Allocation of FPGA DSP-Macros in Multi-Process High-Level Synthesis Systems

Benjamin Carrion Schafer
The Hong Kong Polytechnic University
Department of Electronic and Information Engineering
b.carrionschafer@polyu.edu.hk

Design Automation and Reconfigurable Computing LAB
DARC

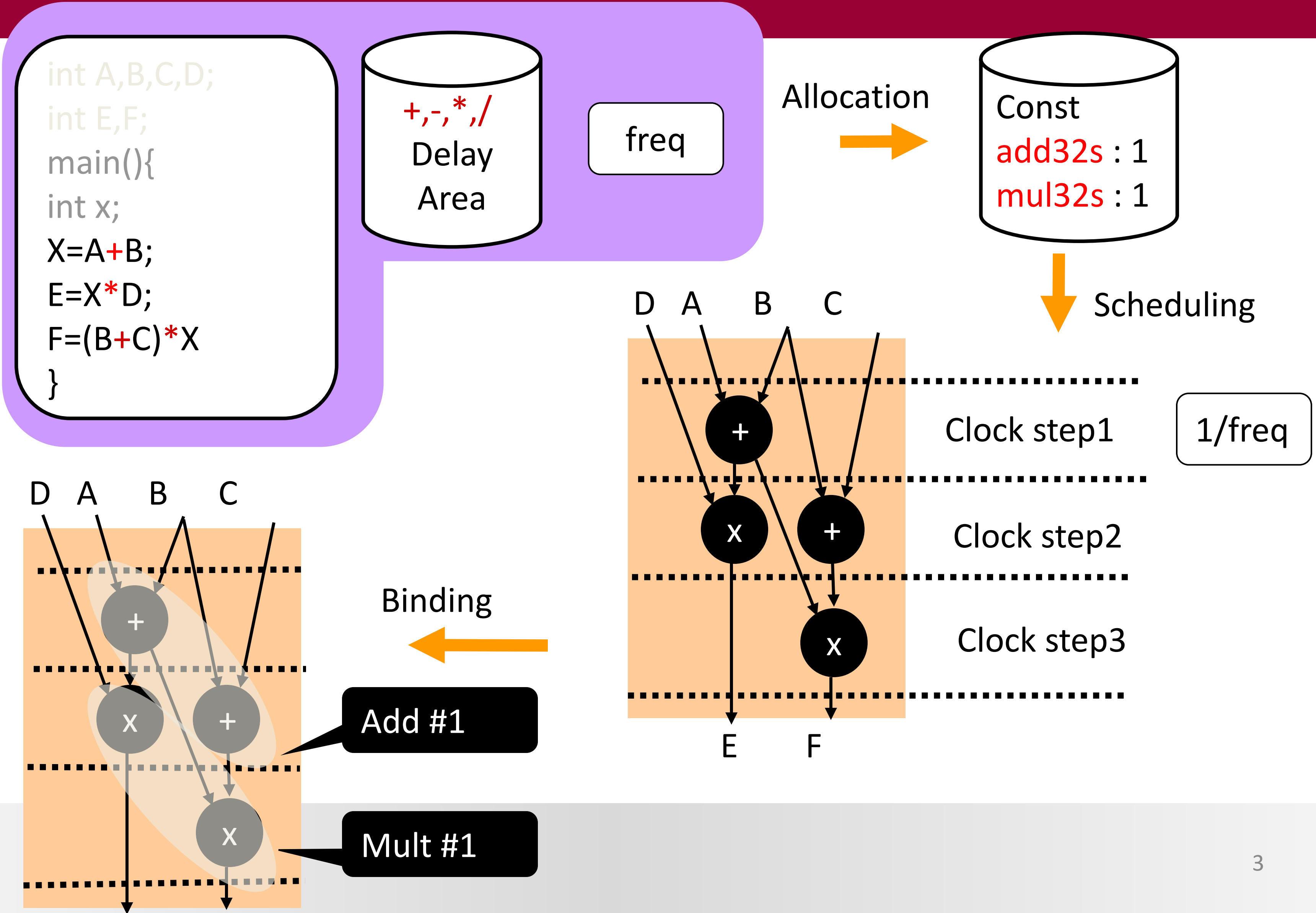
@DARC_lab

The Hong Kong Polytechnic University
香港理工大學

Outline

- High Level Synthesis – Resource Sharing Overview
- Motivational Example
  - ASIC vs. FPGA resource sharing/ functional unit DSE
- FPGA DSP-macros
- Motivation for effective methods to allocate DSP macros across multiple-processes DSP-macro allocation method: *Allocation of DSP-macros for Multiple Processes* (ADSP_MULTP)
  - Step 1 :Functional Unit Design Space Exploration
  - Step 2 : DSP-macro sensitivity computation
  - Step 3 : Sorting based on sensitivity
  - Step 4 : Allocate DSP-macros
- ADSP_MULTP variations
- Experimental Setup and Results
- Conclusions

High Level Synthesis 101



```
int A,B,C,D;
int E,F;
main(){
int x;
X=A+B;
E=X*D;
F=(B+C)*X
}
```

HLS Resource Sharing

Behavioral Description in C

```
char A,B,C,D;
char E,F;
main(){
char X;
X = A + B;
E = X * D;
F = (B + C) * X;
}
```

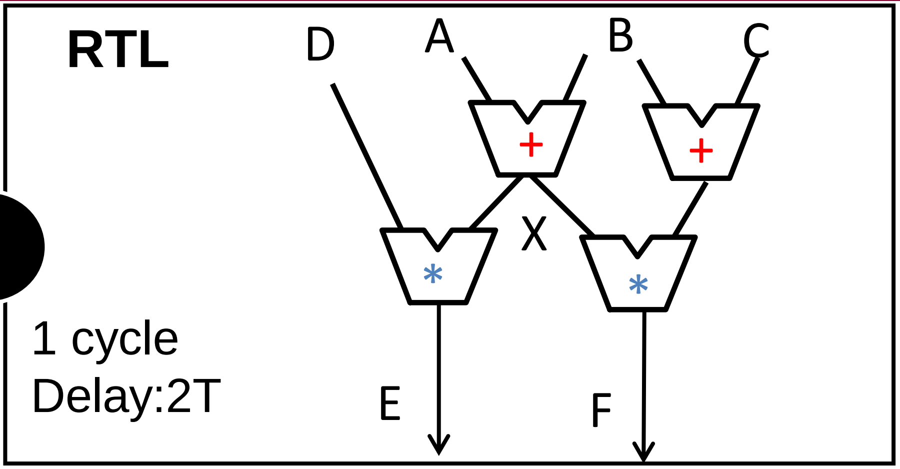
+ : 2
* : 2

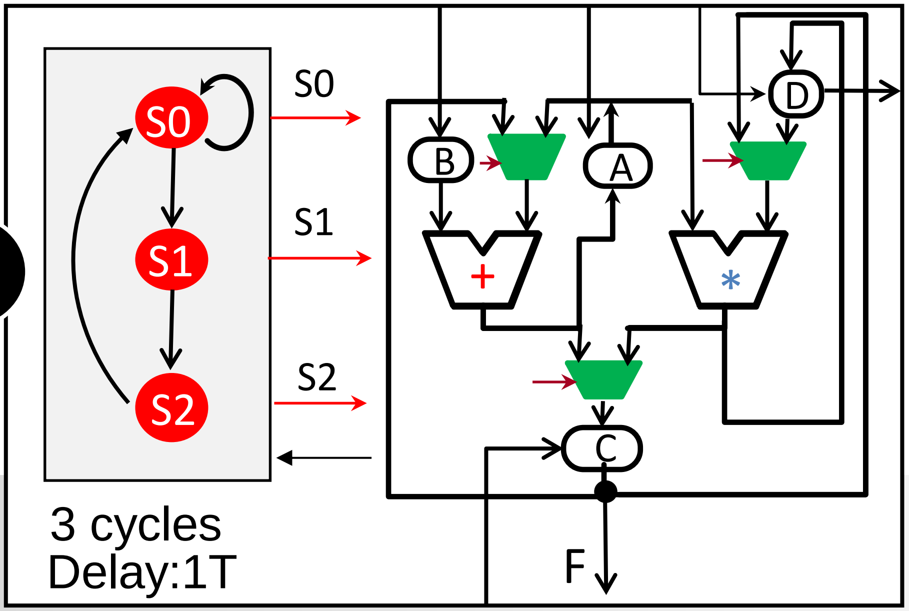
1

FU constraints

+ : 1
* : 1

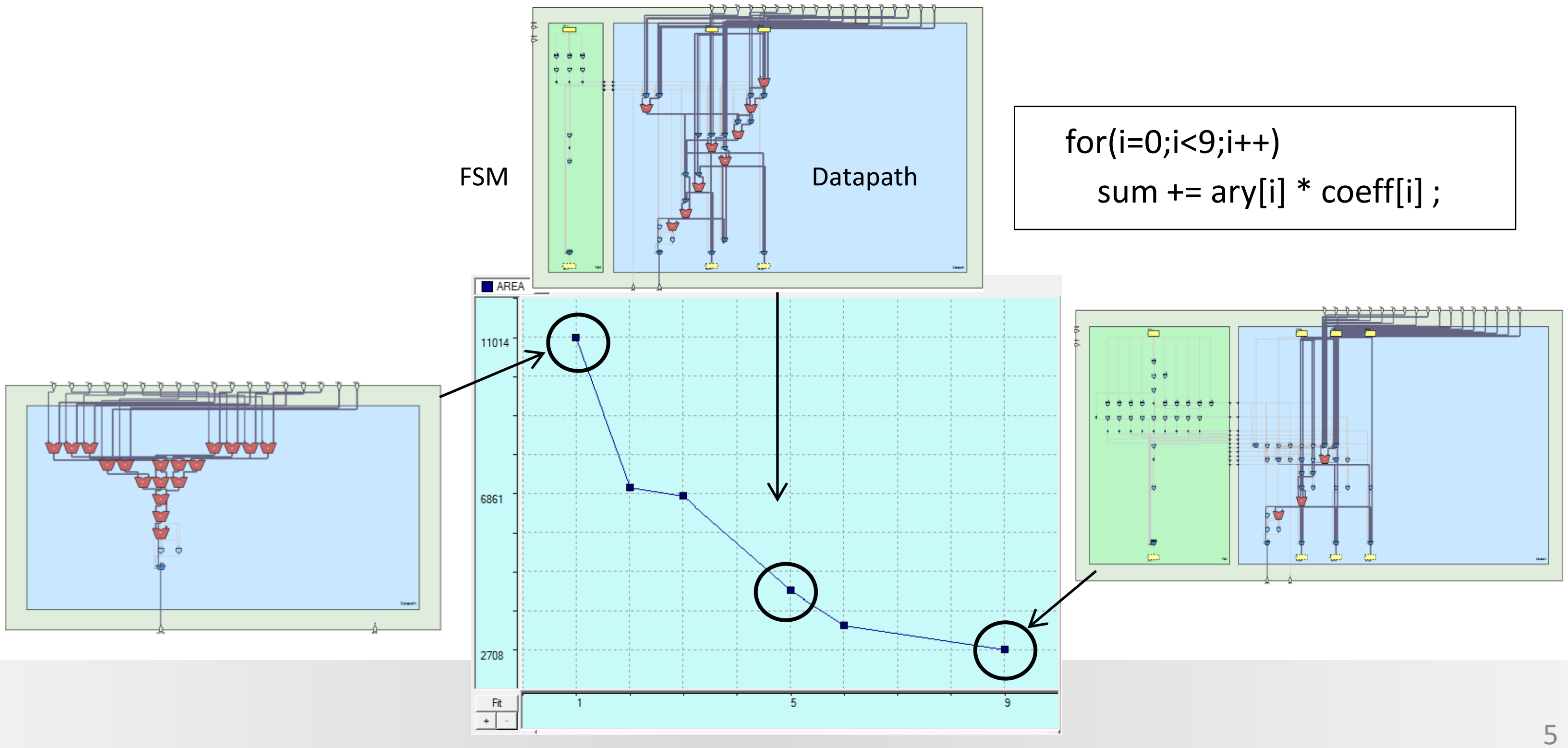
2





ASIC Resource Sharing/FU DSE

- Resource Sharing
  - A single functional unit (FU) is re-used among different computational operations in the behavioral description
  - Can lead to smaller designs
- 9-TAP FIR filter example targeting ASIC Nangate 45nm@100MHz

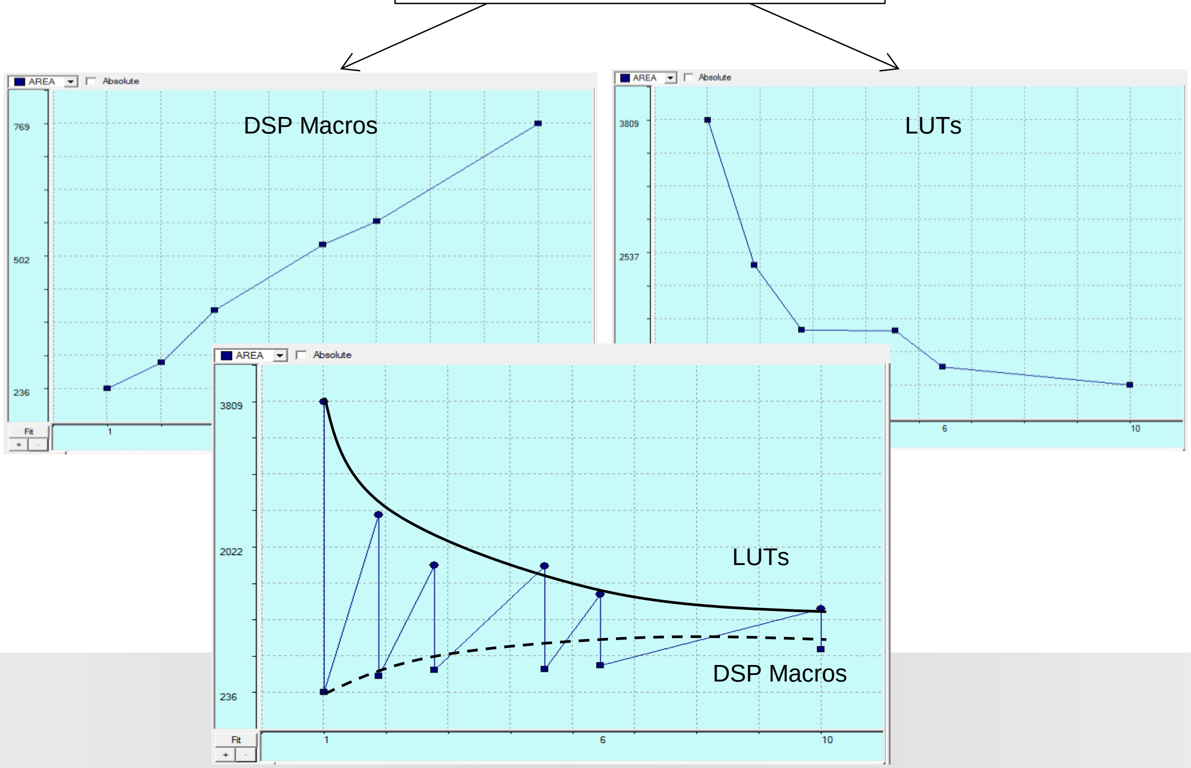


```
for(i=0;i<9;i++)
   sum += ary[i] * coeff[i] ;
```

FPGA Resource Sharing/FU DSE

- Same FIR filter targeting a Xilinx Virtex6 FPGA
- In default mode Area↑ when #FU↓ because the FUs (MAC) are now mapped to the FPGAs DSP macros
- DSP macros are ***free*** in terms of area, while Muxes are not

```
for(i=0;i<9;i++)
   sum += ary[i] * coeff[i] ;
```



Observations when targeting FPGAs

- Always use FPGAs DSP-macros
- Reduce the amount for resource sharing as much as possible

BUT

- FPGAs have increased to a point that entire systems can now be implemented on a single device
- HLS is a single process synthesis → One process is synthesized and optimized at a time

➔ An Effective method to allocate DSP macros across multiple-processes is needed which minimizes the total design Area

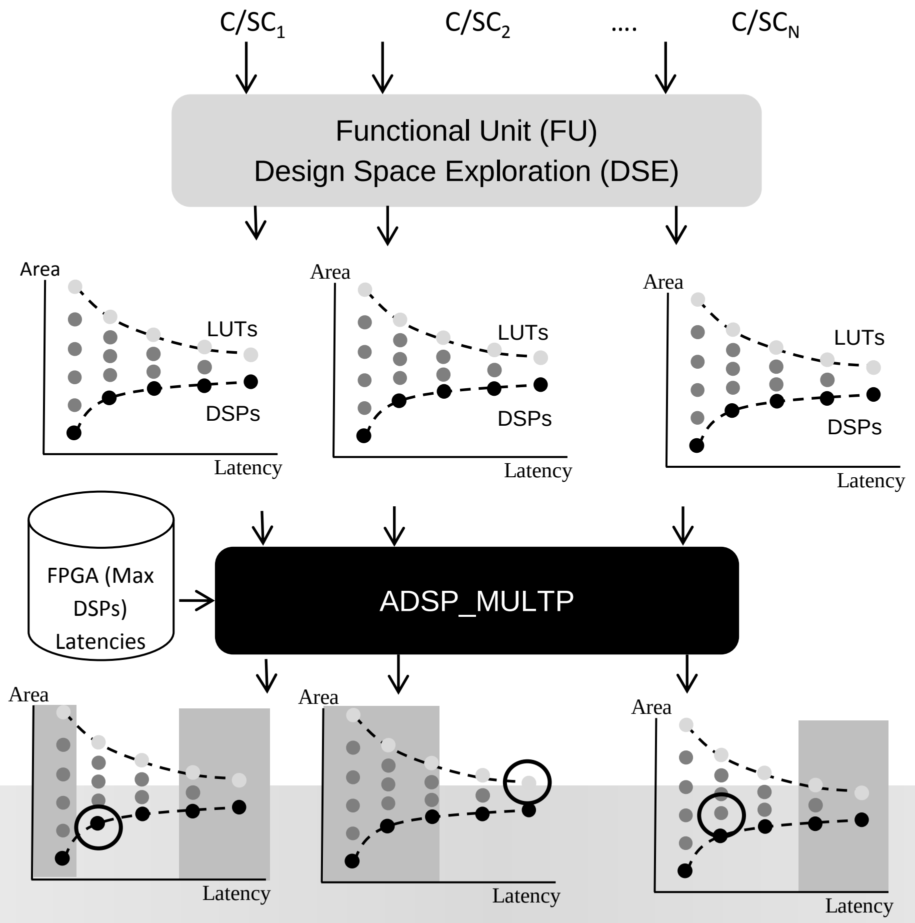
FPGAs DSP-macros

- High-end FPGAs have large number of DSP-macros, but cannot be used for consumer products
- Consumer products a very price sensitive
- DSP applications extremely DSP-macro intensive
- FIR filter consumes 9 DSP48E1s macros and 24 Slice LUTs when fully parallelized

FPGA	# DSP-macros (family dependent)	DSP-macros	Price ($)
Xilinx Virtex7 (high-end)	1,260-3,600	25x18 multiplier, 48-bit accumulator, and pre-adder	X,000 USD
Xilinx Artix7 (low-end)	60-740	25x18 multiplier, 48-bit accumulator, and pre-adder	X USD
Altera Stratix5 (high-end)	512-3,926	18x18multiplier – variable precision multipliers, 64-bit accumulator	X,000 USD
Altera Cyclone5 (60-740)	60-740	18x18multiplier– variable precision multipliers, 64-bit accumulator	X USD

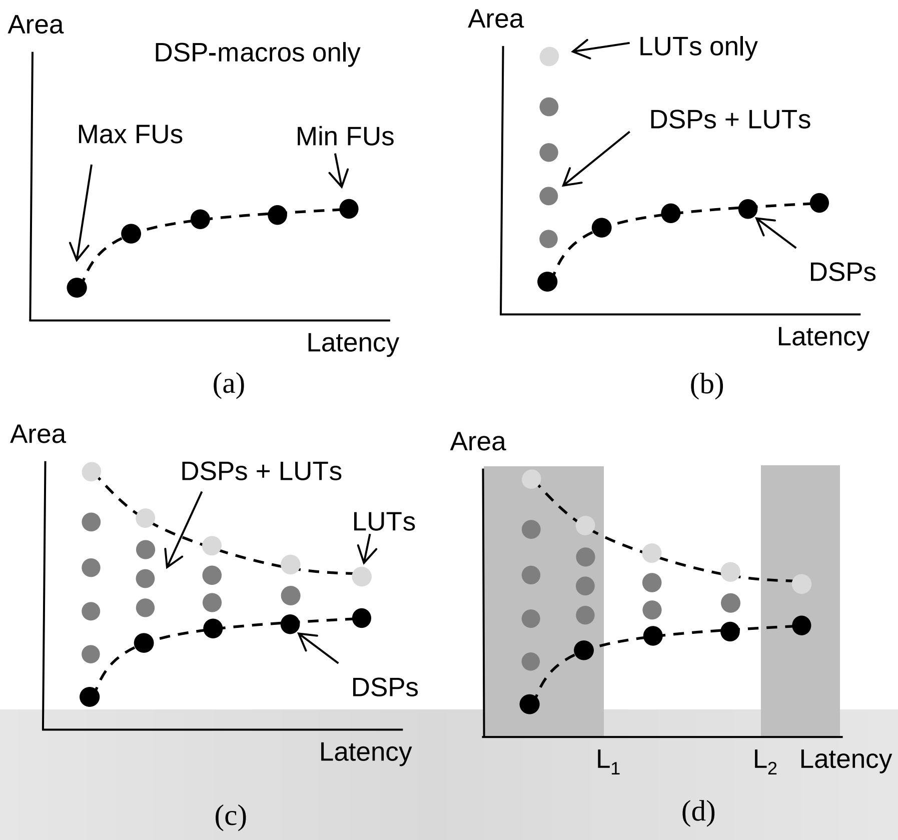
Proposed Method : *Allocation of DSP-macros for Multiple Processes* (ADSP_MULTP)

- 2 main steps sub-divided into 4 smaller
  - Perform FU Design Space Exploration (DSE) for each process
  - Decided how to best allocated the available DSP-macros given a set of latency constraints for each process



Step 1: FU Design Space Exploration

- Perform FU Design Space Exploration (DSE) for each process by:
  1. Synthesize behavioral description in default mode to maximize parallelism and extract FU constraint file with max FUs needed
  2. Reducing the number of FUs by 20% in constraint FILE
  3. Map the FUs to DSP-macros and LUTs



Mapping MAC to DSP-macros or LUTs

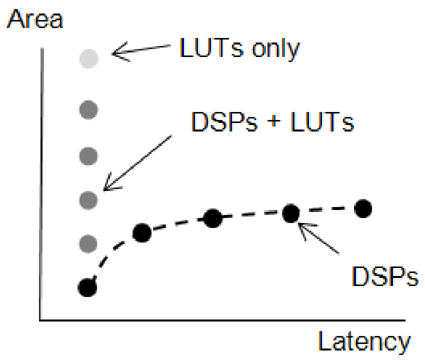
- HLS tools do not allow fine grain controllability of where to map single operations. E.g.

```
for(i=0;i<9;i++)
  sum += ary[i] * coeff[i] ;
```

- How to map X MAC to DSP-macros and 9-X to LUTs?

➔RTL generated by HLS is parsed by the FU explorer and automatically edited adding FPGA vendor specific synthesis directive. E.g. Xilinx:

```
attribute use dsp48 : string;
attribute use dsp48 of mul16s9ot : signal is "no";
attribute use dsp48 of mul16s8ot : signal is "yes";
```



➔ Need to make sure that timing is still met after logic synthesis !

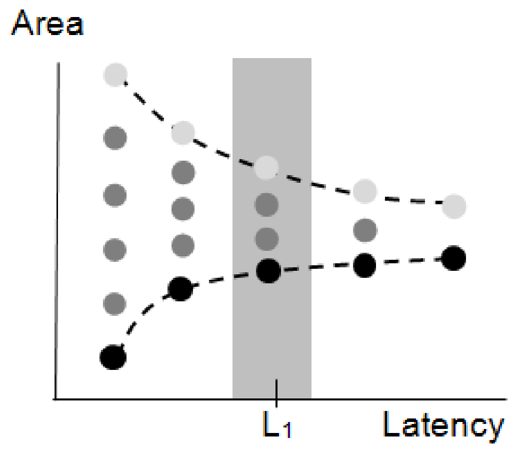
Motivation for Full FU DSE

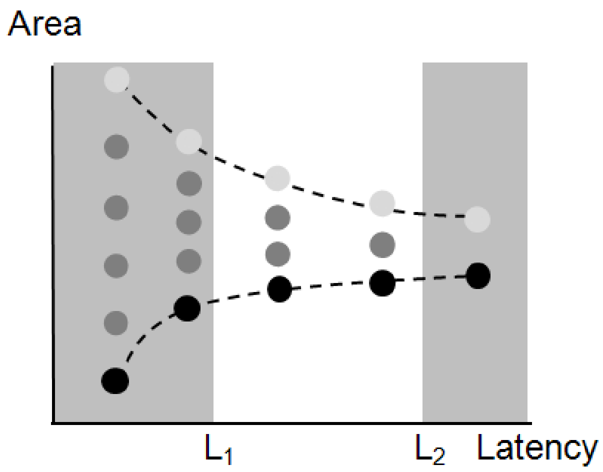
- Most designs returned by the DSE are not Pareto-optimal, although parabolic behavior in some cases. BUT:

- Often the design latency is a **global** constraint (either single or range)
- This constraint can vary during different project stages e.g. when the process is integrated into the system or when it is re-used in later projects.

➔ Full exploration results are stored and the most efficient implementation is selected when the latency or latency interval constraint is specified.

➔ Only those designs within the specified latency interval are considered by our method





Step 2: DSP-macro Sensitivity Calculation

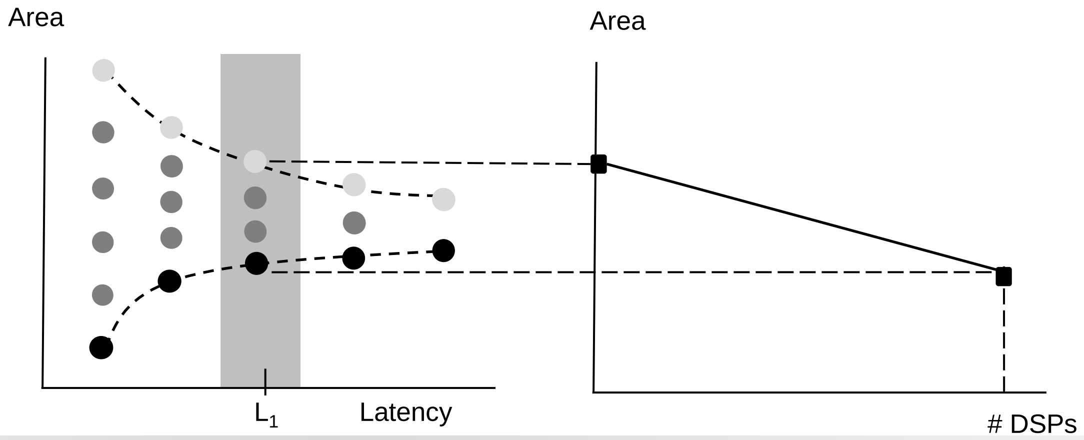
- Use DSP-macro Sensitivity *S* as priority criteria to map MAC operations to DSP-macro or LUTs
- If latency range is given use the Design Family (DF) with design with smallest area
- *S* is computed for the given latency

  Δ Area= Area max - Area min;

  Δ DSP= DSPs max - DSPs min;

  S=Δ Area/Δ DSP;

ProcessN (P_N)

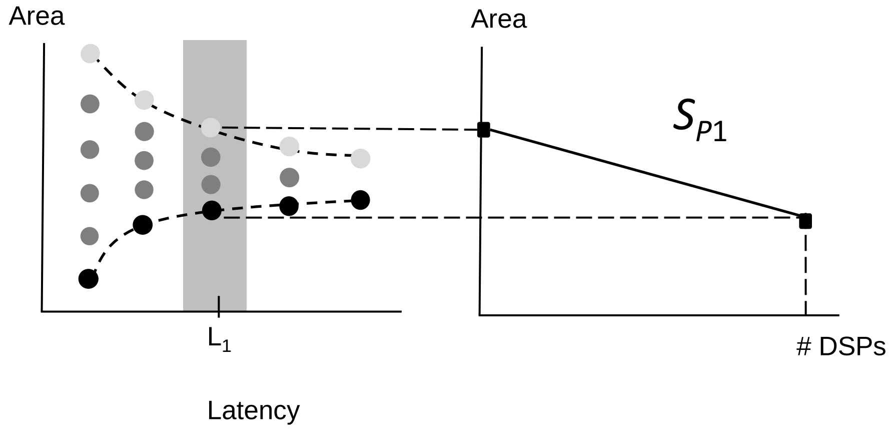


Step 3: Sensitivity Based Process Sorting

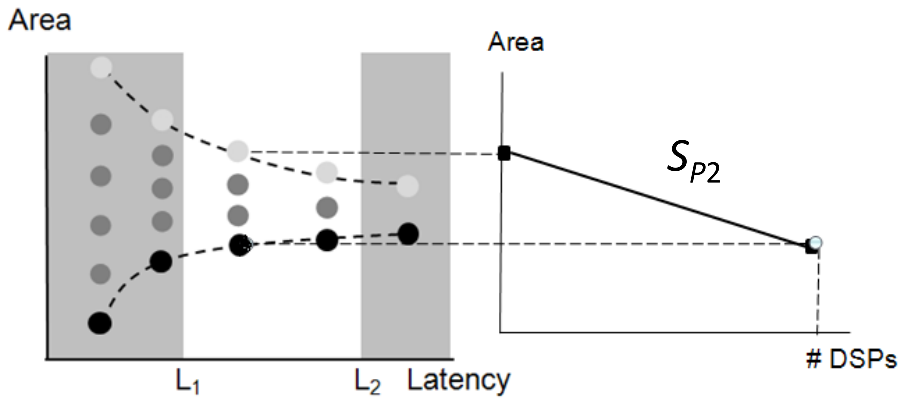
- Sort all the processes in the given system using S as sorting criteria

$$S_{P1} > S_{P2} > S_{Pn}$$

Process1 (P_1)



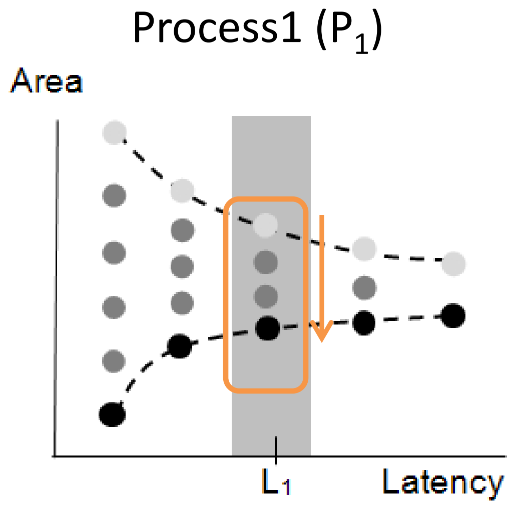
Process2 (P_2)

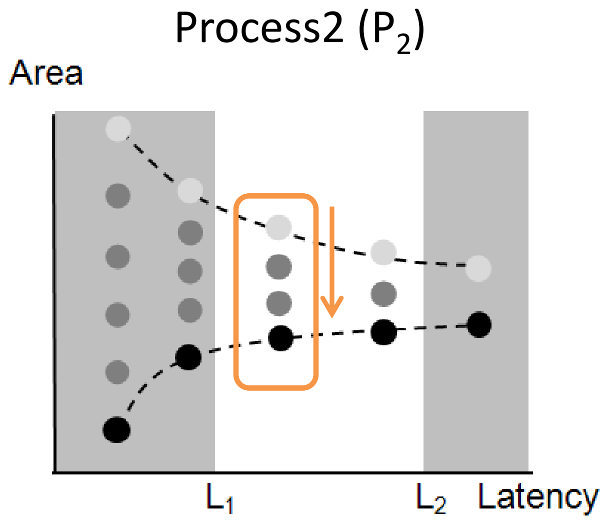


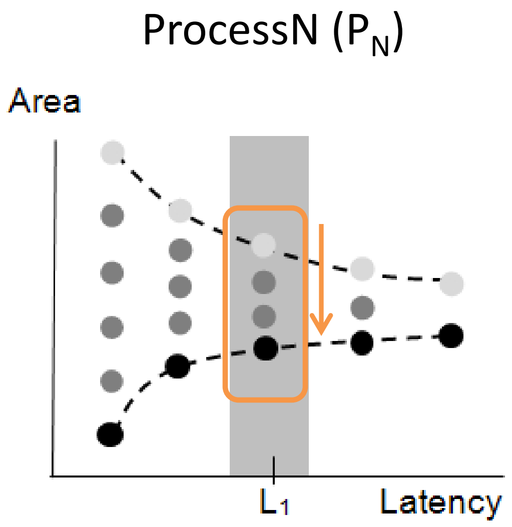
Step 4: DSP-macro allocations

- Greedy DSP-macro allocation process
- Allocated DSP-macros to process P_i with highest Sensitivity S_i until no more DSP-macros are needed OR the DSP-macro budget is exhausted

  $$With\ S_{P1} > S_{P2} > S_{Pn}$$



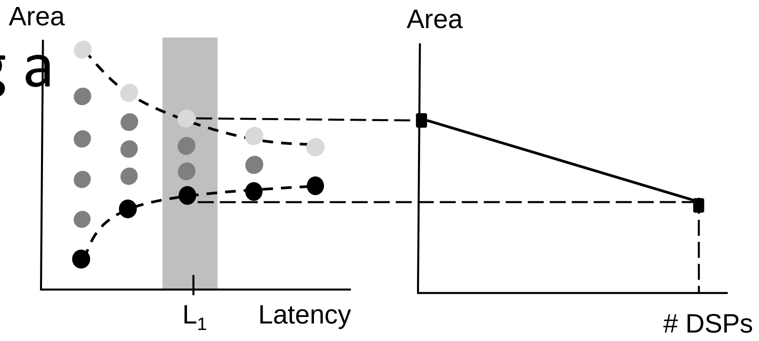




ADSP_MULTP Variation

Method Weakness

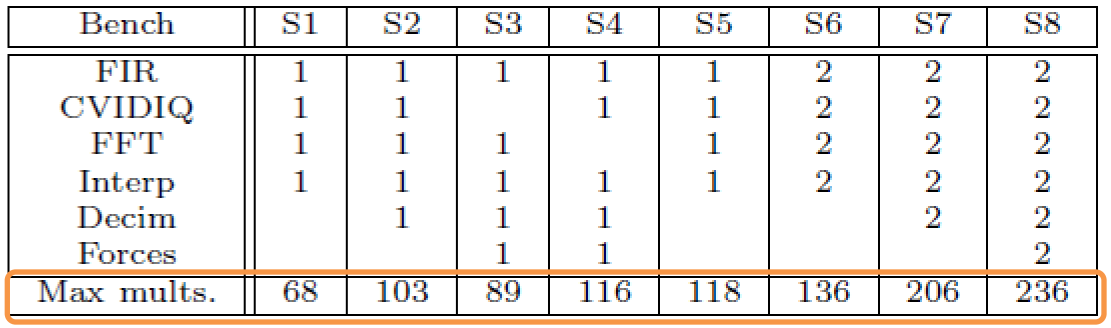
- Assumes that the effect of mapping a MAC onto a DSP-macro is linear within the same design family.
- Size of mapped muxes grows in a none-linear way and hence the sensitivity *S*



- To better understand the impact of the non-linearity in the sensitivity a variation of our proposed method was implemented ➔ *ADSP_MULTP fast brute.*
- Exact same steps as the original method except step4 ➔performs a brute force search trying all possible DSP-macro assignments within the selected *DFs* only

Experimental Setup

- 6 DSP intensive applications chose and grouped together
- Generate 8 complex benchmarks



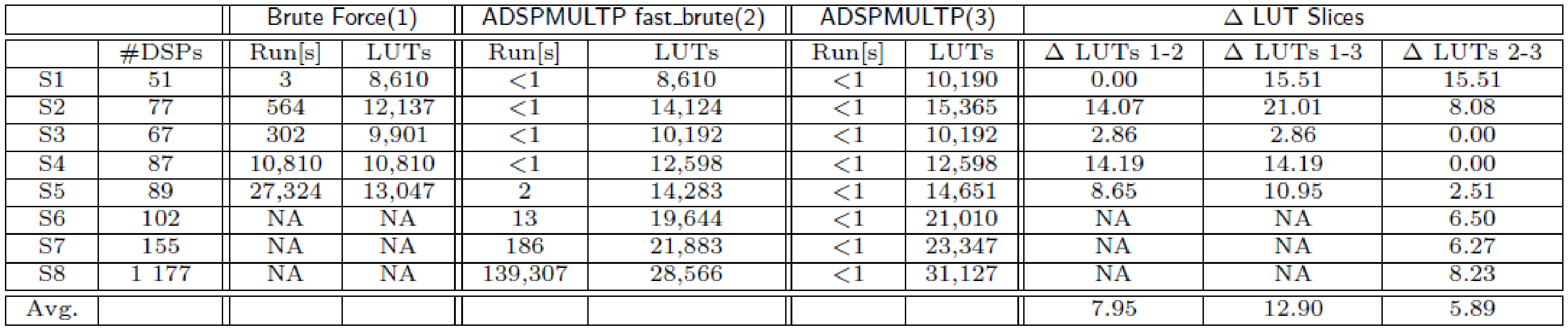
Bench	S1	S2	S3	S4	S5	S6	S7	S8
FIR	1	1	1	1	1	2	2	2
CVIDIQ	1	1		1	1	2	2	2
FFT	1	1	1		1	2	2	2
Interp	1	1	1	1	1	2	2	2
Decim		1	1	1			2	2
Forces			1	1				2
Max mults.	68	103	89	116	118	136	206	236

- The HLS tool used is CyberWorkBench v.5.2 from NEC
- The number of LUTs and registers reported are extracted from Xilinx's ISE 14.2
- FPGA is a Xilinx Virtex 6 VCX130T
- Target HLS frequency is 75MHz

Experimental Results

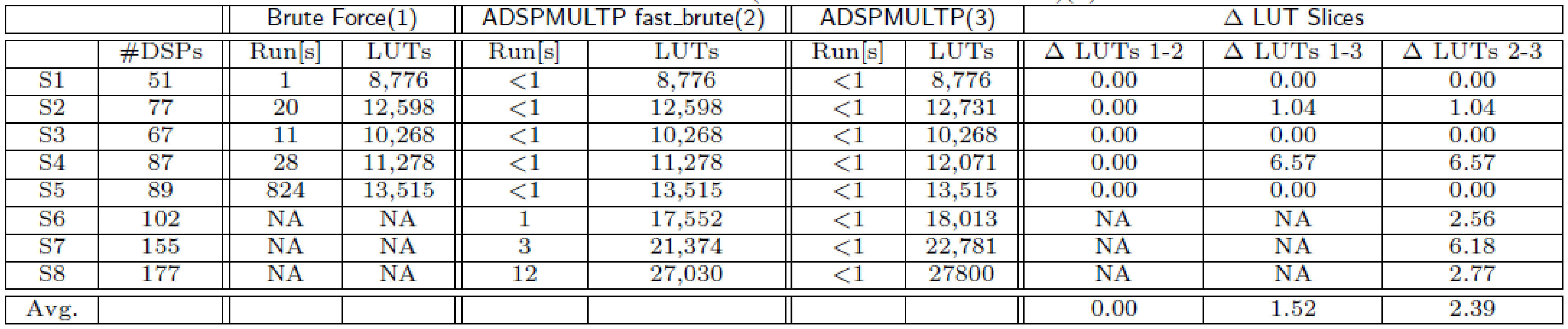
- Brute force vs. ADSP_MULTP fast_brute vs. ADSP_MULTP
- DSP-macro budget is set to 75% of the total number of multiplications that each complex benchmark would need in order to maximize its parallelism
- Brute force running up to 4 days
- Random latency range that covers less than 1/4 of the total latency range was chosen for each of the processes

EXPERIMENTAL RESULTS1 (WITHOUT LATENCY CONSTRAINT) (A)



		Brute Force(1)		ADSPMULTP fast_brute(2)		ADSPMULTP(3)		Δ LUT Slices		
	#DSPs	Run[s]	LUTs	Run[s]	LUTs	Run[s]	LUTs	Δ LUTs 1-2	Δ LUTs 1-3	Δ LUTs 2-3
S1	51	3	8,610	<1	8,610	<1	10,190	0.00	15.51	15.51
S2	77	564	12,137	<1	14,124	<1	15,365	14.07	21.01	8.08
S3	67	302	9,901	<1	10,192	<1	10,192	2.86	2.86	0.00
S4	87	10,810	10,810	<1	12,598	<1	12,598	14.19	14.19	0.00
S5	89	27,324	13,047	2	14,283	<1	14,651	8.65	10.95	2.51
S6	102	NA	NA	13	19,644	<1	21,010	NA	NA	6.50
S7	155	NA	NA	186	21,883	<1	23,347	NA	NA	6.27
S8	1 177	NA	NA	139,307	28,566	<1	31,127	NA	NA	8.23
Avg.								7.95	12.90	5.89

EXPERIMENTAL RESULTS2 (WITH LATENCY CONSTRAINT)(B)



		Brute Force(1)		ADSPMULTP fast_brute(2)		ADSPMULTP(3)		Δ LUT Slices		
	#DSPs	Run[s]	LUTs	Run[s]	LUTs	Run[s]	LUTs	Δ LUTs 1-2	Δ LUTs 1-3	Δ LUTs 2-3
S1	51	1	8,776	<1	8,776	<1	8,776	0.00	0.00	0.00
S2	77	20	12,598	<1	12,598	<1	12,731	0.00	1.04	1.04
S3	67	11	10,268	<1	10,268	<1	10,268	0.00	0.00	0.00
S4	87	28	11,278	<1	11,278	<1	12,071	0.00	6.57	6.57
S5	89	824	13,515	<1	13,515	<1	13,515	0.00	0.00	0.00
S6	102	NA	NA	1	17,552	<1	18,013	NA	NA	2.56
S7	155	NA	NA	3	21,374	<1	22,781	NA	NA	6.18
S8	177	NA	NA	12	27,030	<1	27800	NA	NA	2.77
Avg.								0.00	1.52	2.39

Summary and Conclusions

- Motivated the need to have effective methods to assign DSP-macros to multi-process systems
- Presented a method to allocate FPGA's DSP-macros efficiently across multiple processes synthesized using HLS
- Introduced the concept of sensitivity *S* to allocate DSP-macros across the different processes
- Demonstrated that our method achieves very good results compared to the brute force optimal solutions extremely quick