2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum

Classification of Massively Parallel Computer Architectures

Muhammad Ali Shami
Department of ES
Royal Institute of Technology
16440 Kista, Stockholm
Email: shami@kth.se

Ahmed Hemani
Department of ES
Royal Institute of Technology
16440 Kista, Stockholm
Email: hemani@kth.se

***Abstract*—Faced with slowing performance and energy benefits of technology scaling, VLSI/Computer architectures have turned from parallel to massively parallel machines for personal and embedded applications in the form of multi and many core architectures. Additionally, in the pursuit of finding the sweet spot between engineering and computational efficiency, massively parallel Coarse Grain Reconfigurable Architectures(CRGAs) have been researched. While these articles have been surveyed, they have not been rigorously classified to enable objective differentiation and comparison for performance, area and flexibility. In this paper, we extend the well known Skillicorn taxonomy to create new classes, present a scoring system to rate these classes on flexibility, and present equations for early estimation of area and configuration overheads. Furthermore, we use this extended classification scheme to classify and compare 25 different massively parallel architectures that covers most of the reported CGRAs and other well known multi and many core architectures.**

I. Introduction

Parallelism is becoming mainstream. This is reflected in the widespread adoption of multi-processor architectures that powers desktop PCs, mobile phones, video games, graphics and personal high performance scientific computing that uses general purpose GPUs. The research community is also devoting increased attention to all aspects of research in parallel computing and programming models as can be seen in figure 1. The figure shows the number of publications in different field of parallel computing for last 15 years. It can be seen that the research interest in parallel computing specially in multi-core and reconfigurable computer architectures has increased significantly in the last five years. The seminal survey on Landscape of Parallel Computing[1] from Berkeley attests to this trend in increased interest from the academic world. Massively parallel reconfigurable computing both fine grain FPGAs and Coarse Grain Reconfigurable Architectures(CGRAs) have also been subject of intense research and is finding widespread adoption in industry. These massively parallel reconfigurable computing architectures have also been surveyed by Reiner Hartenstein [2], Raj Krishnamurthy[3] and Max Baron[4].

Survey papers are of great value to the research community, especially new participants, as it provides a single source of information on the most significant research and the challenges in a specific domain. This paper complements the survey papers on reconfigurable computing and the more wider survey paper on the parallel computing landscape[1] by providing a classification scheme that would enable the community to objectively compare and differentiate present and newer parallel computing architectures, including multi/many core architectures, CGRAs and FPGAs.

Flynn[5] classified computer architectures into four categories SISD, SIMD, MISD and MIMD. This taxonomy is perhaps the oldest, simplest and the most widely known. Skillicorn[6] citing the broadness of Flynn's taxonomy as a limitation introduced a new way of classifying the computer architectures in a more comprehensive manner. He used Data Processor (DP), Instruction Processor (IP), Data Memory (DM), Instruction Memory (IM) as the building blocks of a computer architectures and classified them according to the their number (0, 1 or n) and how they are connected one to one or one to many. The limitation in Skillicorn's taxonomy is the granularity of building blocks. Because of higher granularity, these building blocks cannot exchange their roles. Therefore the number of these elements in a computer architecture will remain fixed (0,1, or n). This limitation restricts the application of Skillicorn's taxonomy on modern reconfigurable architecture(FPGA, CGRAs), where the basic blocks are of finer granularity (gates, LUT, CLBs) and can assume the role of either IP, DP or a memory element. So the number of IP, DP or memory elements in these architecture changes upon reconfiguration and is variable denoted by symbol 'v'.

In this paper, we use Skillicorn's taxonomy as the starting point and extend it to cover the richer parallel and reconfigurable computing landscape that has emerged since the paper was published in 1988. We also improve the predictive power of this taxonomy by naming the classes according to some rules. We also define flexibility and give each class a relative flexibility value to compare them against each other. Using our extended classification, we also try to predict the area of the target architecture. Finally We apply the Skillicorn's extended taxonomy on existing parallel and reconfigurable architectures like multi/many core architectures, CGRAs and FPGAs.

Section-II extends the skillicorn's original taxonomy to apply on richer parallel and reconfigurable computing landscape. Section-III talks about the predictive power of this taxonomy. Section-IV applies this taxonomy on modern parallel and reconfigurable architecture and finally Section V gives

978-0-7695-4676-6/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPSW.2012.42

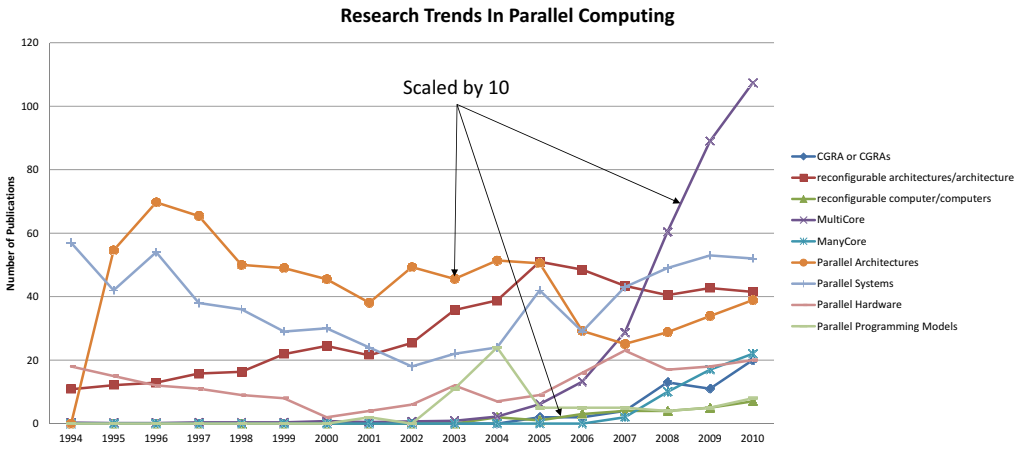


Fig. 1. Research Trends in Computer Architecture (Compiled using IEEE Database)

conclusion.

II. Extension to Skillicorn's Taxonomy

Skillicorn's taxonomy divides a computer architecture into four basic building blocks i.e. a) Instruction Processor (IP), b) Data Processor (DP), c) Instruction Memory (IM) and d) Data Memory (DM). Skillicorn then classify computer architecture on the bases of number of IPs and DP in a system and the interconnection between a)IP-DP, b)IP-IM, c)DP-DM and d)DP-DP. Using this approach, Skillicorn is able to classify uni and parallel processors into classes and have extended Flynn's taxonomy significantly. However there are some problems with this taxonomy and these are:

A. Variable Instruction and Data Processors

In skillicorn's taxonomy, the granularity of the basic building blocks is at the level of Instruction Processor(IP), Data Processor (DP), Data Memory (DM) and Instruction Memory (IM). This implies that the number of IP, DP, IM and DM should be decided at the design time of an architecture. Since these basic building blocks have different properties and cannot exchange their roles, the number of these blocks remains fixed in a computer architecture. This is not true for many reconfigurable computer architectures where the granularity of building blocks is fine (gates) and they can be configured to be either IP or DP of a computer architecture. FPGA is an example, in which the basic building block (Configuration Logic Block) can assume the role of either IP or DP. Such architectures can create both the data-flow or instruction flow machines. To represent such architectures in Skillicorn's taxonomy, we have introduced another element 'v' which means that the number of IPs and DPs in an architecture is variable and can change on reconfiguration of the basic building blocks.

B. IP-IP Connectivity

Skillicorn described an Instruction Processor (IP) as a state machine which determines the next instruction to be executed based on the previous instruction and the state of the Data Processor(DP) receiving the instructions from it. As an example of an IP, Skillicorn used the example of state machine of a Von Neuman Uni-Processor which do not accept any input from neighboring state machines. Therefore he didn't consider inputs from neighboring state machines to play a role in determination of next instruction. Further, he applied this Von Neuman definition of an IP on Parallel/Multi-Processor systems and hence ignored the possibility of IP-IP connectivity in his taxonomy table. We recognize that the system do exists in which an IP (state machine) can be connected to another IP(state machine), or a bigger IP can be divided among two smaller IPs (State machines). Therefore, in the enhanced Skillicorn's table we introduce the IP-IP connectivity column which opens up a number of possible parallel computer architectures (class 13-14 and 31-47) listed in Table-I.

C. Extended Classification

Using the above mentioned improvements in Skillicorn's taxonomy, we created a table with extension to Skillicorn's classification and introduced 19 new classes. The original Skillicorn's taxonomy does contain hierarchy but was not named explicitly. This limited the predictive and explanatory power of the classification scheme as address by Subrata Dasgupta[7]. To address this issue he has tried to produce a hierarchical taxonomy, however made things more complicated by using terminology from chemical engineering. We tried to improve the predictive and explanatory power of extended classification by naming these classes. To do so, we divide a name into a)Machine Type, b) Processing Type, and c) Sub-Processing Type as shown in figure 2. By naming these classes according to the rule described below, we can compare two or more architectures in terms of their similarities or differences.

1) Machine Type (MT): In the naming hierarchy, the type of machine takes the primary branch as shown in figure 2. The type of the machine is defined by the presence or absence

TABLE I
EXTENDED TABLE FROM SKILLICORN'S TAXONOMY

S.N	Gran.	IPs	DPs	IP-IP	IP-DP	IP-IM	DP-DM	DP-DP	Comments
Data Flow Machines−− > Single Processor									
1.	IP/DP	0	1	none	none	none	1-1	none	DUP
Data Flow Machines−− > Multi Processors									
2.	IP/DP	0	n	none	none	none	n-n	none	DMP-I
3.	IP/DP	0	n	none	none	none	n-n	nxn	DMP-II
4.	IP/DP	0	n	none	none	none	nxn	none	DMP-III
5.	IP/DP	0	n	none	none	none	nxn	nxn	DMP-IV
Instruction Flow−− > Single Processor									
6.	IP/DP	1	1	none	1-1	1-1	1-1	none	IUP
Instruction Flow−− > Array Processor									
7.	IP/DP	1	n	none	1-n	1-1	n-n	none	IAP-I
8.	IP/DP	1	n	none	1-n	1-1	n-n	nxn	IAP-II
9.	IP/DP	1	n	none	1-n	1-1	nxn	none	IAP-III
10.	IP/DP	1	n	none	1-n	1-1	nxn	nxn	IAP-IV
11.	IP/DP	n	1	none	n-1	n-n	1-1	none	NI
12.	IP/DP	n	1	none	n-1	nxn	1-1	none	NI
13.	IP/DP	n	1	nxn	n-1	n-n	1-1	none	NI
14.	IP/DP	n	1	nxn	n-1	nxn	1-1	none	NI
Instruction Flow−− > Multi Processor									
15.	IP/DP	n	n	none	n-n	n-n	n-n	none	IMP-I
16.	IP/DP	n	n	none	n-n	n-n	n-n	nxn	IMP-II
17.	IP/DP	n	n	none	n-n	n-n	nxn	none	IMP-III
18.	IP/DP	n	n	none	n-n	n-n	nxn	nxn	IMP-IV
19.	IP/DP	n	n	none	n-n	nxn	n-n	none	IMP-V
20.	IP/DP	n	n	none	n-n	nxn	n-n	nxn	IMP-VI
21.	IP/DP	n	n	none	n-n	nxn	nxn	none	IMP-VII
22.	IP/DP	n	n	none	n-n	nxn	nxn	nxn	IMP-VIII
23.	IP/DP	n	n	none	nxn	n-n	n-n	none	IMP-IX
24.	IP/DP	n	n	none	nxn	n-n	n-n	nxn	IMP-X
25.	IP/DP	n	n	none	nxn	n-n	nxn	none	IMP-XI
26.	IP/DP	n	n	none	nxn	n-n	nxn	nxn	IMP-XII
27.	IP/DP	n	n	none	nxn	nxn	n-n	none	IMP-XIII
28.	IP/DP	n	n	none	nxn	nxn	n-n	nxn	IMP-XIV
29.	IP/DP	n	n	none	nxn	nxn	nxn	none	IMP-XV
30.	IP/DP	n	n	none	nxn	nxn	nxn	nxn	IMP-XVI
31.	IP/DP	n	n	nxn	n-n	n-n	n-n	none	ISP-I
32.	IP/DP	n	n	nxn	n-n	n-n	n-n	nxn	ISP-II
33.	IP/DP	n	n	nxn	n-n	n-n	nxn	none	ISP-III
34.	IP/DP	n	n	nxn	n-n	n-n	nxn	nxn	ISP-IV
35.	IP/DP	n	n	nxn	n-n	nxn	n-n	none	ISP-V
36.	IP/DP	n	n	nxn	n-n	nxn	n-n	nxn	ISP-VI
37.	IP/DP	n	n	nxn	n-n	nxn	nxn	none	ISP-VII
38.	IP/DP	n	n	nxn	n-n	nxn	nxn	nxn	ISP-VIII
39.	IP/DP	n	n	nxn	nxn	n-n	n-n	none	ISP-IX
40.	IP/DP	n	n	nxn	nxn	n-n	n-n	nxn	ISP-X
41.	IP/DP	n	n	nxn	nxn	n-n	nxn	none	ISP-XI
42.	IP/DP	n	n	nxn	nxn	n-n	nxn	nxn	ISP-XII
43.	IP/DP	n	n	nxn	nxn	nxn	n-n	none	ISP-XIII
44.	IP/DP	n	n	nxn	nxn	nxn	n-n	nxn	ISP-XIV
45.	IP/DP	n	n	nxn	nxn	nxn	nxn	none	ISP-XV
46.	IP/DP	n	n	nxn	nxn	nxn	nxn	nxn	ISP-XVI
Universal Flow Machine−− > Spatial Computing									
47.	LUTs	v	v	vxv	vxv	vxv	vxv	vxv	USP

of Instruction Processor (IP). If a machine has an Instruction Processor (IP) connected to the Data Processor (DP) then this machine is an Instruction Flow machine. In such machines, instructions are fetched to decide which data element is to be processed next. Based on the instruction, the machine fetches the data elements from the memory and performs relevant operation on it.

A data flow machine is one in which there is no Instruction Processor. The data elements carry instructions which are then executed on the arrival of the data at the inputs of the processing elements. These instructions may execute out of order, and totally depend on the availability of the data.

Universal Flow machine is one which can implement both Instruction flow or data flow machines. Fine grain architectures

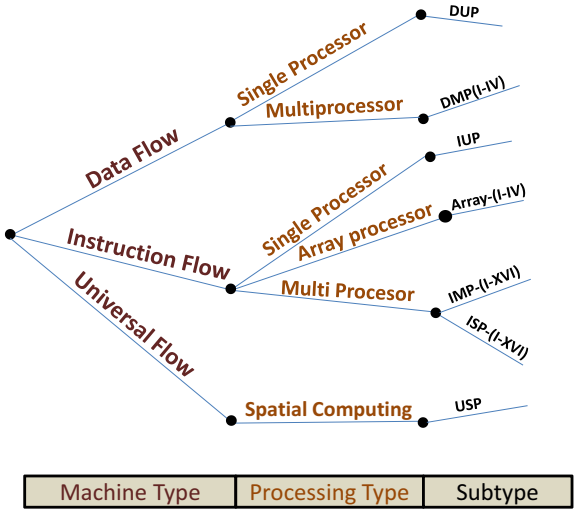
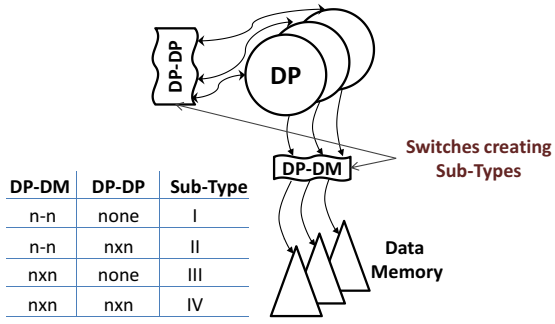


Fig. 2. Hierarchy of Computing Machines

with granularity finer than IP or DP can implement either Instruction Flow or Data Flow machine and are called Universal Flow machine in our taxonomy.

2) Processing Type (PT) and Sub-Processing Type (SPT): Processing Type (PT) depends on number of DPs in data flow or DPs and IPs in Instruction Flow machine respectively. The Sub-Processing type depends on the kind of switches used for IP-DP, DP-DP, DP-DM, IP-IP and IP-IM connectivities. The 'Processing Types' represents the degree of parallelism and the 'Sub-Processing' type represents the flexibility of an architecture. The Types and Sub-types of our classes in Table-I are discussed below according to instruction and data flow paradigm.

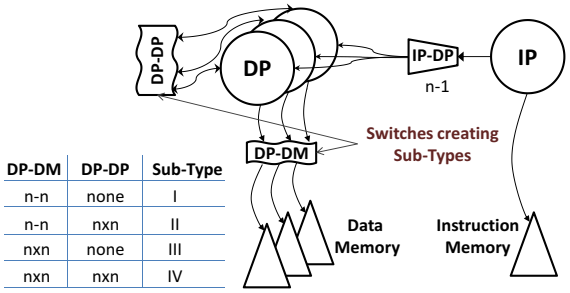
a) Data Flow Paradigm: In Table-I a machine is single processor data flow machine if it has single DP attached to a DM. If the number of DP in a Data Flow Machine is increased to 'n', then system will become a Multi-Processor system. Single Processor system has only one class however, the Multi-Processor system has four sub-classes because of the type of switches which can be used for DP-DM and DP-DP connectivity. These classes are named Data Flow Multi Processors(DMP)(I-IV) as shown in the figure 3 and Table-I.



DP-DM	DP-DP	Sub-Type
n-n	none	I
n-n	nxn	II
nxn	none	III
nxn	nxn	IV

Fig. 3. Skillicorn's Data Flow Machine with Sub-Types defined in this paper

b) Instruction Flow Paradigm: An Instruction Flow Machine is a Uni-Processor if it has one IP and one DP in the system and is donated by IUP (Instruction Flow Uni Processor). However, if a single IP is connected to 'n' DPs, it becomes an Array processor which has four sub-types because of the connectivity of DP with itself or with Data memory (DM) as shown in figure 4. These subtypes are represented by Array Processor(AP)(I-IV). The sub-types represent the flexibility these Array Processors have in them. For instance, in AP-I every DP is directly connected to a DM and this organization cannot be changed. However, in AP-II the DP can be connected to any other DP in addition to a direct connection with a DM. Furthermore in AP-III a DP can be connected to any of the DM in the system. Finally in AP-IV, a DP can be connected to any of the DM or DP in the system. The AP-IV is the most flexible of all the sub-types. The classes from 11-14 in Table-I contains more than one IP connected to single DP which is not possible in a real world system. These classes are practically not implementable (NI) and hence have got no name.



DP-DM	DP-DP	Sub-Type
n-n	none	I
n-n	nxn	II
nxn	none	III
nxn	nxn	IV

Fig. 4. Skillicorn's Array Processor with Sub-Types defined in this paper

A system is a Multi-Processing system if it has more than one Instruction Processor (IP) as shown in figure 5. Classes 15-46 are examples of Multi Processing Systems. In Class 15 there is a direct connection between IP-DP, DP-DM and no connection between DP-DP. This system is least flexible multi processing system and is an example of separate Von Neumann Machines and is named as IMP-I. The flexibility increases with the ability of components to connect other components as can be seen in the Table-I and is referred by sub-classes IMP(I-XVI) with IMP-XVI being the most flexible and IMP-I being the least flexible. Our enhancement to Skillicorn's taxonomy

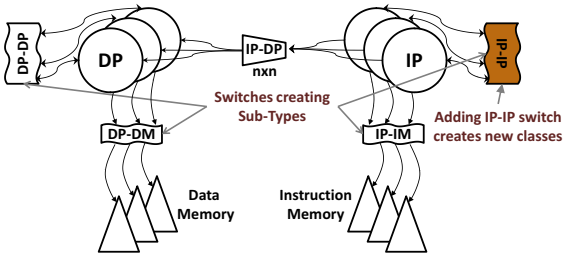


Fig. 5. An Illustration of Instruction Flow Spatial Processors

give rise to the classes from 31-46(also 13-14). These are the classes in which IP can be connected to another IP in a system as shown in figure 5. Those system in which IPs and DPs can be connected together to create a bigger or more complex IP and DP are called Spatial Computing Systems. In such systems the size and dimensions of Data and Instruction Processor can be changed to match the resources needed to run an algorithm. Spatial computing system is super set of all the systems discussed above in instruction flow paradigm. The classes from 31-46 are variants of Multi-Processing systems with the ability of IP to connect to other IPs. Such systems have the ability to create complex computing machines by connecting IPs or DPs together to create a single complex IP or DP. This kind of computing is called Spatial computing and this class of architecture is named as Instruction Flow Spatial Processing (ISP). The sub-classes for ISP have the same property as that of multi-processing and are named ISP(I-XVI)

c) Universal Flow Paradigm: In Universal Flow Machine, all the building blocks have much finer granularity and are of the same kind. They can be connected to any of the other building block and can assume the role of an IP, DP, IM or DM on proper reconfiguration. To allow such flexibility, these building blocks are complemented with rich interconnection network and can be connected to any other building block in the system as shown in figure 6. FPGA is an example of such systems. These systems are most flexible and can create any of the above mentioned computing machine. However this flexibility comes at the cost of reconfiguration overhead in terms of configuration bits and routing resources.

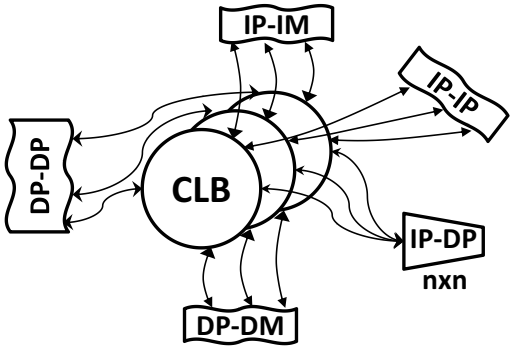


Fig. 6. An Illustration of Universal Flow Spatial Processors

III. Predictive Power of Extended Taxonomy

One of many advantages of a classification scheme is its predictive power. A successful classification scheme can predict the behavior or key characteristics of its classes. Using the extended classification scheme discussed above, we can predict the following

A. Comparison

By just looking at the names of the classes in our extended classification one can compare two or more architectures in

terms of similarities or differences. The first letter represents either this machine belongs to data, instruction or universal flow architecture. The second letter inform us if this machine is a Uni, Array or Multi-Processor. And finally the number at the end tells us about the interconnection between IP-IP, IP-IM, DP-DM and DP-DP. These properties are same for Array Processor or Multi-Processor if they have the same sub-type number at the end i.e. IAP-I and IMP-I will have same IP-IP, IP-IM, DP-DM and DP-DP connectivity. Using this scheme, by looking at the names of the classes we can clearly and easily predict their similarities or differences.

B. Flexibility

The relationship between flexibility and configuration overhead is inversely proportional. An FPGA is most flexible at the cost of enormous reconfiguration overhead while an ASIC is least flexible at no reconfiguration cost. A number of architectures have been proposed which fall between ASIC and FPGA in terms of flexibility and reconfiguration overheads. Some of them claim[8] that they are as flexible as FPGA with performances close to ASIC. Experimental results from the architectures are used to compare their speed or energy efficiency, however there is no metric which one could use to compare computer architectures in terms of flexibility. In this section we have used the properties of the architecture classes discussed above to compare for their flexibility. Flexibility

TABLE II
RELATIVE FLEXIBILITY VALUES FOR DIFFERENT CLASSES

ST	Flx.	ST	Flx.	ST	Flx.	ST	Flx.
Data Flow$--> $ Uni Processor (+0)							
DUP	0	-	-	-	-	-	-
Data Flow$--> $ Multi Processor (+1)							
DMP-I	1	DMP-II	2	DMP-III	2	DMP-IV	3
Instruction $--> $ Uni Processor (+0)							
IUP	0	-	-	-	-	-	-
Instruction Flow$--> $ Array Processor (+1)							
IAP-I	1	IAP-II	2	IAP-III	2	IAP-IV	3
Instruction Flow$--> $ Multi Processor (+2)							
IMP-I	2	IMP-II	3	IMP-III	3	IMP-IV	4
IMP-V	3	IMP-VI	4	IMP-VII	4	IMP-VIII	5
IMP-IX	3	IMP-X	4	IMP-XI	4	IMP-XII	5
IMP-XIII	4	IMP-XIV	5	IMP-XV	5	IMP-XVI	6
ISP-I	3	ISP-II	4	ISP-III	4	ISP-IV	5
ISP-V	4	ISP-VI	5	ISP-VII	5	ISP-VIII	6
ISP-IX	4	ISP-X	5	ISP-XI	5	ISP-XII	6
ISP-XIII	5	ISP-XIV	6	ISP-XV	6	ISP-XVI	7
Universal Flow$--> $ Fine Grained(+3)							
USP	8	-	-	-	-	-	-

arises by having the ability to change the organization of an architecture. An architecture in which the connectivity of the components cannot be changed, is not at all flexible. In our extended taxonomy table, we can see that IMP-II is more flexible than IMP-I simply because in IMP-II DP-DP switch is of type 'x'. However we can also observe that IMP-I is more flexible than IAP-I. This is because of the fact that IMP-I can act as an array processor if all the processors are executing the same program. However, IAP-I cannot be an IMP-I since IAP-I cannot execute 'n' different programs at the same time. Furthermore we also observe that IAP-I is more flexible than IUP because IAP-I can act as a uni-processor by turning off its extra DPs. However, IUP cannot act as an IAP-I simply because it doesn't have enough DPs. This discussion shows that the flexibility in our table arises because of two facts, a) number of IPs or DPs in the system and b) number of switches of type 'x' in the system.

To Calculate flexibility values for all the classes discussed above, we have created a scoring system. According to our system, the presence of 'n' IPs or DPs each will get 1 point. Furthermore, presence of every switch of type 'x' will get another point. Using this scoring system we calculated the flexibility number of the above mentioned classes as shown in the Table-II. The universal flow machines get one extra flexibility number because of 'variable number' of IPs and DPs. We further plot these flexibility numbers on graph as shown in figure.

It is important to clarify here that the flexibility under discussion above is the ability to morph a hardware in different kind of computing machines. A IUP is flexible enough to implement any kind of algorithm with enough instruction storage. However, to exploit data parallelism or instruction parallelism a different kind of computing machine is needed than a IUP. It is this flexibility we are taking about, i.e. ability of a computer architecture to morph into a different computing machine to efficiently calculate a specific algorithm. The flexibility numbers are related to compare these machines. The numbers for data-flow and instruction flow machines are not comparable because these machines cannot replace each other. However their numbers can be compared with the flexibility number of a universal flow machine.

C. Area

Another predictive power of this classification scheme is its ability to predict area. The Area of an architecture is given by Eq.1. In a data flow machine, the first part involving IP and IM will be ignored. By looking at the type of the machine and number of IPs and DPs in a machine, the area of an architecture is easily predicted and is also compared against area of other architecture. The area of an architecture increases by increased flexibility, because the switch of type 'x' takes more area than a switch of type '-'.

$$Area = N * A_{IP} + N * A_{IM} + A_{IP-IP} + A_{IP-IM} + N * A_{DP} + N * A_{DM} + A_{DP-DP} + A_{DP-DM} \quad (1)$$

D. Configuration Overheads

The classification scheme under discussion can also predict the number of configuration bits(CB) required to configure a reconfigurable architecture. The CBs required to configure the individual components are calculated individually. These CBs depends on the type, functionality and IOs of a component and change accordingly. For instance, a full cross bar switch will require more bits than a limited crossbar. The configuration bits for individual components are then substituted in the Eq. 2 to calculate the total configuration bits of a target

reconfigurable architecture.

$$CB = N*CW_{IP}+N*CW_{IM}+CW_{IP-IP}+CW_{IP-IM} + N*CW_{DP}+N*CW_{DM}+CW_{DP-DP}+CW_{DP-DM} \quad (2)$$

IV. Classification

The field of reconfigurable computing was started with the land mark paper of G.Estrin[9]. Since then a number of research and industrial reconfigurable computer architectures have been developed. In this section we applied the classification scheme discussed in Section-II on existing Uni/Multi-Processors and reconfigurable computer architectures. We classified these architectures into classes and listed them in Table-III. The purpose of this classification is to see how these architectures resembles or differ from others. This classification also help predict the flexibility of these architectures. In this Table-III the direct interconnection between two components is represented by '-' and interconnection through a full crossbar is represented by 'x'. The value n is replaced by actual values in the architecture where ever it is possible. For scalable architectures, where the number of IPs and DPs can be changed at design time without modifying the architecture (template based architectures) we continue using value n. The value 'n' represent a constant i.e. in an architecture there are constant number of DP and IP which is 'n'. A DP/IP can be switched off, unconnected or is connected to another DP/IP to create a complex DP/IP, however the DPs and IPs do not change their role in the architectures and their number remains fixed which is n. On the other hand, the value 'v' represent variable number of IPs or DPs i.e. $v >= 0$. This means that in an architecture the role of DP can be changed to IP by configuring it and vice versa. The following table consists of architectures with their architectural properties and their taxonomic names.

The Table-III shows a number of uni, parallel and reconfigurable computer architectures, with their properties and their flexibility number. The ARM7TDMI[10] and AT89C51[11] are the instruction flow uni-processors with relative flexibility value of zero.

IMAGINE[12], MorphoSys[13], REMARC[14], RICA[8], PADDI[15], PACT XPP[16], Chimaera[17] and ADRES[18] are the array processors of Type-II. Each one of these has a host processor controlling the DPs in the reconfigurable fabric. In IMAGINE 6 DPs can be connected to each other or the multi-ported register file through a circuit switched network. MorphoSys has an RC fabric arranged in rows and columns. The RC cells in the fabric can be connected to each other and the frame buffer which is used for storage. REMARC has NANO processors which are also arranged in row and columns and can be connected to each other and the memory. Every NANO processor stores instructions locally but a single global control unit provides the program counter. RICA is an architectural template which is generated for domain specific applications. It consists of Instruction Cells (DPs) loosely coupled to the data memory though I/O ports and tightly coupled to a RISC processor. The PEDDI consists of 8 processor each with data-path (DP) and local control (IP) to control the data-path. These processors are connected to each other and to the I/O bus through a crossbar switch. A global instruction sequencer provides instructions to all the processors in a VLIW fashion. Chimaera consists of a Reconfigurable Array connected to a Shadow Register File. A host processor controls the operations of Reconfigurable Array and the Register File. The Reconfigurable Array in Chimaera is different than others because it consists of FPGA style 2/3 input lookup tables. ADRES template consists of a RISC processor and a reconfigurable fabric of data processors (RC) arranged in rows and columns. The first row of the RC fabric consists of data processors tightly connected to the multi-ported register file. The rest of the RC cells (DPs) in the RC fabric are loosely coupled to the RC cells(DPs) in the first row by a mux-based interconnection network and have no direct connectivity to register file.

MONTIUM[19], GARP[20], Piperech[21], [22] and EGRA[23] are the architectures which belongs to Array Processors of Type-IV. MONTIUM tile consists of 5 data-path units which are connected to 10 memory banks through a full circuit switched network. A sequencer controls the operations of the data-path, interconnects and the memory units in a VLIW fashion by a direct connection between these component and the sequencer. GARP also consists of a MIPS processor tightly connected with a reconfigurable fabric. The reconfigurable fabric in GARP consists of rows and columns. Every row consists of 23 2-bit logic elements. These logic elements are connected together to create a bigger data-path. The logic elements (DP) in GARP are loosely coupled with the memory. Piperench consists of rows of Processing Elements (PEs) which are connected together by a horizontal BUS. Every row has N number of elements. In addition to horizontal BUSes, Piperench also have Vertical BUSes to connect these PEs together. These PEs are controlled by a single Input Controller and are connected to an input/output fifo. EGRA is an architectural template which consists of ALU, Multiplier and Memory blocks placed in rows and columns. The exact placement and the number of each component depends on the target application. These components are connected together by nearest neighbor, vertical and horizontal BUSes. An external control is used to control a Reconfigurable ALU cluster which comprises of these ALU, Multiplier and Memory blocks.

CortexA9[26], Intel Core2Duo[27] and PEDDI-2[25] fall in the category of IMP-I. PEDDI-2[25] consists of 48 processing elements, each with its own local control unit, connected together by a hierarchal interconnection network. The data processing (DP) elements are tightly coupled to the local control (IP) and to the local data memory (DP). Intel Core2Duo has two while CortexA9 has four IPs directly connected to 4 DPs (two and four processors) working in parallel.

Pleiades[28] is an IMP-II architecture which consists of a host processor connected to satellite processors. The satellites processors are connected together through a circuit switched

TABLE III
SURVEY OF MODERN PARALLEL AND RECONFIGURABLE ARCHITECTURES

Architectures	IPs	DPs	IP-IP	IP-DP	IP-IM	DP-DM	DP-DP	Name	Flexibility
ARM7TDMI[10]	1	1	none	1-1	1-1	1-1	none	IUP	0
AT89C51[11]	1	1	none	1-1	1-1	1-1	none	IUP	0
IMAGINE[12]	1	6	none	1-6	1-1	6-1	6x6	IAP-II	2
MorphoSys[13]	1	64	none	1-64	1-1	64-1	64x64	IAP-II	2
REMARC[14]	1	64	none	1-64	1-1	64-1	64x64	IAP-II	2
RICA[8]	1	n	none	1-n	1-1	n-1	nxn	IAP-II	2
PADDI[15]	1	8	none	1-8	1-8	8-1	8x8	IAP-II	2
Pact XPP[16]	n	n	none	n-n	n-n	n-n	nxn	IMP-II	2
Chimaera[17]	1	n	none	1-n	1-1	n-1	nxn	IAP-II	2
ADRES[18]	1	64	none	1-64	1-1	8-1	64x64	IAP-II	2
Montium[19]	1	5	none	1-5	1-1	5x10	5x5	IAP-IV	3
GARP[20]	1	24xn	none	1-24n	1-1	24nx1	24nx24n	IAP-IV	3
Piperench[21], [22]	1	n	none	1-n	1-1	nx1	nxn	IAP-IV	3
EGRA[23]	1	n	none	1-n	1-1	nxn	nxn	IAP-IV	3
ELM processor[24]	1	2	none	1-2	1-1	2x2	2x2	IAP-IV	3
PADDI-2[25]	48	48	none	48-48	48-48	48-48	48-48	IMP-I	2
Cortex-A9(Quad core)[26]	4	4	none	4-4	4-4	4-4	none	IMP-I	2
Core2Duo[27]	2	2	none	2-2	2-2	2-2	none	IMP-I	2
Pleiades[28]	n	n	none	n-n	n-n	n-1	nxn	IMP-II	3
RaPiD[29]	n	m	none	nxm	nxn	m-1	mxm	IMP-XIV	5
Redefine[30]	0	64	none	none	none	22x1	64x64	DMP-IV	3
Colt[31]	0	16	none	none	none	16x6	16x16	DMP-IV	3
DRRA[32]	n	n	nx14	n-n	n-n	nx14	nx14	ISP-IV	5
Matrix[33]	n	n	nxn	nxn	nxn	nxn	nxn	ISP-XVI	7
FPGA[34]	v	v	vxv	vxv	vxv	vxv	vxv	USP	8

network. RaPiD is an IMP-XIV architecture with functional units arranged in a row and connected together by a bus based interconnection network. These functional unit are loosely connected to the memory and are also loosely connected to the instruction processors. In RaPid[29], the instruction processors are connected to the functional units using the same kind bus network which is used for data connectivity. The disadvantage of this approach is that the buses are not scalable and so is the RaPiD as a reconfigurable architecture.

REDEFINE[30] is a static dataflow architecture which executes HyperOP (Coarse Grain Operations) on a reconfigurable fabric. The reconfigurable fabric consists of 8x8 matrix of computational elements (CE) connected together by a packet switched NoC. Every CE contains an ALU (DP), a router which connects the CEs and storage for operations and data. The HyperOps are formed by provided compute and transport meta data by a run-time reconfiguration unit. HyperOps are the sub-graphs in a application's data flow graph. Colt[31] is also data flow architecture architecture having an RC fabric without any instruction processor. The RC fabric consists of a matrix of 4x4 data processing elements, connected together by a crossbar. The data stream consists of routing information and reconfigure colt at run-time. Colt doesn't have any memory but the 6 I/O can be connected to memories.

MATRIX[33], DRRA[32] are the architecture which fall in the category of Instruction Flow Spatial Processors. In MATRIX, every element can be configured to act either as data or instruction storage, Register File or data-path resource. Each element can communicate with other elements using nearest neighbor connectivity, length four bypass and global buses. Although matrix can have variable number of IPs and DPs, it cannot implement data flow architectures and hence fall in ISP-XVI. DRRA is a template which consists of distributed control, memory and data path resources. Every element in DRRA can communicate with every other element at 3-hops on right or 3-hops on left directions. The data-path and memory resources are loosely coupled and can be connected to any of the data path or memory within their window of connectivity. The control element is tightly connected to local data-path and memory resources, and can communicate with other control resources within 3-hop connectivity.

FPGA[34] is the architecture that falls in universal spatial processor category because FPGAs can implement data and instruction flow architectures. FPGAs consists of a number of configuration logic block (CLBs) which can be used in implementation of IPs or DPs. These CLBs can be connected to any other CLB in an FPGA. One can decide at build time that how many IPs or DPs and what kind of connectivity between them is required based on the target algorithm. the DPs in FPGA are loosely coupled to memory elements, and DPs are also loosely coupled to the IPs. The IPs can also be connected together to create a complex IPs. Such architectures allow us to customize the dimensions of data-path by allowing us to change its width, depth and in case of FPGA the bitwidth.

Using the flexibility values, we can compare the above mentioned architectures as shown in the figure-7. It is important to mention that these values are relative values and the values of Instruction Flow and Data flow architectures are not comparable because these architectures cannot substitute each other. However, these values can be compared against a universal flow architecture. The figure shows that the FPGA has the highest flexibility. Matrix and DRRA comes second and third respectably with respect to the flexibility offered by these architectures.

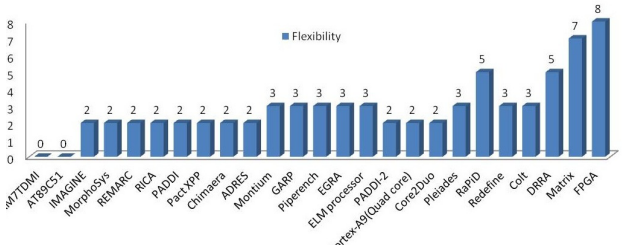


Fig. 7. Comparison of Published Architectures w.r.t their Relative flexibility.

V. Conclusions

This paper uses the Skillicorn's taxonomy and extends it to apply on reconfigurable computer architectures. These improvements include addition of IP-IP switch, representation of variable number of IP and DP to represent architectures like FPGAs. To improve the descriptive power of the resultant taxonomy, we presented a hierarchal naming scheme which helps distinguish different classes in the taxonomy. The paper also presented a scoring system to compare these architectures based on their flexibility and presented equations to predict the area and reconfiguration overhead of a specific class. This work is significant to compare already developed architectures with respect to flexibility. In addition to that, this work is also significant for the design of new computer architectures. By looking into this taxonomy, a designer can decide which computer class offers the required flexibility with minimum configuration overhead for single or set of target applications. Initial estimates of area and configuration overhead gives a designer option to take better design decision earlier during the design life cycle of reconfigurable computer architecture.

References

[1] Bryan Christopher Catanzaro Joseph James Gebis Parry Husbands Kurt Keutzer David A. Patterson William Lester Plishker John Shalf Samuel Webb Williams Katherine A. Yelick Krste Asanovic, Ras Bodik, "The landscape of parallel computing research: A view from berkeley", Tech. Rep. UCB/EECS-2006-183, Electrical Engineering and Computer Sciences University of California at Berkeley, 2006.

[2] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective", in *Proc. Conf Design, Automation and Test in Europe and Exhibition 2001*, 2001, pp. 642–649.

[3] Raj Krishnamurthy, "A survey of next generation reconfigurable architectures for embedded computing", Tech. Rep., College of Computing, Georgia Institute of Technology, 2001.

[4] M. Baron, "Trends in the use of re-configurable platforms", in *Proc. 41st Design Automation Conf*, 2004.

[5] M. J. Flynn, "Very high-speed computing systems", vol. 54, no. 12, pp. 1901–1909, 1966.

[6] D. B. Skillicorn, "A taxonomy for computer architectures", *Computer* vol. 21, no. 11, pp. 46–57, 1988.

[7] S. Dasgupta, "A hierarchical taxonomic system for computer architectures", *Computer*, vol. 23, no. 3, pp. 64–74, 1990.

[8] S. Khawam, I. Nousias, M. Milward, Ying Yi, M. Muir, and T. Arslan, "The reconfigurable instruction cell array", vol. 16, no. 1, pp. 75–85, 2008.

[9] G. Estrin, "Organization of computer systems - the plus variable structure computers fixed", in *Proc. Western Joint Comput. Conf*, 1960, pp. 33–40.

[10] Texas Instruments, *TMS470R1A256 16/32-Bit RISC Flash Microcontroller*.

[11] Atmel, *8-bit Microcontroller with 4K Bytes Flash: AT89C51*.

[12] U. J. Kapasi, W. J. Dally, S. Rixner, J. D. Owens, and B. Khailany, "The imagine stream processor", in *Proc. IEEE Int Computer Design: VLSI in Computers and Processors Conf*, 2002, pp. 282–288.

[13] Guangming Lu, H. Singh, Ming-Hau Lee, N. Bagherzadeh, F. J. Kurdahi, E. M. C. Filho, and V. Castro-Alves, "The morphosys dynamically reconfigurable system-on-chip", in *Proc. First NASA/DoD Workshop Evolvable Hardware*, 1999, pp. 152–160.

[14] Takashi Miyamori and Kunle Olukotun, "Remarc: Reconfigurable multimedia array coprocessor", in *IEICE Transactions on Information and Systems E82-D*, 1998, pp. 389–397.

[15] D.C. Chen and J.M. Rabaey, "A reconfigurable multiprocessor ic for rapid prototyping of algorithmic-specific high-speed dsp data paths", *Solid-State Circuits, IEEE Journal of*, vol. 27, no. 12, pp. 1895 –1904, dec 1992.

[16] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt, "Pact xppa self-reconfigurable data processing architecture", *J. Supercomput.*, vol. 26, pp. 167–184, September 2003.

[17] S. Hauck, T. W. Fry, M. M. Hosler, and J. P. Kao, "The chimaera reconfigurable functional unit", vol. 12, no. 2, pp. 206–217, 2004.

[18] Z. Kwok and S. J. E. Wilton, "Register file architecture optimization in a coarse-grained reconfigurable architecture", in *Proc. 13th Annual IEEE Symp. Field-Programmable Custom Computing Machines FCCM 2005*, 2005, pp. 35–44.

[19] P. M. Heysters, *Coarse-Grained Reconfigurable Processors - Flexibility meets Efficiency*, PhD thesis, Univ. of Twente, Enschede, September 2004.

[20] T. J. Callahan, J. R. Hauser, and J. Wawrzynek, "The garp architecture and c compiler", *Computer*, vol. 33, no. 4, pp. 62–69, 2000.

[21] R. Laufer, R. R. Taylor, and H. Schmit, "Pci-piperench and the swordapi: a system for stream-based reconfigurable computing", in *Proc. Seventh Annual IEEE Symp. Field-Programmable Custom Computing Machines FCCM '99*, 1999, pp. 200–208.

[22] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, and R. Laufer, "Piperench: a coprocessor for streaming multimedia acceleration", in *Proc. 26th Int Computer Architecture Symp*, 1999, pp. 28–39.

[23] G. Ansaloni, P. Bonzini, and L. Pozzi, "Egra: A coarse grained reconfigurable architectural template", vol. 19, no. 6, pp. 1062–1074, 2011.

[24] James Balfour, William Dally, David Black-Schaffer, Vishal Parikh, and JongSoo Park, "An energy-efficient processor architecture for embedded systems", *IEEE Comput. Archit. Lett.*, vol. 7, pp. 29–32, January 2008.

[25] A.K. Yeung and J.M. Rabaey, "A 2.4 gops data-driven reconfigurable multiprocessor ic for dsp", in *Solid-State Circuits Conference, 1995. Digest of Technical Papers. 42nd ISSCC, 1995 IEEE International*, feb 1995, pp. 108 –109, 346.

[26] ARM, *White Paper: The ARM Cortex-A9 Processors*, Sept 2009.

[27] Intel Corporation, "Intel core2 duo processor and mobile intel 4 series express chipset family development kit", 2008.

[28] J. M. Rabaey, A. Abnous, Y. Ichikawa, K. Seno, and M. Wan, "Heterogeneous reconfigurable systems", in *Proc. IEEE Workshop Signal Processing Systems SIPS 97 - Design and Implementation*, 1997, pp. 24–34.

[29] Darren Cronquist, Darren C. Cronquist, Paul Franklin, Chris Fisher, Miguel Figueroa, and Carl Ebeling, "Architecture design of reconfigurable pipelined datapaths", in *In Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, 1999, pp. 23–40.

[30] Mythri Alle, Keshavan Varadarajan, Alexander Fell, Ramesh Reddy C., Nimmy Joseph, Saptarsi Das, Prasenjit Biswas, Jugantor Chetia, Adarsh Rao, S. K. Nandy, and Ranjani Narayan, "Redefine: Runtime reconfigurable polymorphic asic", *ACM Trans. Embed. Comput. Syst.*, vol. 9, pp. 11:1–11:48, October 2009.

[31] Ray Bittner Jr, Peter M. Athanas, and Mark D. Musgrove, "Colt: An experiment in wormhole run-time reconfiguration", in *in High-Speed Computing, Digital Signal Processing, and Filtering Using Reconfigurable Logic. SPIE*, 1996, pp. 187–194.

[32] M. A. Shami and A. Hemani, "Control scheme for a cgra", in *Proc. 22nd Int Computer Architecture and High Performance Computing (SBAC-PAD) Symp*, 2010, pp. 17–24.

[33] E. Mirsky and A. DeHon, "Matrix: a reconfigurable computing architecture with configurable instruction distribution and deployable resources", in *Proc. IEEE Symp. FPGAs for Custom Computing Machines*, 1996, pp. 157–166.

[34] altera, "http://www.altera.com/".