Large Scale IP prototyping

Guri Kamelo

Nadav Rosenblat

Intel Corporation

Haifa, Israel

ABSTRACT

The requirement for FPGA prototyping for big designs has increased greatly over the passing years . One of the main tasks for this is IP partitioning to spread upon several FPGA devices. This introduces several main challenges:

1. *Correct Logic Spread & partitioning to keep minimal FPGA utilization and eliminate route and timing issues.*
2. *Data crossing from FPGA to FPGA (usually done over TDM) introduces the challenge of choosing ideal TDM ratios to keep maximal performance.*
3. *Planning correct Logic placement to enable IO access such as PCIe or GTX.*

This paper will review the main key stages in prototyping using proto compiler and other tools. we will review the main issue we had , and our approach for solution i.e clock and reset , partitioning , interconnect & constraints

Table of Contents

Table of Figures, Tables, Images

1. Introduction – why prototyping?

SoC / IP Prototyping has three main objectives:

a. *Shift Left – this term implies shortening development time of the following:*
   Firmware (FW) development while Si Device is unavailable.
   Driver Software development while Si Device is unavailable
   Various applications development. By this we usually mean SoC / IP tests which are ported from the RTL environment to the FPGA prototyped environment.
   Since our discussion is on IP prototyping then Shift Left is achieved also for the SoC development in which the IP is ported to.
b. *Low Cost vs. Emulation systems.*
c. *Using the application tests from (a) the prototyped IP can be used to detect HW bug, i.e design validation advantage and also reconstructing HW bugs in a more efficient manner.*

2. Prototyping – key principals

Prototyping key principals vs. ASIC design are compared below:

	Asic	prototype
keep functional logic	f(x)	f(x)
keep cycle accurate	1 clock	1 clock
control IP interfaces	full control	partial
IP bandwidth	based on SOC	need to mimic BW range
Push button model	Not there yet	almost there

Table1: key principals

If deviating from these key principals then correct FW/SW or design validation might not be developed correctly. There is a great effort to keep IP functionality untouched. However, this requirement might have its limitations which are generally introduced when running the prototyped RTL through the FPGA synthesizer which could have different requirements vs. RTL compiler. Cycle-accuracy is usually kept but becomes an issue when performing IP partitioning (discussed later on..).

3. IP description

A general overview of the IP we have been discussing can be shown in the following figure:

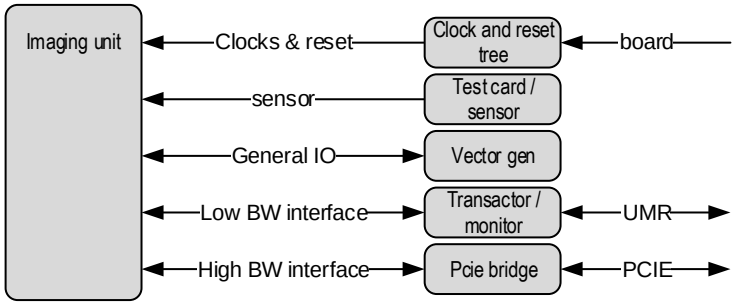

Figure 1: IP description

A general description of a Large Scale IP can be shown in the following figure:

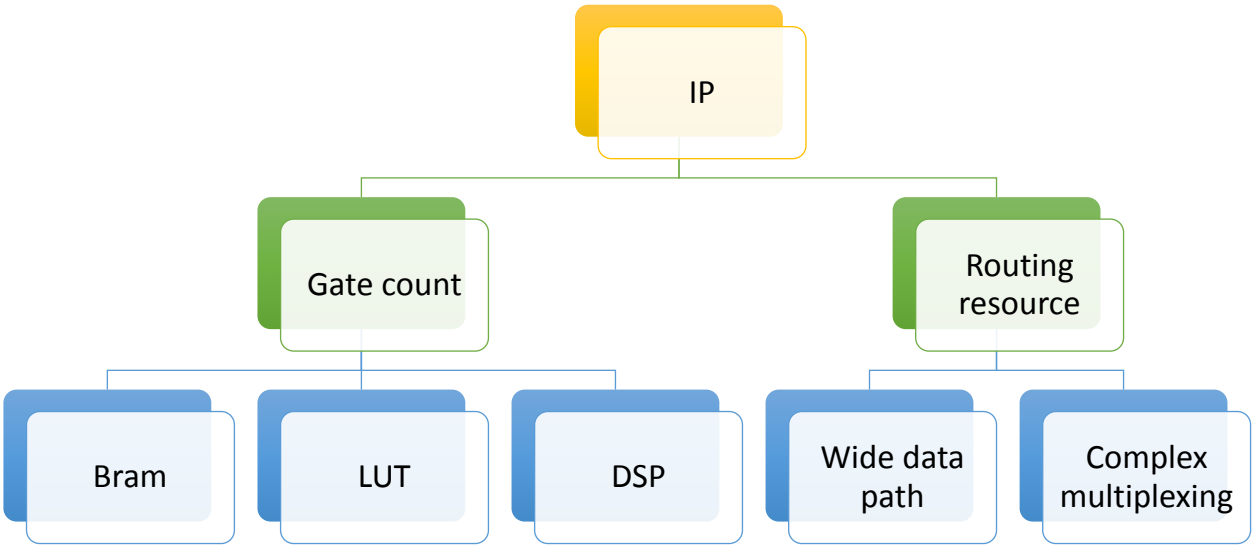

Figure 2: Large Scale IP general content

The challenges of prototyping a Large Scale IP can be divided into two major aspects:

1. *Very large gate count which results in an excessive utilization rate even if using the industry's most dense devices (such as XILINIX Virtex7 series..). A "rule-of-thumb" to describe the high utilization rate is usually above 70%.*
2. *High routing resources which is a result of muxing large interfaces.*

4. Partitioning Principals

It is, of course, advised to try and avoid IP partitioning as it introduces a limitation to the IP clock frequency that can be applied and the partitioning flow is complex and costly. However, if the scale of the IP is such that high utilization will be reached, then partitioning must be performed.

Partitioning usually implies dividing the IP design into several FPGA devices.

The following four items need the most attention:

1. ***Clock Tree*** *– since we are dividing the design, the Clock Tree will also have an effect.*
2. ***Reset Tree*** *– reset tree division is also a factor although mostly less complex then Clock tree.*
3. ***Connection types*** *that are planned to be used: Direct, TDM, HSTDM, GTX …*
4. ***IO limitations****. By this we usually mean limited FPGA pin count.*

4.1. Clock Tree

The IP Clock Tree is composed of various system clks which are used by the original IP design. As a general rule—if there is an IP block which its Clock Tree has to go through partition—use the FPGA board CLK to drive it rather then "transfer" the clock from FPGA to FPGA.

A general Clock Tree partitioning is shown in the figure below

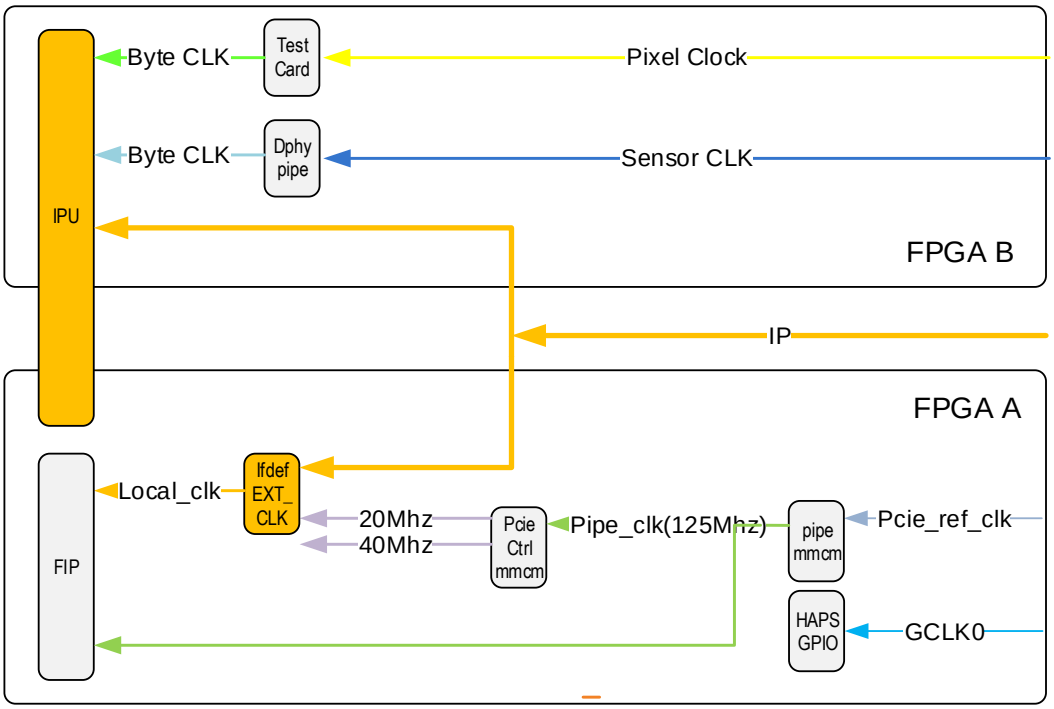

Figure 3: Clock Tree partitioning example

4.2. Reset Tree

It is advised that also the Reset Tree and various reset signals will not be "transferred" from FPGA to FPGA but rather use the FPGA board reset.

TDM components which are used to connect the FPGAs also contain a reset signal. The TDM reset is usually performed 1st. Once TDM reset and TDM training is completed, then the IP reset can be applied. If PCIe (as in our IP example..) is used, then its reset is performed last.

The Reset tree partition and the reset flow can be shown in the following figures

:

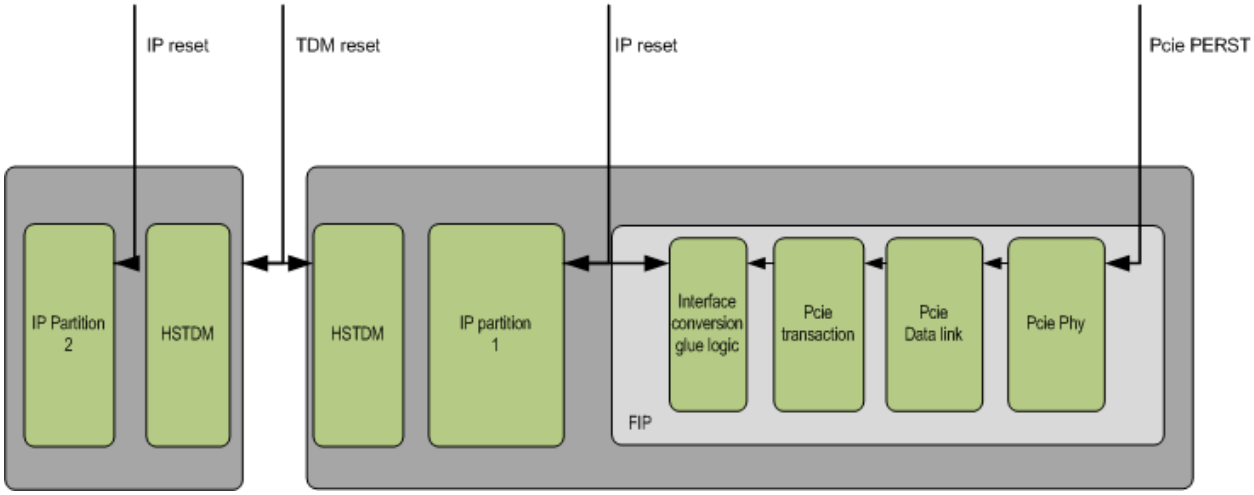

Figure 4: Reset Tree partitioning

Figure 5: Reset ordering

4.3. Connection Types

Several connection types can be considered for FPGA to FPGA signals transfer:

1. *Direct connection—no signal multiplexing.*
2. *Single Ended TDM- usually lower frequency & BW but higher IO benefit.*
3. *High-Speed TDM (HSTDM) – high frequency & BW but lower IO benefit as it is composed of differential signaling.*
4. *GTX – very high frequency & BW but not cycle accurate as previous three.*

Following figure illustrates the above:

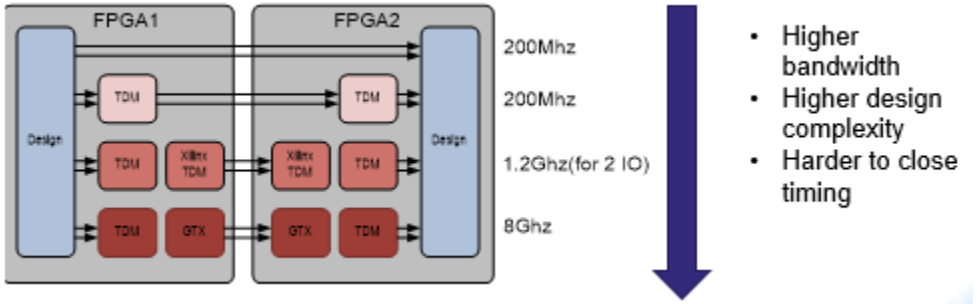

Figure 6: connection types & benefits

4.3.1. HSTDM considerations

The main trade-off in using HSTDM is its latency vs. ratio. If a high signal count is to be routed into the HSTDM then its delay will grow. The total signal delay between FPGAs using HSTDM is shown in the figure below:

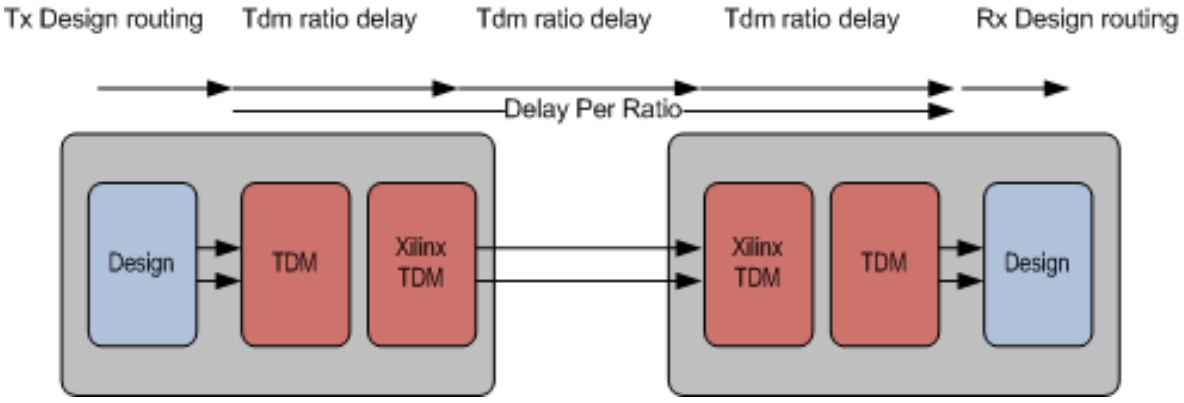

Figure 7: total signal delay using HSTDM

Tx design route & Rx design route is the delay from the last design FlipFlop to the Tx TDM and from the Rx TDM to the first Design FlipFlop. Since the HSTDM is cycle-accurate then the total delay per ratio can be looked upon as a blackbox with a total Max.delay. This delay also includes the physical interface which are basically HapsTrak cables in the HAPS-70 system.

Following table shows the delay values vs. HSTDM ratio on the HAPS-70 system:

LVDS Frequency 1100Mhz	
TDM ratio [for 2 IO]	delay per Ratio[nSec]
8	34
16	50
24	58
32	68
40	76
48	82
56	90
64	98
72	104
80	112
88	118
96	126
104	134
112	140
120	148
128	156

Table2: HAPS-70 HSTDM delay vs. Ratio

The increase in delay if a high TDM ratio is used causes the IP clk frequency decrease.

Following table shows the design delay slack for a given IP clk frequency vs. TDM Ratio dependency:

TDM ratio\ Design clock	1Mhz	5Mhz	6.3Mhz	10Mhz	20Mhz
8	966	166	116	66	16
16	950	150	100	50	0
24	942	142	92	42	-8
32	932	132	82	32	-18
40	924	124	74	24	-26
48	918	118	68	18	-32
56	910	110	60	10	-40
64	902	102	52	2	-48
72	896	96	46	-4	-54
80	888	88	38	-12	-62
88	882	82	32	-18	-68
96	874	74	24	-26	-76
104	866	66	16	-34	-84
112	860	60	10	-40	-90
120	852	52	2	-48	-98
128	844	44	-6	-56	-106

Table3: design delay slack

Values in white are positive slacks. In yellow are marginal slack and in red are negative slacks, i.e. no timing margins. In-order to pass the place & route Static Timing Analysis- positive values must be chosen.

In the HAPS-70 system the interconnects between FPGA to FPGA are done by HapsTrak connectors. The table below describes an example from one of our prototyping projects and shows the amount of signals to be transferred, the number of FPGA IOs required & the number of HapsTrak connectors this will consume:

TDM ratio 64:2				
Source	Destination	signals	Number of IO	Number of connectors
A	B	3128	98	3
A	C	4504	142	3
A	D	5526	174	4
A	A'	5526	174	4
A	B'	2286	72	2
Total A		20970	660	16

Table4: example project- FPGA to FPGA signals, IOs & connectors

The topology in this example is FPGA A connected to 5 other FPGAs (A,B,C,D FPGAs located on one HAPS system, A',B' FPGAs located on a 2nd HAPS system).

5. Place & Route considerations

HSTDMs physical placement on the FPGA also has to be considered. In certain cases, if the HSTDMs are all concentrated in an FPGA layer in which high frequency logic is to be placed, this may cause severe congestion as shown in the image below (taken from XILINX Vivado tool):

Image 1: very high congestion of both logic & HSTDMs

Solution was to place the HSTDMs in a separate layer from the high speed logic. In the HAPS-70 system this is done by choosing different connector numbers which can be mapped to various FPGA layers (SLRs). The image below shows the congestion spread after performing this separation:

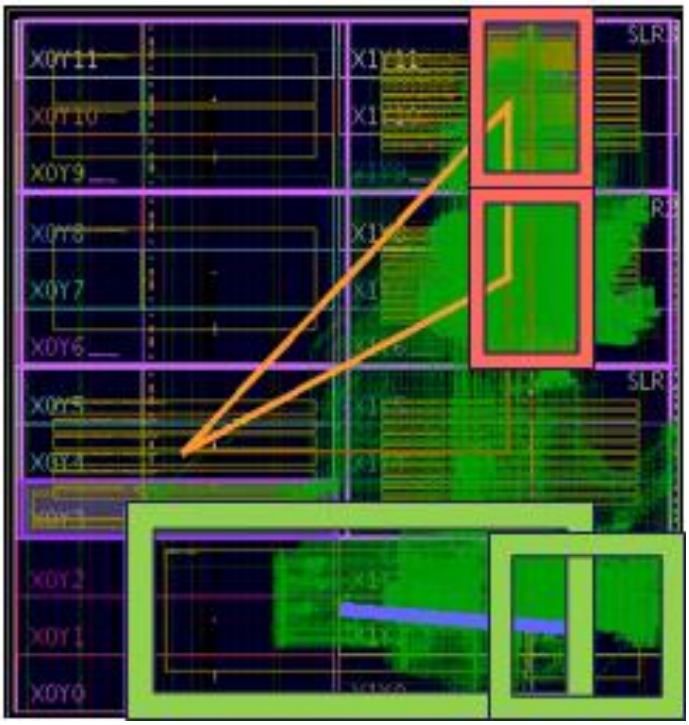

Image 2: congestion spread after HSTDM & logic separation

6. References

[1] Synopsys ProtoCompiler & HAPS-70 S-48 User Guides