DATA PACKET PROCESSOR (DPP) CORE

Manuel R. Muro, Jr. (Digital Data Innovations (DDI), Beaverton, Oregon, USA; manny@ddi-ip.com)

ABSTRACT

This paper discusses a new approach to packet processing with a unique hardware architecture that provides configurability combined with the performance achieved by dedicated hardware. This architecture can generically be referred to as a Data Packet Processor (DPP). In the same manner that a DSP engine focuses on the processing of signals or a GPU (Graphics Processor Unit) focuses on graphics processing, the DPP specializes on the domain of packet processing. By focus on the packet processing domain, the DPP can more efficiently carry out packet processing operations than could be realized by a general purpose microprocessor. At the same time the DPP architecture is protocol agnostic to the point of enabling a single design block to implement multiple packet protocols in hardware, at line speeds. The DDP can also be used to develop fully customized packet protocols that can be changed in the field, especially in applications where the security of a proprietary protocol has been compromised.

1. INTRODUCTION

In the more traditional sense of Software Defined Radios (SDR) is to use software along with DSPs to replace traditional analog design blocks such as: mixers, filters, amplifiers, modulators/demodulators, detectors, etc. [1]. Such analog circuits exhibited temperature and manufacturing variation issues along with the inability to easily change the functionality of such circuits should the system requirements change. Configurable DSPs to the rescue! However, this is just part of the SDR system, the part that handles how informational bits are represented over the communication channel. When the communication protocol's complexity and data rates are low, a general purpose processor can handle the workload of converting the informational bits into larger informational quantities such as bytes and words, which then represents data packets per a specific digital communication protocol such as the WiFi™ standard. As the data rate and complexity of communication protocols increase with the ever increasing demand for more data bandwidth and overall performance (i.e. power and speed), the use of general purpose microprocessor architectures will no longer fill the demand for future communication system requirements. One solution is to use specialized hardware acceleration in conjunction with a general purpose microprocessor which effectively causes a break with the overall philosophy of SDR. Should the communication protocol need to change, the hardware acceleration logic would need to change. A change that would only be practical if the hardware acceleration was done in a FPGA, but not an option if done in an ASIC. But even with the FPGA approach timing may no longer be met with the protocol change or may become resource limited and require a larger more costly FPGA. Just as the DSP architecture helped to enable flexibility to solve the baseband processing challenges of signal processing the architecture described in the remaining sections will define a new processor architecture, generically referred to as the Data Packet Processor (DPP), which will focus on the challenges of packet processing with a solution that enables changes to be made more freely to the overall digital communication protocol's packet format even if the DPP design was implemented in an ASIC. Aside from enabling flexibility to address the challenges of packet processing, the DDP will enable faster exploration of current protocol standard changes and enable proprietary protocols to be more easily deployed for niche markets or in cases where changes are needed for security reasons.

2. WHAT CONSTITUTES THE DPP

The DSP architecture diverged from the traditional microprocessor architectures, be they CISC/RISC or VonNeumann/Harvard, by adding extra data access busses (X & Y) to read the data samples and the coefficients, along with the instruction word, in a single clock cycle while adding the key hardware logic to support a single cycle multiply with accumulation in order to implement the traditional DSP algorithms, i.e. FFT, FIR, IIR, etc. The DSP architecture typically also employs special address generation logic to handle bit-reversed and modulo addressing, tasks that would be too messy and time consuming to be done in software. The manners in which the DPP diverges from the traditional microprocessor are far more numerous due to the additional levels of complexities of the workload associated with packet processing.

An overall objective for stating what constitutes the DPP is to define an architecture that will enable "operations" to move from the software realm back into hardware realm while still being flexible enough to be easily updated at a later stage in the product development cycle. In particularly,

"operations" which involve tight timing related constraints or complexities that are best off-loaded from the main system processor. Based on the author's experience, the boundary point to determine what is best moved back into hardware and what is to remain in software is that which involves *bit manipulation* and *system level state decisions*. With the conclusion being to keep all bit manipulation tasks in hardware and all system level state decision making tasks with the software on the system's host processor.

2.1 Just One Decision To Make

The DPP's primary objective is to process data packets as fast and as efficiently as possible while providing the flexibility to change how the data packets get processed. Such processing needs to be done at line speeds, where data is coming so fast that there is not time for a pause and there are no rewind options either. In such an environment there is not a lot of time to make many decisions. However, the DPP does have to make one decision and that decision is: *What is the Data Packet Type?* That decision process is really like deciding which train track to send a train down. Once the train passes the switch, the decision cannot be changed.

As a result of only needing to make the determination of what the received data packet type is, i.e. not an issue on the transmit side, the DPP can be defined without the complexities associated with advanced microprocessor concepts such as branch-predicting and the associated "scoreboarding". This is not to say that there are not challenges in making that decision, but making such a decision is at a reduced level of complexity in comparison.

2.2 Bit Granularity

The traditional microprocessors were architected to handle 8-bit quantities, and multiples thereof, with data management down to the bit-level handle by using one, two or three of the base instructions for the *base* data quantity being supported. As a result of that reality, most protocols are defined with this *base* quantity in mind such that if a protocol only needs 1 to 7 bits, or really any number of bits that are not a nice multiple of 8, a full 8 bits, or multiple thereof, would be reserved. In the past, this was a harmless overhead to carry. However, in the present, the carrying of extra "reserved" bits consume both power to transmit and consumes transmission time on the channel, which effectively make them overhead costs from both the power and performance perspectives.
In even an example situation experienced by the author via a change to the WiMedia™ UWB protocol, the payload length field was extended from 12-bits to 16-bits with the new 4-bits placed in "reserved" bits within the packet header and they were not adjacently located to the original 12-bits within the data packet sequence. Such a change to the protocol created "headaches" for both the hardware and software engineers. Keeping in mind that "reserved" bit(s) still consume power and channel time. Such protocol changes should be more manageable from the engineering perspective than they are in practice; hence the vision for the DPP.

So unlike the traditional microprocessor architectures, the DPP needs to be architected in a manner that it can easily handle data quantities with bit-level granularity in order to be truly more efficient than a general purpose microprocessor architecture.

2.3 Dual Control Paths With Shared Resources

Aside from applications which perform either receive (RX) or transmit (TX) only, a bulk of SDR applications involve both RX and TX paths. In particularly, applications that call for reliable links will need both RX & TX even if a device is only a sink or source for information. A key advantage to the DPP architecture is that the overall structures of the data packets are known a priori and follow a common format sequence. As a result a common set of "sequences" and/or "data structures" can be used to either analyze or generate a data packet as defined by a given packet processing protocol. Tasks such as zero-padding needed for the transmit paths would be handled differently by the receive path, but still needs to be "handled". In other words, the same coding sequence used to transmit can be the same code sequence used to receive the data packets for a given protocol.

Then in the context of generating auto-response data packets, the receive path can share packet source and destination information with the transmit path such that the source will become the destination and the destination will become the source. This would occur only if the two control paths have access to a common register set.

2.4 Programmable CRC Modules

Key to making a truly flexible and independent data packet processor is being able to support an appropriate number of CRC modules, as some protocols employ multiple CRC calculations (i.e. separate header and payload CRCs, etc.), but also the CRC module will also need to be flexible enough to support different CRC polynomials to be a "true" SDR system, in order to support more than one communication protocol.

2.5 Security and Other Data Path Support

As the packet's data, be it header or payload data, moves through the DPP system's data paths, the data needs to be

analyzed in a parallel but controlled fashion by different blocks such as Security or Compression engines in tandem with the CRC modules previously discussed, before being sent out to data buffers. Support for such data path modules are key to defining the DPP architecture.

2.6 Data Word And Instruction Word Breakup

Typically with most microprocessor architectures, the size of the instruction word is closely correlated to the processing data word size, but there are certainly exceptions. In particularly, in the "Von-Neumann" type architectures there is a rather close correlation as the system memory can be used for instructions or data. However, as the demands to process more information, and to process information faster, the "bond" between the microprocessor's processing data word and instruction word size are best to be separated. For the DPP, there does not have to be any correlation at all. The size of the processing data word should be flexible with bit-level granularity, TOTALLY independent of the DPP's instruction word size.

The instruction word size is driven by the number of base instructions to be supported and the number of "work" registers supported by the architecture. In the context of packet processing, the processing data word size should be driven by the bit rate for the protocol being supported and the achievable speed that can be obtained using the available technology offerings. As an example, in order to ensure the DPP can support the line rate speeds, assuming that the target technology process is limited to clock rates of 200MHz, in order to support a 10Gbps serial protocol the processing data word size needs to be 50 bits as a minimum. Should the protocol bit rate change to 80Gbps, with the same technology constraints, the processing data word size would need to change to a 400 bits minimum. The DPP architecturally needs to be defined in a manner to easily support such flexibility with allowable changes to the processing data word width, but independent of the instruction word size.

2.7 Flexible Single Cycle Operation(s) And Word Size

At the heart of packet processing is the effective processing and management of fields within the protocol's packet format, in particularly the processing of the packet header fields. A key operation performed by the DPP is the proper extraction of the packet header fields, followed by the proper handling of the payload data. With the processing data word properly sized to the target data rate for the protocol, the DPP will process N bits of data, i.e. the processing data width, from the packet's bit stream in a single clock cycle. Depending on the overall protocol's packet format, the number of fields to be extracted can vary, but at the extreme there will be "N" fields to be extracted. So the DDP's architecture is defined to handle a varying number of field extraction operations with the number of such extractions way less than "N". The number of field extractions supported by the DPP architecture is a key attribute to define the DPP in much the same way that the largest data word size defines the "class" of microprocessors.

Aside from supporting a flexible number of field extractions, by inference from the bit-level granularity requirement the DPP will also need to support variable data word sizes during each of these field extractions. In order to provide execution gains over the general purpose microprocessor architecture, the DPP needs to carry out the processing of a variable number of extractions along with variable data extraction widths in a single clock cycle as the next processing data word will need to be processed in the next clock cycle. The net result is that the DPP can vary both the number and size of data extractions to be performed from one clock cycle to the next.

2.8 Auto-Response Frames

As with most Medium Access Controller (MAC) designs, the DPP would also need to support the generation of auto-response frames with the added challenge of doing it in a "generic" manner to support different communication protocols. With the typical auto-responses being a Clear-To-Send (CTS) in response to a Request-To-Send (RTS) or the automatic acknowledge for a received frame. And more so, the DPP provides the ability to introduce new auto-response frames that were not anticipated in the initial protocol planning, be it a "standard" or proprietary protocol.

2.9 ACK Vector Support

As the pressures to improve overall system performance, newer communication protocols use an ACKnowledge Vector in a single data packet to provide the status some defined number of prior data packets, packets which require an acknowledge in order to determine if a given packet needs to be re-transmitted. In older protocols, the overhead costs of sending a "single bit" of information can greatly impact the effective utilization of a network channel, hence protocols are moving to an ACK Vector approach. The management of such bit manipulation of the vector is less than desirable to done in software, but the requirement to do such work in the DPP architecture remains, while maintaining flexibility.

2.10 Data Packet Timing

Whether the architecture is a general purpose microprocessor or a more specialized architecture such as a DSP or Graphics processor, the concept of time is simply the system processor clock. Ignoring the use of Real-Time-Clock (RTC) modules in a system, the processor clock is the only point of reference to establish a time base. With respect to communication protocols, the base time unit is the protocol's bit time. From this bit time there are other timing constructs that the DPP needs to properly address. Such timings are defined by the protocol and need to be met in order to ensure the reliability of the overall communication network. The management of the data packet timing is traditionally handled by protocol specific hardware timing logic, typically a protocol specific state machine, or is managed in the software by using a system timer (or one built into the microprocessor) which results in interrupts to service these timing intervals. Instead of managing the complexities in software, the DPP contains logic that can more efficiently and more accurately manage such timing events. With the appropriate micro-architecture defined, the timing is managed in hardware while still providing the flexibility that would be found in implementing such timing controls in software. Such timing related management tasks include:

- Is there enough time to transmit another packet?
- Was the expected packet response received in the expected timing window?
- Send packet transmission as soon as possible without violating the protocol's timing rules.

Even when such timing is managed in hardware, it is typically done with a complex state-machine which may or may not be easy to adapt when a protocol change occurs. The handling of such timing related tasks, i.e. non-data rate protocol timing changes, are carried out in a manner to maintain the overall flexibility of the DPP architecture.

2.11 Encapsulation

For protocols that want to provide a path with future growth potential and even some legacy protocols, the DPP architecture would be incomplete without support for encapsulation. Since there is the overall goal of providing flexibility throughout the DPP architecture, the approach to address encapsulation with the DPP is to leave the bit-granularity with the extraction logic but with the advent of wireless protocols like LTE (Long Term Evolution), the support needed for encapsulation is beyond the simplistic view of Type/Length/Values (TLV) constructs. As an example, in the case of LTE, the protocol follows a T-L-T-L-T-L-T-L-V-V-V-V sequence, with the number of such encapsulations being totally variable, instead of the conventional T-L-V-T-L-V-T-L-V-T-L-V sequence.

2.12 Other DPP Options

Aside from low-level requirements mention in prior sections, there are other system level considerations that can be used to define or characterize the DPP architecture. Some possibilities at the "system level" include using a single DPP implementation to:

- Simultaneously process different protocols
- Simultaneously handle multiple data streams
- Auto Protocol Detection
- Efficient data route to minimize system bus transactions

Without getting into the implementation details, the DPP can be implemented to support multiple protocols and/or channels without the DDP's system clock running at the sum of the frequencies needed to support the channels or protocols as if they were implemented by an individual instance of the DPP architecture; with such functionality being vital to effective and efficient implementation of communication protocols that support MIMO (Multi-In/Multi-Out) functionality. Certainly the architecture for the DPP can also be extended to include additional host processor off-load tasks in much the same way that the DPP can off-load tasks like Auto-Response Frames or the management of the bit vector associated with the ACK vector.

3.0 SYSTEM BENEFITS ACHIEVED WITH THE DPP ARCHITECTURE

Aside from the "flexibility" theme repeated throughout the prior sections to define the DPP, the DPP architecture provides benefits that are worth repeating and some which may not be as apparent from the discussion so far.

3.1 Reduced Memory Requirements

Since the DPP is so focused on the specific domain of packet processing the instruction sequences used to carry out the RX or TX processing is identical and can thus be stored in a single dual-ported memory. That would at least halve the memory requirements. However, there is another memory reduction component from the instruction perspective that exists due to the DPP's domain focus on packet processing, such that a **single** DPP instruction can easily replace 5 to 20 general purpose processor instructions to carry out an identical task.

Then from the overall system perspective, especially those that involve more than one communication protocol, the RX and TX data FIFOs along with the Register Files can be "recycled" for use in carrying out the system's other communication protocols, be they a standard and/or proprietary protocol, due to the DPP's protocol agnostic nature.

Aside from reducing the system's areas costs, a secondary benefit is power reduction due to shorter routes and a reduction in the overall system's "idle" memory resources.

3.2 Reduced Resource Requirements

Aside from the shared memory resources, for systems that need to support multiple protocols, the following modules can be "recycled": CRC, Hashing, Security (i.e. DES, AES, etc.) and Compression/Decompression. Since each system typically has a processor along with some hardware acceleration to implement the different protocols, the number of processors in the system can be reduced, i.e. start-up and static power savings can be realized along with area saving! Then from the system level perspective, a single DMA channel can also be shared.

3.3 Efficient Clock Utilization

With the definition outlined in this paper for the DPP architecture, the DPP can more efficiently utilize each system clock when compared to how a general purpose microprocessor clock cycle can be utilized to achieve the amount of work. Aside from the DPP carrying out more work per clock cycle, the DPP can minimize the overall systems bus transactions via a lower need for instruction and data system bus transactions. The reduction in system bus transactions helps to reduce the overall system power consumption, aside from the power savings which is achieved by using a lower clock frequency; hence more efficient clock utilization can be achieved.

3.4 Reduced Software Complexity

In many areas of technology there is a big push to move more and more functionality into software due to the flexibility that doing so provides. However, such a move, from an engineering effort perspective, is certainly from being a free cost. Surprisingly, due to the complexity of some software solutions, things have reached a point where the software royalties have surpassed the hardware royalties. One of the driving forces used to define the DPP architecture was to reduce the software complexity while maintaining the flexibility of software along with achieve packet processing performance levels that were simply unachievable with predominately software based approaches.

4.0 THE DDP'S ROLE IN SDR AND LIMITATIONS

After the baseband processing operations of the SDR, i.e. the digital "radio" operations have completed their signal processing, the DPP can effectively be viewed as being responsible for the processing of the digital data stream of ones and zeroes into manageable quantities of information that define the packets as defined by the appropriate digital communication protocol.

Since the DPP should be viewed a peripheral module in a communication system, the DPP still needs some kind of host processor and ideally an appropriate memory sub-system with DMA and appropriate memory control support. However, this host processor does not need to be a 32 or 64 bit processor instead it can easily be an 8 or 16 bit microprocessor.

Although the DPP provides a great deal of flexibility from the packet processing perspective, there are limits to the DPP's flexibility in much the same way that if a solution initially needed less than 32K bytes of program memory and the system only has 32K bytes of program memory, but a change to the system's firmware is needed and the change now requires 36K bytes, the system's flexibility is "broken". The DPP has similar such constraints with respect to the number of field extractions to be performed or the maximum number of bits that get processed in a single clock cycle. However, the "cost" to add some cushion to the implementation of a DPP based solution is both flexible and manageable. For example to meet the target data rate, let's say only 80 bits needed to be processed per clock cycle to achieve the target data rate, the number of bits could easily be set to say 96 bits or simply to 90 bits due to the bit-granularity requirement. However, such a "cushion" comes with the added benefit of reducing the initial clock frequency for the DPP. The concepts of "cushion" in this context is the same as adding extra gates or flops to system, just in case, but with the difference being that the "extras" added at a higher system conceptual level.

Then there are the standards limitations to be considered, as with any solution, such as latency and providing enough system bus bandwidth to move the volumes of data appropriately.

5.0 DDP'S APPLICATION SPACE

The typical SDR systems involve data rates which are slow enough that a general purpose can keep up. However, as the system performance requirements increase due to the data

rate increases and the need to support MIMO concepts the traditional approaches will no longer suffice.

5.1 High Performance Protocols

Although, the DPP can easily implement protocols such as BlueTooth™ and ZigBee™ the performance of a DPP solution would be overkill for these specific protocols. When the communication system engineer hits a HARD performance wall, a DPP approach should be considered.

5.2 Multiple Protocols

As alluded to in prior sections, another key application space for the DPP architecture is in systems that need to support multiple protocols. Such applications include the bridging of legacy systems to newer systems or simply systems that need to exist in environments that involve multiple protocols pathways, i.e. the modern smart phones support at three wireless protocols with more to be added in the future.

5.3 Proprietary And Emerging Protocols

Another application space for the DPP, which is the original genesis for the DPP vision, is proprietary protocols systems. Although using standard protocols has advantages for most applications, a generalized standard cannot in all practicality solve all communication system needs. There are opportunities or problems that a company may choose to pursue such that a standard protocol may not fill the target product's need, in which case they may need to develop a custom or proprietary communication protocol. There is also the case when a "new" standard is being defined which is not proprietary, but is rather an emerging protocol. Then there is the situation when multiple "standards" for a given problem space is coming out, which does a company bet on? With a DPP architecture approach, such risks can be significantly mitigated if not eliminated altogether.

5.4 Overall Protocol Security

Then there is the aspect of security, be it business or government related, such that the desired solution needs to be flexible enough to change the communication protocol for the entire system when the system is already deployed in the field and the security of the protocol has been compromised. The DPP architecture for packet processing provides the pathway to address this challenge.

As with any new technological concepts there are applications for the DPP architecture that are beyond the imagination of the author.

6.0 SUMMARY AND CONCLUSION

The DPP architecture provides the foundation for a framework to more efficiently address the domain of packet processing in much the same way that a DSP addresses the domain of signal processing or a GPU addresses the domain of graphics processing, while still providing the flexibility to change the manner in which such processing is done. There are many dimensions to defining the DPP architecture, an architecture that provides a new perspective on how packet processing can be performed for achieving both greater efficiency and increased system performance.

The flexibility provided by the DPP architecture enables the development of newer and better communication protocols while still providing a platform to bridge with legacy protocols, thus allowing faster deployment of new communication technologies from the packet processing perspective. The DPP architecture is the final component to defining a truly complete **Software Defined Radio** system solution.

7. REFERENCES

[1] http://en.wikipedia.org/wiki/Software-defined_radio

[2] "Processor Description Languages", Prabhat Mishra & Nikil Dutt, 2008, ISBN: 978-0-12-374287-2

[3] "Network Processors: Architecture, Programming & Implementation", Ran Giladi, 2008, ISBN: 978-0-12-370891-5

[4] "Wireless Communications and Networking", Vijay K. Garg, 2007, ISBN: 978-0-12-373580-5

[5] "TCP/IP Illustrated, Volume1: The Protocols", W. Richard Stevens, 1994, ISBN: 0-201-63346-9

[6] "Digital And Analog Communication systems, 6th Edition", Leon W. Couch, II, ISBN: 0-13-081223-4.

[7] "Computer Architecture: A Quantitative Approach", John L. Hennessy & David A. Patterson, 2003, ISBN: 1-55860-596-7.

[8] "Embedded DSP Processor Design: Application Specific Instruction Set Processors", Dake Liu, ISBN: 978-0-12-374123-3.