P2012: Building an Ecosystem for a Scalable, Modular and high-efficiency Embedded Computing Accelerator

Luca Benini[†], Eric Flamand, Didier Fuin, Diego Melpignano
STMicroelectronics, Grenoble, France
[†]also with DEIS - Università di Bologna, Bologna, Italy
luca.benini@unibo.it, eric.flamand@st.com, didier.fuin@st.com, diego.melpignano@st.com

***Abstract*- P2012 is an area- and power-efficient many-core computing fabric based on multiple globally asynchronous, locally synchronous (GALS) clusters supporting aggressive fine-grained power, reliability and variability management. Clusters feature up to 16 processors and one control processor with independent instruction streams sharing a multi-banked L1 data memory, a multi-channel DMA engine, and specialized hardware for synchronization and scheduling. P2012 achieves extreme area and energy efficiency by supporting domain-specific acceleration at the processor and cluster level through the addition of dedicated HW IPs. P2012 can run standard OpenCL and OpenMP parallel codes well as proprietary Native Programming Model (NPM) SW components that provide the highest level of control on application-to-resource mapping. In Q3 2011 the P2012 SW Development Kit (SDK) has been made available to a community of R&D users; it includes full OpenCL and NPM development environments. The first P2012 SoC prototype in 28nm CMOS will sample in Q4 2012, featuring four clusters and delivering 80GOPS (with single precision floating point support) in 15.2mm^2 with 2W power consumption.**

I. Introduction

The Platform 2012 (P2012) project aims at moving a significant step forward in programmable accelerator architectures for next-generation data-intensive embedded applications such as multi-modal sensor fusion, image understanding, mobile augmented reality. P2012 is an area- and power-efficient many-core computing fabric, and it provides an architectural harness that eases integration of hardwired accelerators. The P2012 computing fabric is highly modular, as it is based on multiple processor clusters implemented with independent power and clock domains, enabling aggressive fine-grained power, reliability and variability management.

Clusters are connected via a high-performance fully-asynchronous network-on-chip (NoC) and feature up to 16 processors with independent instruction streams (MPMD) sharing single-cycle access multi-banked level-1 data memories, a multi-channel DMA engine, and specialized hardware for synchronization and scheduling acceleration. P2012 achieves extreme area and energy efficiency by aggressive exploitation of domain-specific acceleration at the processor and cluster level. Each processor can be specialized at design time with modular extensions (vector units, floating-point units, special-purpose instructions). Clusters can easily become heterogeneous computing engines thanks to the integration of coarse-grained hardware processing elements. Hardware-software interaction is facilitated by the local and global interconnect which efficiently supports point-to-point stream communication. The ultimate goal of P2012 is to fill the area and power efficiency gap between general-purpose embedded CPUs and fully hardwired application accelerators.

From the software viewpoint, P2012 supports different programming models for a wide range of performance and platform portability objectives. Standards-based Programming Models are based on industry standards that can be implemented effectively on the P2012 platform. OpenCL is supported since 2011, and OpenMP support is planned for 2012. The Native Programming Model (NPM) is closely coupled to the platform and provides the highest level of control on application-to-resource mapping, at the expense of abstraction. Advanced Programming Models (typically data flow variations) are tuned to exploit combinations of HW and SW processing elements within a cluster. P2012 programming tools assist the developer from high-level application capture and simulation, to the analysis, debug and visualization of the performance- and power-optimized version of the application mapped onto the fabric, including its interaction with a host processor Programming-model awareness is maintained at all abstraction levels and for all the different tools provided.

The P2012 ecosystem is rapidly shaping up. A complete SDK for OpenCL and NPM is available today for a community with more than thirty partners. The full runtime source is available to promote independent development of innovative software environments. OpenCV bindings have also been prototyped to facilitate acceleration of computer vision and image understanding applications. A variety of virtual platform options are supported for software development, performance analysis and optimization, HW-SW design exploration. An FPGA emulation board will be available by Q2 2012, and the first P2012 SoC will sample in STM 28nm CMOS technology [1] by Q4 with 4 fully populated clusters, each capable of 19GOPS (with full floating point support) in 3.8mm^2 of silicon with a 0.5W power consumption.

II. P2012 PositioninG

The digital IC industry shift toward multi-core platforms is now well under way. There are two main architectures that are driving this evolution: general purpose shared memory multi-cores (GP-SMPs) and programmable graphic units (GP-GPUs). Both families have to strike a reasonable balance between

978-3-9810801-8-6/DATE12/©2012 EDAA

power budget, the so called power wall, and achievable performance. GP-GPUs are the forefront for what concerns the number of cores, with up to thousands of them, strongly factorizing hardware resources, and heavily relying on a data-parallel workload assumption, whereby the same program/instruction is executed on many different data items. Maximum performance is achieved assuming that thousands of small thread, mostly identical, can be exposed. General purpose SMP, being more driven by coarse grain parallelism, limits the core count to less than 10 today, maximum performances is achieved assuming that a few large independent threads can be exposed.

At the same time, driven by the convergence between tethered and mobile devices, we see a rapidly growing demand for a new type of interactions between the user and the device based on the understanding of the environment sensed in multiple manner (image, motion, sound, …) striving to create more friendly user interfaces (augmented reality, virtual reality, haptic, …). The good news is that these are problems exposing a relatively high degree of parallelism; the bad news is that, differently from traditional graphics and image processing, parallel threads usually expose a behavior heavily dependent on the local data content, resulting into many truly independent parallel computations. In this case general purpose coarse grain parallelism cannot deliver and GP-GPU becomes extremely inefficient due to extreme thread divergence. In this gap we are positioning P2012. In addition, we are adding the following design requirements:

- We want the architecture to be easily scalable even from the point of view of design effort, in practice this means a "step and repeat" based approach where the building blocks are highly decoupled .
- We want the solution to be deep sub-micron process friendly; for this, sophisticated adaptive schemes need to be used to reduce the safety margins part of the sign-off process. It is interesting to note that adaptive schemes are also leading to extremely power efficient solutions.
- We want the architecture to be easily programmable and open, sticking as much as we can to existing parallel languages such as OpenCL or OpenMP in order to enable a rich software eco system.
- Finally, due to a the large dispersion of economical sweet spots from market to market we believe it is essential to be able to derive easily and rapidly heterogeneous implementations combining software with hardware accelerated blocks still remaining within the same architectural framework and relying on the same programming model.

Combining all these requirements we designed P2012 as a flexible, parametric and scalable architecture that covers a spectrum going from a pure SMP cluster-based solution down to a software controlled hardware pipeline with all the intermediate points. Fig. 1 summarizes the key characteristics of our solution, with respect to GP-SMP, GP-GPU and hardwired special-function pipeline.

The radar chart combines several normalized figures of merit, namely: peak performance (GOPS), peak performance vs. average performance, performance per power unit (GOPS/W), performance per cost unit (GOPS/mm^2), flexibility. In the next section, we will see that P2012 can be configured as a hardware pipeline with software control (P2012-HWSW), as a homogeneous multi-core cluster (P2012-SW) as well as a full spectrum of HW-SW intermediate solutions.

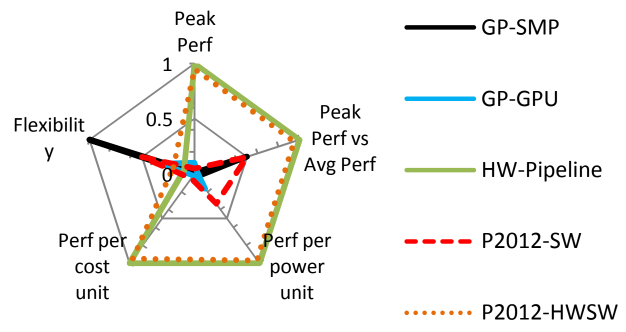

Fig. 1 P2012 positioning

The bottom line of Fig. 1 is that P2012-SW is a competitive architecture for embedded computing, with more architectural flexibility than current GP-GPUs as well as significantly higher power and area efficiency than GP-SMP (and moderately higher efficiency than GP-GPUs). On the other hand, it also offer a viable design platform to build customized HW pipelines with high design productivity. In the reminder of the paper we will focus primarily on P2012-SW, as this is the target of our first silicon implementation. However, will also touch upon the key architectural features supporting hardware specialization.

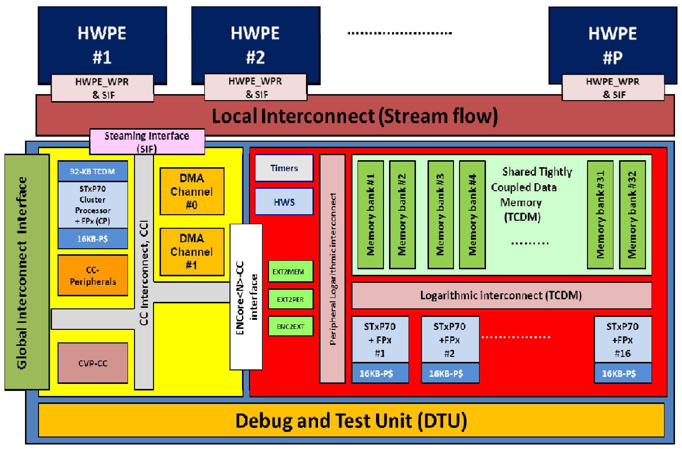

Fig. 2 The P2012 configurable cluster architecture

III. ARCHITECTURE

P2012 can be described as a GALS fabric of tiles, called clusters, connected through an asynchronous global NoC [2] (GANOC). The P2012 cluster, depicted in Fig. 2, aggregates a multi-core computing engine, called ENCore and a cluster controller. The ENCore cluster can host a number of processors (processing elements – PEs) varying from 1 to 16 (the maximal configuration). The extensible processor STxP70-V4 is the PE for the ENCore. It is a cost effective and customizable 32-bit RISC core supported by a comprehensive state-of-the-art development toolset. The STxP70-V4 architecture is a 32-bit load/store architecture with a variable-length instruction-set encoding (16, 32 or 48-bit), allowing manipulation of 32-bit, 16-bit or 8-bit data words and supporting predicated execution to minimize the branch penalty impact. The STxP70-V4 core is implemented with a 7-stage pipeline, and it can execute up to two instructions per clock cycle (dual issue). The P2012 baseline STxP70 core is configured with a 32-entry register file, two zero-cycle overhead hardware loops, a 32-bit hardware multiplier, a second 32-bit core ALU and an interrupt controller with 16 maskable interrupt lines and 1 non-maskable interrupt.

The STxP70-4 architecture offers a simple way to extend the basic STxP70 instruction set with application-specific instructions. An STxP70-4 version with a floating-point unit extension (FPX) is used for the first silicon implementation of P2012. In this configuration, one single ENCore16 cluster can execute up to 16 floating point operations per cycle. The dual-issue architecture of STxP70-4 ensure that this FLOP rate is sustainable as loads, stores and integer operations can be issued in parallel with the FLOPs. In addition, since all cores have independent instruction issue pipelines, there is no single-instruction, multiple-data restriction on execution, which is a common restriction of GPUs. This greatly simplifies application porting and development.

The processing elements have a L1-private program cache (P$, 16KB direct mapped). The ENCore16 processing elements do not have private data caches or memories, therefore avoiding memory coherency overhead. All ENCore processers share a L1, 256-KB (32-bank) tightly coupled data memory (TCDM). The TCDM can therefore support a throughput one data access per PE on each clock cycle. This L1 memory has been architected with a banking factor (number of memory banks divided by the number of processor) of 2 to reduce memory bank conflicts occurring when several PEs attempt to access the same memory bank simultaneously. The performance of most digital systems today is limited by the performance of their interconnection between logic and memory, rather than the performance of the logic or memory itself. The ENCore16 architecture solves this problem by using a high-performance, ultra-low latency network to interconnect its processing elements and on-chip shared memory banks.

ENCore provides scheduling and synchronization acceleration by means of a Hardware Synchronizer (HWS). The HWS includes a synchronization module, an event and an interrupt generator, a dynamic allocator, and a fault-tolerance module. The synchronization module provides a hardware-supported acceleration of various synchronization mechanisms: semaphores, mutexes, barriers, joins, etc. The event and interrupt generators allow any processor to generate an event or interrupt (a PE sees the event as a low-priority interrupt) to any ENCore16 PE. To avoid systematic polling operations, automatic notification is made possible via the programmable notifier which can send trigger events on specific synchronization values. The Interrupt generator also contains a System Messenger providing a set of mail-boxes to generate interrupt notifications to a target PE, whenever a new message is stored. The Dynamic Allocator allows the system to dynamically assign a task to an available PE. It is able to check the resource states (PE fault status, dynamic interrupt priority level table, etc.) and to allocate the best processor to execute the task.

The cluster controller (CC) consists of a cluster processor sub-system, a DMA sub-system, a CC interconnect, and three interfaces: one to ENCore16, one to global fabric-level GANOC and one to the local asynchronous network for plugging in hardwired accelerators. The CC contains a cluster processor designed around a STxP70-V4 dual-issue core with FPX, 16-KB of program cache and 32-KB of local data memory. The cluster processor, in conjunction with its cluster controller peripherals, is in charge of booting and initializing the ENCore16 PE. It also performs application deployment on the ENCore16 PEs, some of the error handling, as well as energy management through the Clock Variability and Power (CVP) module. The CVP exposes process, variability and temperature sensors information and generates clocks. The DMA sub-system is made of 2 independent DMA channels. It performs the data block transfers from the external memory to the internal memory and vice versa while the various PE are operating. The CC interconnect supports intra and inter-cluster communication.

As depicted in Fig. 2, the P2012 cluster can link the ENcore, described above, with a set of hardware accelerators (Hardware Processing Elements, or HWPEs) that provide cost-optimized implementation for those functions for which a software implementation would be too expensive in terms of area and energy (e.g. advanced motion compensation in video encoding). Local, point-to-point communication between HPWEs is provided by an asynchronous local interconnect which ensures complete physical decoupling (clock and power domains) for maximum energy efficiency and robustness. The orchestration of the HWPEs, as well as the synchronization and communication with external memories and other SoC actors are provided by the cluster controller.

The local interconnect (LIC) is based on the asynchronous NoC technology also used at the fabric level. It natively supports flow-based communication, i.e. data flowing between HWPEs are not converted into read and write transactions, but they are managed directly as data packets by the network. This maximizes payload efficiency as addresses do not need to be packetized into flits. Moreover, data exchange and input-output FIFO management at the boundary of each HWPE are natively supported by the network through and end-to-end credit-based protocol. This greatly reduces the latency of control information communication (e.g. queue status), thereby minimizing FIFO sizes. Read-write communication is also supported though read-write wrappers, and this is a convenient abstraction when accessing or updating memory mapped HWPE configuration registers and intra-PE memories.

The DMA engine(s) create the bridge between the "read-write" environment of the external memory hierarchy and the flow-based communication among HWPEs. The read-write (RW) plugs manage transactions to external memories (e.g. to fetch lines from a frame buffer) and the streaming plugs stream the data though the LIC. Therefore, from a data-flow perspective, the cluster controller is the source and the sink of the streaming computation performed by the hardware-accelerated cluster. In addition, the CC processor has fast, local access to the configuration and control ports of HWPUs through the RW wrapper in the LIC: it can therefore run control code to orchestrate and configure IO and HWPE computation to run-time settings. The CC processor ensures a significant level of run-time flexibility for IO management and fine-grained control of HWPU operation modes even in a fully-accelerated cluster where the ENCore cluster does not contain any programmable processors, but only DMA buffering memory.

P2012 provides a modular architectural template to build programmable accelerators: the CC and the LIC create a bridge between the maturing high-level synthesis tools and scalable multi-core architecture. Moreover, our design flow ensures continuity in the cost optimization trajectory. Starting from a software-centric implementation, computationally intensive kernels can be progressively moved into hardware accelerators and the P2012 facilitate the construction of a mixed hardware-software cluster and the automatic generation of control code for orchestrating data flow between hardware and software components.

Power, thermal, variability management are essential features in computing architectures targeting deep-submicron CMOS implementation. P2012 makes use of several hardware-assisted control loops to reduce design-time margin and to improve energy efficiency. Each cluster has a local clock, generated with a small-size and highly reactive frequency-locked-loop. Clock speed can be adjusted in a few cycles on a per-cluster basis with no inter-cluster constraints thanks to GALS architecture. The fabric interconnect is fully asynchronous, hence no global chip-wide clock distribution is required. Static and dynamic variability are managed though a number of distributed sensors, both direct (critical path monitors, both embedded an replica-based) and indirect (thermal sensors, both absolute and relative). Sensor are accessible through memory-mapped registers clustered in the CVP module. Hence it is possible to implement feedback-based software policies for operating points selection. In addition, hardware-based programmable triggers are also available to provide an ultra-fast reaction mechanism to emergency signals coming from the sensors.

IV. Software and Tools

As depicted in Fig. 3, the P2012 software stack provides at the top the programming models used to develop parallel applications, namely OpenCL and the Native Programming Model (NPM). The MIND [3] SW component toolchain helps compiling applicative binaries (to be dynamically downloaded on the cluster by the host), bindings for cluster-local or remote communication as well as parts of the cluster runtime. SW components lifetime is controlled by the Comete CM in the host and operated by the Comete CO in the cluster controller. The other functions of the so-called resident runtime (orange boxes) include the task scheduler and dispatcher, memory allocators, resource and power monitoring, the communication driver with the host and the low-level execution loop in the Encore processors. Different execution engines can be used in the cluster, one example being the Reactive Task Manager that allows easy fork/join and duplication of worker jobs on the available ENCore processors. P2012 HW resources (e.g. DMA for memory transfers) are accessed by the runtime and by application kernels or components through a HW Abstraction Layer (HAL).

Depending on the programming model, the CC can run either NPM application components or a data flow scheduler (to use the HW blocks in the cluster) or an OpenCL runtime. A host OpenCL 1.1 API library can be used to execute kernels on P2012. A host Linux driver allows complete observability and control of P2012 clusters.

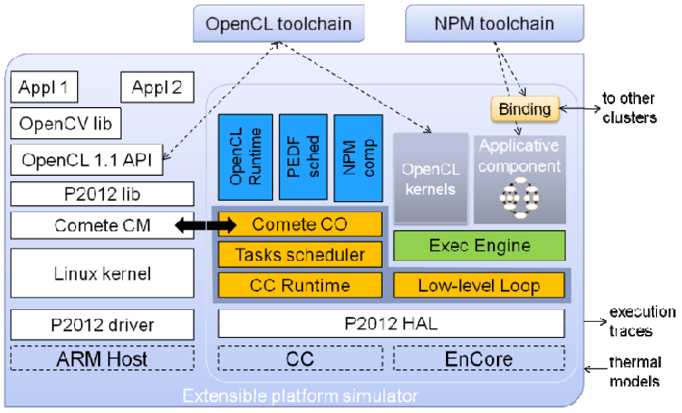

Fig. 3 The P2012 Software Stack

The flexible SW architecture described above makes it easy to integrate new programming models into P2012, in particular, OpenMP support is being added.

The P2012 programming tools provide solutions for multiple levels of the development cycle: from high-level application capture and simulation, using the supported programming models, to the analysis, debug and visualization of the performance- and power-optimized version of the application mapped onto the fabric, leveraging the P2012 runtime. This range of solutions is embodied in a Software Development Kit (SDK), which also includes the plugin-extensible virtual platform models for simulation and execution trace analysis and an Eclipse-based IDE.

From the software viewpoint, the P2012 architecture is PGAS (partitioned global address space). All processors have full visibility all the memories with no aliasing: hence, it is possible for processors in one cluster to load and store directly in remote L1 memories in other clusters. The same holds for L2 and main memory (see next section). Loads are blocking, while stores are posted. A relaxed memory consistency model is hardware-supported through memory barrier instructions. Synchronous and asynchronous DMA-assisted memory copy functions are the preferred way to hide the access latency to remote memories.

From an application viewpoint, P2012 can efficiently accelerate computationally intensive machine vision algorithms. therefore a dedicated version of the well-known OpenCV library is provided.

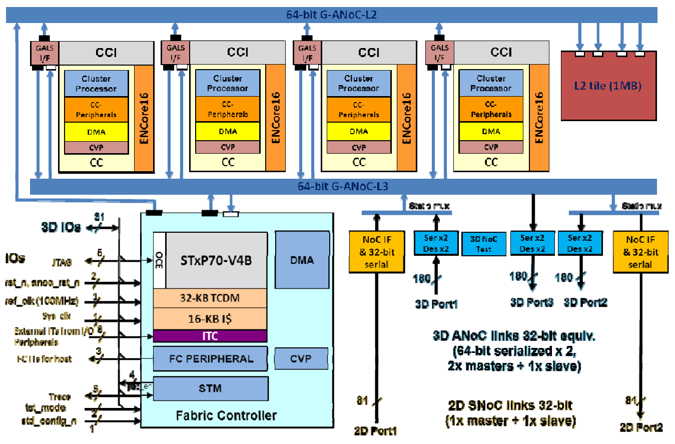

Fig. 4 Block diagram of the Flexible SoC

V. THE P2012 FLEXIBLE SOC

The first silicon embodiment of P2012 is the flexible SoC depicted in Fig. 4. The main objective of the SoC is to demonstrate on silicon software productivity and computational efficiency of a P2012-based computing fabric, rather than minimizing power or area through HW specialization. Hence, the four clusters in the SoC are homogeneous, featuring the maximal configuration ENCore16 with FP support. The SoC also features a 1MB L2 memory for code and data buffering and a fabric controller (FC) for fabric-level runtime and interaction with the off-chip host. The chip is GALS, with 5 clock and power domains (one per cluster, plus one for FC and L2).

One key innovation in the physical implementation of the SoC is its flexibility in off-chip connectivity. The die can be configured as an "accelerator chiplet" for three-dimensional die-stacking by appropriately setting the static MUXes shown on the right hand side of the figure. In this 3D mode (color-coded with light blue in the figure) the fabric interface to host and main memory goes through three wide (180 pins) IO ports (two initiators, one target) driven by micro-buffers and tied to micro-pads for die stacking. The bandwidth supported in this configuration is 3.2GBps. In addition (not shown in the figure), power and ground are also delivered through a "vertical plug". In this configuration the die will be flipped and stacked on top on a host SoC with CPU, peripherals, standard IOs and DRAM interfaces.

A second 2D configuration is supported by the static MUXes. In this mode traditional board-level high-speed interface (color-coded in light orange) will link the fabric with the external host and main memory. This interface is physically driven through a smaller number of standard IO pads (two 81-pin ports). While the 3D interface will feature a higher GBps/mW (energy efficiency) rating than the 2D one, the latter will have a shorter manufacturing cycle, and it won't require a 3D-ready host SoC. The 2D configuration is suitable for low-cost flip-chip packaging and for interfacing the SoC with on-board FPGA-based hosts (e.g. Xilinx Zynq [4] or Altera ArriaV SX [5] devices).

The SoC is being implemented in STMicroelectronics' low-power 28nm CMOS process [1]. Target chip area is below 26mm^2, with 15.2mm^2 dedicated to the 4 clusters. On-going backend trials confirm that clock frequency is between 500 and 600 MHz (depending on PVT corner). The flexible clocking scheme with closed-loop control enables fast adjustment to the operating conditions, thereby greatly reducing conservative guard-banding on clock frequency at run-time. Each cluster features 16 (ENCore 16) +1 (CC) dual issue processors, hence peak GOPS per cluster is between 17 and 20. The power distribution grid of the SoC is designed to handle power delivery in both 3D and 2D configurations. Expected average power consumption per cluster under heavy workload is 0.5W. The chip power grid is designed to sustain up to 4W of power dissipation (at 1.1V, 125C), but its aggressive power management features enables energy-proportional operation up to a few hundred mW average power.

VI. CONCLUSION

The P2012 architecture provides a flexible implementation fabric to span the wide efficiency spectrum between fully programmable homogeneous many-cores and application specific accelerators. It is designed from the ground up for variability tolerance, energy management, reliability and comes with a complete set of programming models and tools to accelerate ecosystem buildup and productive software development. The P2012 SDK and virtual platforms are available today. The first P2012 is a 3D-stacking-ready SoC in 28nm technology with four homogeneous clusters, which will be sampling in Q4 2012.

REFERENCES

[1] F. Arnaud, S. Colquhoun, A.L. Mareau, S. Kohler, S. Jeannot, F. Hasbani, R. Paulin, S. Cremer, C. Charbuillet, G. Druais, P. Scheer, "Technology-Circuit Convergence for Full-SOC Platform in 28 nm and Beyond, International Electron Devices Meeting, 2011.
[2] Y Thonnart, P. Vivet, F. Clermidy, A fully-asynchronous low-power framework for GALS NoC integration, DATE 2010.
[3] MIND Component framework project - Online: mind.ow2.org
[4] Zynq-7000 Extensible Processing Platform – Online: www.xilinx.com
[5] Arria V FPGA SX SoC – Online: www.altera.com