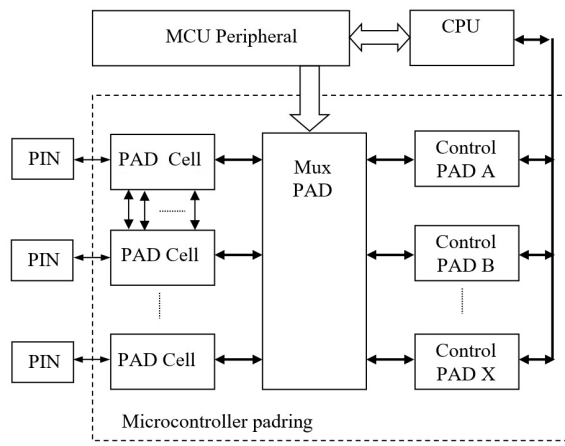


Figure 5. Microcontroller padring components diagram.

In the RTL code that instantiate the pad cells, mux pad and port control, we also have the interconnections of all other pad cell signals (eg: power ring signals) which do not route to a peripheral or other blocks.

IV. METADATA AND DATABASE

All information related to these connections, pad cells and blocks needs to be standardized and stored in a database to be used in the RTL generation.

Each SoC has a unique padring RTL code and needs to be regenerated with specific parameters at every new design. The first metadata is the pad metadata and is composed of various parameters. An example of a pad metadata is shown in Table 1.

Technology name defines the set of pad cells that have common electrical and technology characteristics like pads with the same voltage or a specific technology (such as 250nm, or 180nm, etc.).

To facilitate the use of various technologies we define an *alias* name for each kind of pad cell which is stored in the Connection Name field. The *alias* is related to the functionality embedded in each cell like port, power, corner, etc. Usually this name is chosen to reflect the function that this specific pad set can perform.

The pad cell name is also included in this metadata in the Pad Name field in order to allow the correct and clear cell selection. This information is needed because the pad type (function) may correspond to different pad cell names depending on the chosen technology.

The Mux Possible information is a flag (true/false) that defines if multiple functions will be shared by that pad. Generally this flag is true in the IO pad cells. It opens the possibility to add multiplexing of any kind of pad cell.

The last information we add for this metadata is the Connection fields. They show pins and signals that should be connected to each pad cell port. At this point, we should notice that a peripheral signal that

Table 1. Pad cell metadata

Technology				
io90u3v_V2				
Conection Name				
PT				
PAD NAME				
p_io_3v_4m				
MUX POSSIBLE				
TRUE				
Conection				
Pin Name	**Pin Type**	**Connect To**	**Connect To Type**	**Description**
IPP_INA_3V	output	ipp_ina_$NAME	output	
IPP_INA_MUX_3V	output	ipp_ina_mux_3v_$NAME_nc	wire	
IPP_OUTA_MUX_3V	inout	ipp_outa_mux_3v_$NAME_nc	wire	
IPP_OUTA_3V	inout	ipp_outa_$NAME	inout	
PAD	inout	pad_$NAME	inout	
IPP_IND	output	ipp_ind_$NAME	output	
IPP_IND_3V	output	ipp_ind_3v_$NAME_nc	wire	
IPP_IBE	input	ipp_ibe_$NAME	input	Input Buffer Enable
IPP_IFE	input	ipp_ife_$NAME	input	Input Filter Enable
IPP_DO	input	ipp_do_$NAME	input	Data Out
IPP_OBE	input	ipp_obe_$NAME	input	Output Buffer Enable
IPP_ODE	input	ipp_ode_$NAME	input	Open Drain Enable
IPP_DSE	input	ipp_dse_$NAME	input	Drive Strenght Enable
IPP_PUE	input	ipp_pue_$NAME	input	PullUp Enable
IPP_PUS	input	ipp_pus_$NAME	input	PullUp polarity Select
IPP_SRE	input	ipp_sre_$NAME	input	Slew Rate Enable
VBUF_3V	inout	VBUF_3V	wire	
VDD_3V	inout	VDD_3V	wire	
VSS_3V	inout	VSS_3V	wire	
VDD_LV	inout	VDD_LV	wire	

connects to padring is also standardized. With standardization, it is possible to change de name of a signal with addition of a variable.

Pin Name and Pin Type fields identify the pad pin and its direction. In the connection side, we have also the name and the type of what will be connected in that pad pin to generate the RTL code. If the connection type is a wire, then an internal signal will be generated and will not be presented as an input/output of the padring port interface in RTL code. Generally, this wire connection is an internal pad cell to pad cell connection (physical padring connection).

A brief Description is added, so this information can be propagated into the generated RTL code and helps human interpretation.

The second metadata relates to the function a pin can support. Each Function has a unique defined name. This name is used in a higher-level table to select the correct function each pin can perform for the current microcontroller SoC design. See Table 2 for more details.

If the design needs a different connection other than the one used in previous designs, a new function metadata needs to be added to the database, reflecting the modified connection. A unique name for this newer function needs to be created.

The connection value to the pad can be a hard coded value (0/1) or a signal name. Notice that some names have internal variables attached and are used with higher metadata information to generate the correct signal name. These names and connections are generated at the end of RTL generation process.

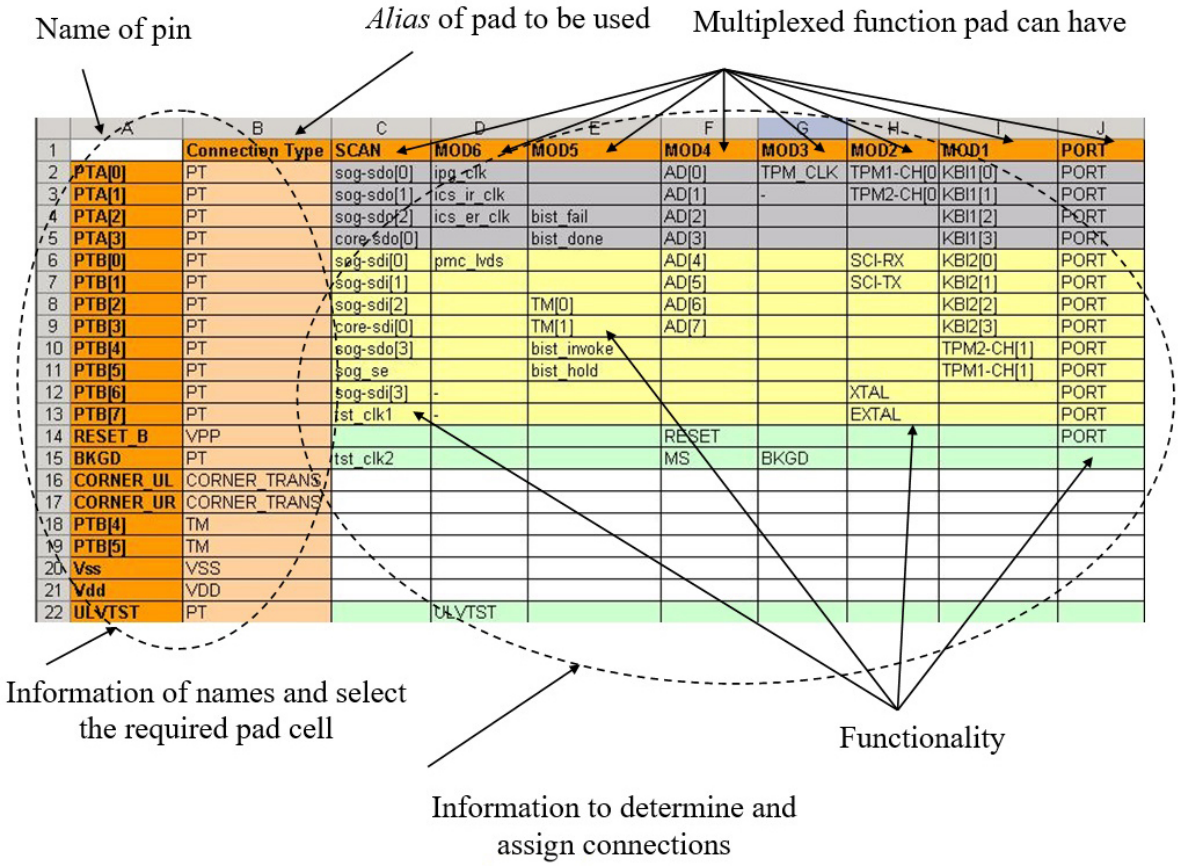
Each technology has its own set of functions and the function names correlate to the same functionality throughout technologies. It is not shown in Table 2 but each function is a subset metadata of a specific technology.

Adding a new function for a new peripheral usage is just as easy by including a new table indicating how the new functionality configures the pad control signals. Also exporting to other technology is just a matter of changing signal names from one technology pad set to another.

The third and last metadata is the project metadata and stores the specific information of each microcontroller SoC design, as shown in Figure 6. Using this methodology, most of the padring definition and development of new SoCs is made with the support of this metadata.

Table 2. Function metadata.

Function Name	
PORT	
PinName	**PinVal**
ipp_ind_$MOD_$PIN_NAME	ipp_ind_port_PIN_NAMEPIN_NUMBER
ipp_obe_$MOD_$PIN_NAME	ipp_obe_port_PIN_NAMEPIN_NUMBER
ipp_do_$MOD_$PIN_NAME	ipp_do_port_PIN_NAMEPIN_NUMBER
ipp_ode_$MOD_$PIN_NAME	0
ipp_dse_$MOD_$PIN_NAME	ipp_dse_pctl_PIN_NAMEPIN_NUMBER
ipp_pue_$MOD_$PIN_NAME	ipp_pue_pctl_PIN_NAMEPIN_NUMBER
ipp_pus_$MOD_$PIN_NAME	1
ipp_ibe_$MOD_$PIN_NAME	ipp_ibe_port_PIN_NAMEPIN_NUMBER
ipp_hys_$MOD_$PIN_NAME	1
ipp_sre_$MOD_$PIN_NAME	ipp_sre_pctl_PIN_NAMEPIN_NUMBER
ipp_ife_$MOD_$PIN_NAME	1
Function Name	
TPM-CH	
PinName	**PinVal**
ipp_ana_en_$MOD_$PIN_NAME	0
ipp_port_en_$MOD_$PIN_NAME	ipp_port_en_$FUNC_NAME$FUNC_NAME_NUMBER$FUNC_PIN_NAME$FUNC_PIN_NUMBER
ipp_ind_$MOD_$PIN_NAME	ipp_ind_$FUNC_NAME$FUNC_NAME_NUMBER$FUNC_PIN_NAME$FUNC_PIN_NUMBER
ipp_obe_$MOD_$PIN_NAME	ipp_obe_$FUNC_NAME$FUNC_NAME_NUMBER$FUNC_PIN_NAME$FUNC_PIN_NUMBER
ipp_do_$MOD_$PIN_NAME	ipp_do_$FUNC_NAME$FUNC_NAME_NUMBER$FUNC_PIN_NAME$FUNC_PIN_NUMBER
ipp_ode_$MOD_$PIN_NAME	0
ipp_dse_$MOD_$PIN_NAME	ipp_dse_pctl_PIN_NAMEPIN_NUMBER
ipp_pue_$MOD_$PIN_NAME	ipp_pue_pctl_PIN_NAMEPIN_NUMBER
ipp_pus_$MOD_$PIN_NAME	1
ipp_ibe_$MOD_$PIN_NAME	ipp_ibe_$FUNC_NAME$FUNC_NAME_NUMBER$FUNC_PIN_NAME$FUNC_PIN_NUMBER
ipp_hys_$MOD_$PIN_NAME	1
ipp_sre_$MOD_$PIN_NAME	ipp_sre_pctl_PIN_NAMEPIN_NUMBER
ipp_ife_$MOD_$PIN_NAME	1
ipp_offval_$MOD_$PIN_NAME	0



	A	B	C	D	E	F	G	H	I	J
1		Connection Type	SCAN	MOD6	MOD5	MOD4	MOD3	MOD2	MOD1	PORT
2	PTA[0]	PT	sog-sdo[0]	ipg_clk		AD[0]	TPM_CLK	TPM1-CH[0	KBI1[0]	PORT
3	PTA[1]	PT	sog-sdo[1]	ics_ir_clk		AD[1]	-	TPM2-CH[0	KBI1[1]	PORT
4	PTA[2]	PT	sog-sdo[2]	ics_er_clk	bist_fail	AD[2]			KBI1[2]	PORT
5	PTA[3]	PT	core-sdo[0]		bist_done	AD[3]			KBI1[3]	PORT
6	PTB[0]	PT	sog-sdi[0]	pmc_lvds		AD[4]		SCI-RX	KBI2[0]	PORT
7	PTB[1]	PT	sog-sdi[1]			AD[5]		SCI-TX	KBI2[1]	PORT
8	PTB[2]	PT	sog-sdi[2]		TM[0]	AD[6]			KBI2[2]	PORT
9	PTB[3]	PT	core-sdi[0]		TM[1]	AD[7]			KBI2[3]	PORT
10	PTB[4]	PT	sog-sdo[3]		bist_invoke				TPM2-CH[1]	PORT
11	PTB[5]	PT	sog_se		bist_hold				TPM1-CH[1]	PORT
12	PTB[6]	PT	sog-sdi[3]	-				XTAL		PORT
13	PTB[7]	PT	tst_clk1	-				EXTAL		PORT
14	RESET_B	VPP				RESET				PORT
15	BKGD	PT	tst_clk2			MS	BKGD			
16	CORNER_UL	CORNER_TRANS								
17	CORNER_UR	CORNER_TRANS								
18	PTB[4]	TM								
19	PTB[5]	TM								
20	Vss	VSS								
21	Vdd	VDD								
22	ULVTST	PT		ULVTST						

Figure 6. Project metadata..

In Column A we add the name for each pad instance. This is the name we desire in the instance declaration of the RTL when the pad is instantiated. The name needs to be unique and can be created using brackets.

In Column B we select the "Connection Type" which is related to *alias* used in the pad cell metadata. The selection name can only be chosen from possible *alias* names stored in the metadata database, thus avoiding the usage of non-existing cells in a specific technology.

Columns C through J are related to the mux pad block and they define each possible function that this pad will perform in the specific SoC design. Again, the function name is selected from possible names stored in the metadata database and can be composed with brackets for more than one pad with same function.

There is additional information needed for the correct generation of RTL code, as illustrated in Figure. 7.

The Author Name is inserted in the RTL code header; Technology field selects the technology pad set in the metadata database. Using this methodology, we can easily change the pad cell set for different technology. If the selected pad cell set does not present a function defined in the metadata database, then the spreadsheet is colored in red warning this function metadata needs to be included in the selected technology metadata.

Another pad information we need to provide is the signal values when the pad is not used - Off Val Values -so the pad mux block can provide a default value when a not exist function is selected.

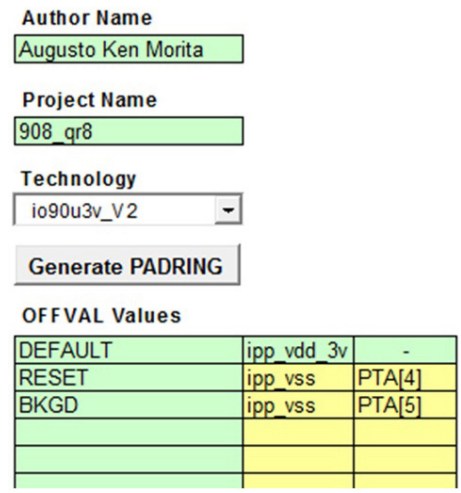


Figure 7. Project specific.

V. USAGE AND RESULTS

Using this methodology and providing the defined data described in this work, we can generate all RTL code for a microcontroller padring. Some noticeable results are summarized in Table 3. The old method values are estimated based on similar designs schedules. The projects using the new method resulted significant improvement in both execution time and complexity reduction in the padring RTL code generation.

This work was applied in three SoC designs, the 8-bit microcontrollers from Freescale. For the first one, MC9S08MP16 [1], we obtained the chip integration values with the old methodology and then the auto generated code was tested. In the other two designs, MC9S08QB8 [2] and MC9S08SC4 [3], we used the automated RTL generation code from the start. It is relevant to mention that the two last designs had zero defects in padring code. Also we applied the new method in three test vehicles to study a new low power processor design.

It was easy to find some inconsistencies between designs when using the previous method and then allowed us to correct this information by standardizing the padring code.

During the execution phase, this process accelerated the definition/modification of pin specifications. No manual input for signal assignment in addition to the use of metadata gathered from previous projects made it easy to visualize and make the global pin assignment in the design.

To perform the automatic code generation, this work was programmed using Visual Basic language. Spreadsheet was used as a way to provide data in a more friendly way, but all information about connection and multiplexing signals of the method is embedded in the program code. The use of spreadsheet also helps to visualize inconsistencies that might occur during the design setup and development.

The metadata tables are converted to XML (Extensible Markup Language) data format and then the generated metadata database is used for RTL coding. Using database in XML format eliminates the necessity of complex database program or structure.

Pad cells may have subtle (eg: logic active level) or significant differences (eg: more or less signals). The metadata that translates the desired function into the required pad IO signals will handle the generation of the correct connection to each type of pad cells in the RTL.

Only a few designers need to know deeply about the pad signals functionality in order to generate the pad and function metadata. Most integration designers will only reuse the previously defined pad types and functions. If required, the metadata can be easily read, modified, or added. The pad and function definition is done through regular spreadsheet tables and automatically included in the metadata database. Less error is inserted in new SoC designs because only incremental information (pads/functions not used in previous projects) is added to the database.

VI. CONCLUSIONS

Selecting the pad through its type or *alias* (not the pad cell itself) allows smooth migration to other technologies. A given pad type/*alias* may be associated to different pad cells for each technology.

Tedious and subject to error signal-by-signal approach is a risk to final RTL/design quality. It requires lengthy human reviews and verifications, which can be avoided with the currently presented method.

Every new use enriches the metadata database, yet preserving and leveraging the reuse aspect. Subsequent SoCs require, if necessary, only a few incremental pad/functions. The project metadata facilitates the global pin assignment.

Table 3. Comparison of work time needed per type of changes.

	Old Methodology	New Methodology
New Pad spec: • Define functions • Define Pads • Generate RTL and Verification files	2-3 weeks	2-3 days
Modifications: • Change few pads definitions • Regenerate RTL and Verification files	2-4 days	1-2 hour
Technology changes: • Replace all pads • Review connections • Regenerate RTL and Verification files	1-2 weeks	1-2 days

The method can be extended towards physical padring data and processing.

The use of the method in commercial microcontroller projects demonstrated padring design time reductions of as high as 10 times, along with no defects in the generated RTL codes.

REFERENCES

[1] http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08MP16DS.pdf

[2] www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08QB8.pdf

[3] www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08SC4.pdf

[4] A. B. Chong, H. K. Chun, et al., "Unified Padring Design Flow." 2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks, pp. 418- 423, Jun. 2013

[5] A. B. Chong, H. K. Chun, et al., "Unified Padring Design Flow - New Developments and Results" Artificial Intelligence, Modelling and Simulation (AIMS), 2013 1st International Conference, pp. 462- 466, Dec. 2013

[6] http://www.ispd.cc/slides/slides10/2_04.pdf

[7] http://www.defactotech.com/star-rtl-build-signoff/star-padring

[8] Ming-Fang Lai, Hung-Ming Chen,"An Implementation of Performance-Driven Block and I/O placement for chip-package codesign," ISQED, 2008

[9] C. Alcom, D. Dworak, N. Haddad, W. Henley, and P.Nixon, "Kerf Test Structure Designs for Process and Device Characterisation," Solid State Technology, pp. 229-235, May 1985.

[10] D. Ward, A. J. Walton, W. G. Gammie, and R. J. Holwill, "The use of adigital multiplexer to reduce process control chip pad count," in Proc. 1992 Int. Conf. Microelectronic Test Structures (ICMTS 92), pp. 129-133, 1992.

[11] A. Nishimura et al., "Multiplexed test structure: A novel VLSI technology development tool," in Proc. IEEE VLSI Workshop Test Structures, pp. 17–18, Feb. 1986.