物联网和机器人导论
VIPLE工作流编程

Introduction to IoT and Robotics,
based on Visual Programming
Experiments

Yinong Chen
Arizona State University, U.S.A.

Lectures of the Course

http://neptune.fulton.ad.asu.edu/VIPLE/

- **ASU VIPLE** can be used as the lab environment in Introduction to Engineering's fr
. They can be together with the VIPLE tutorial, which is a lab manual for writi
  - L01 - About the Course and Syllabus
  - L02 - CS Related Disciplines
  - L03 - VIPLE - Visual IoT/Robotics Programming Language Environment
  - L04 - ALU Simulation in VIPLE
  - L05 - Number systems
  - L06 - Finite State Machine and Programmig
  - L07 - Algorithms
  - L08 - Event Driven Programming
  - L09 - Programming Langauges
  - L10 - Operating System
  - L11 - Unix and Edison
  - L12 - IoT and RaaS
  - L13 - IoT and Augmented Reality
  - L14 - from OOC to SOC
  - L15 - SOC and Web Software
  - L16 - Presentation Techniques
  - L17 - Big Data
  - L18 - Cloud Computing
  - L19 - Amdahls Law
  - L20 - Ethics Theories

Lecture Outline

1. Introduction to VIPLE
2. General-Purpose Programming
3. Service-Oriented Programming
4. Parallel Programming
5. Event-Driven Programming

Workflow and Visual Programming

- Workflow defines a system through
  - Architecture: Service oriented using orchestration or chorography style
  - Interface: Partners and roles between the components
  - Behaviors: Activities and execution orders of activities
- Workflow is not necessarily visual
- BPEL is a workflow language, and it is text-based (XML)
- Workflow can be easily visualized and is often defined visually
- Oracle SOA Suite and JDeveloper visualize BPEL

Application of Visual Programming Languages

- Most workflow languages today are visualized;
- Simplified workflow languages are used in education:
  - MIT: Scratch - Visual Game Programming
  - University of Virginia and Carnegie Mellon: Alice Visual Game Programming
  - MIT App Inventor: Phone App Visual Programming
  - Lego NXT & EV3 – Visual Robotics Application Development
  - Microsoft Robotics Developer Studio Visual Programming Language (MRDS VPL)
  - ASU VIPLE: Visual IoT/Robotics Programming Language Environment

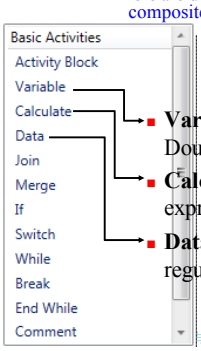
VIPLE Programming Paradigms

VIPLE Features

- General-purpose control flow programming
- Service-oriented computing, supporting RESTful and WSDL
- Parallel / multithreading programming, with underlying threads safety
- Event-driven programming, with built-in and custom events
- Workflow and visual programming
- IoT and Robotics programming

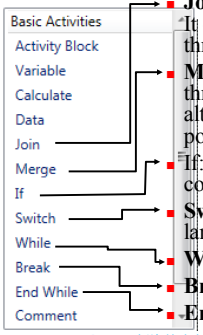
7 Basic Activities of VIPLE

There are dozen of basic activities, and many composite services in VIPLE Repository



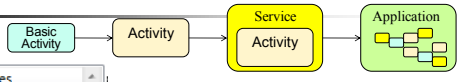
- **Variable**: supports basic types (Int32, Double, String, Boolean, etc.).
- **Calculate**: Calculate the value of any C# expression.
- **Data**: Introducing the constant values in regular programming language

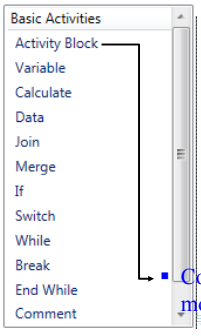
8 Basic Activities (cont.)



- **Join**: proceeds when all threads arrive; It can be used for waiting parallel data or threads.
- **Merge**: proceeds when one of the data or threads arrives. It can be used for waiting alternative data, e.g., creating the returning point of a loop;
- **If**: same as regular programming language construct; It allows multiple conditions.
- **Switch**: same as regular programming language construct;
- **While**: start a loop;
- **Break** exits a loop, and
- **End While** returns to While

9 Basic Activities (cont.)





- Basic Activity and Activity are building blocks of a diagram or flowchart.
- Activity can be wrapped into a service
- Data transfer between
  - activities are through global variables and parameter passing
  - services don't support global
- Construct a composite activity (block or module)

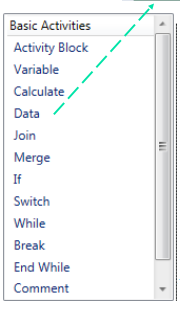
10 VIPLE Services

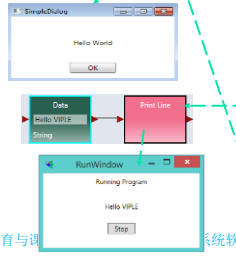
General-purpose and event services

Generic robotic services

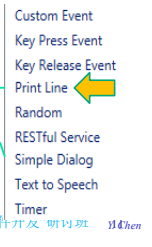
Vendor-specific robotic services

VIPLE Programming: Output

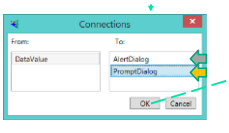


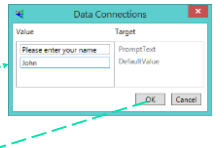


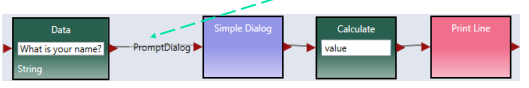
General-purpose and event services



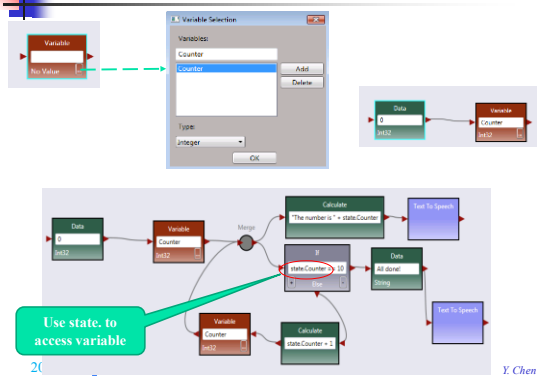
12 VIPLE Programming: Input



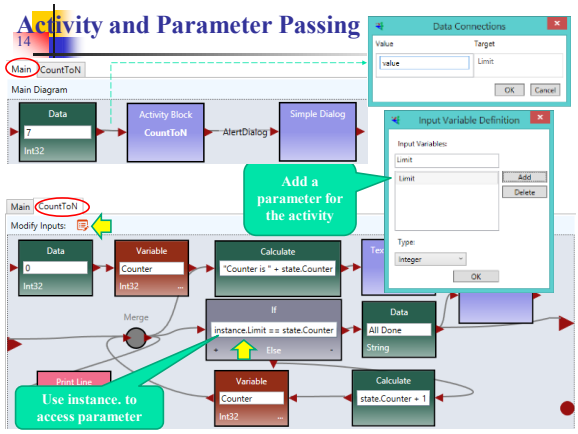




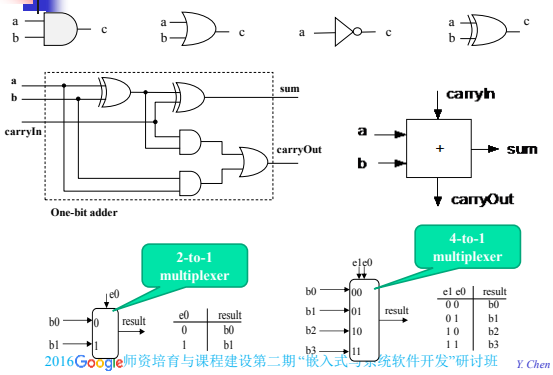
13 VIPLE Programming: Variable and Loop



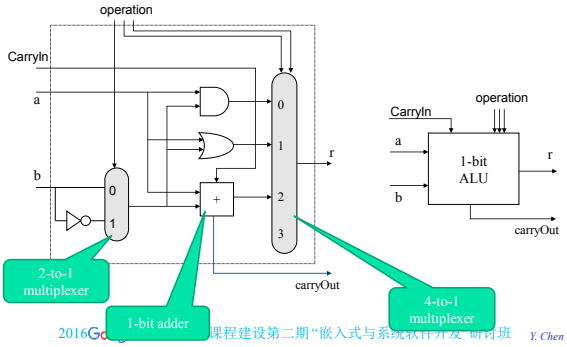
14 Activity and Parameter Passing



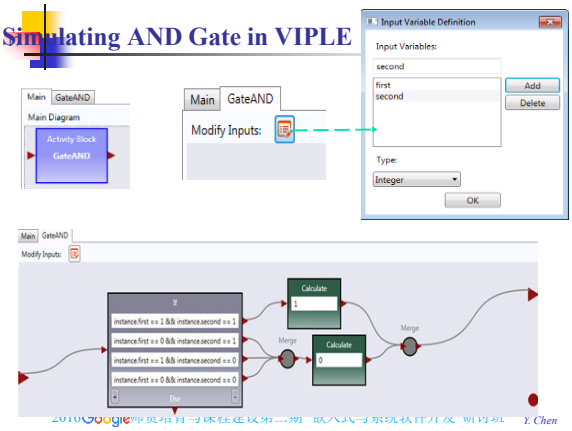
15 Case Study: Simulating an ALU



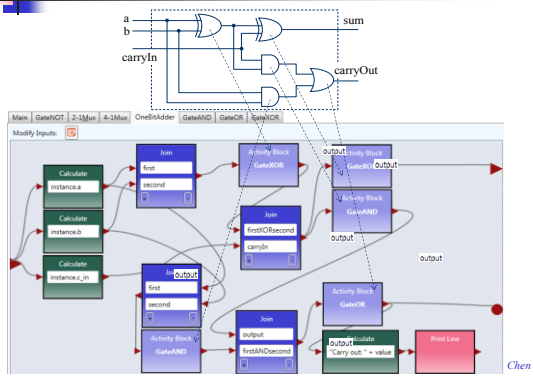
16 One-Bit ALU



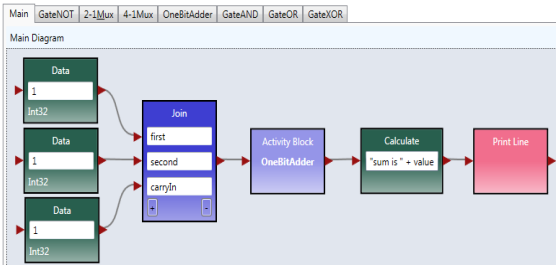
Simulating AND Gate in VIPLE



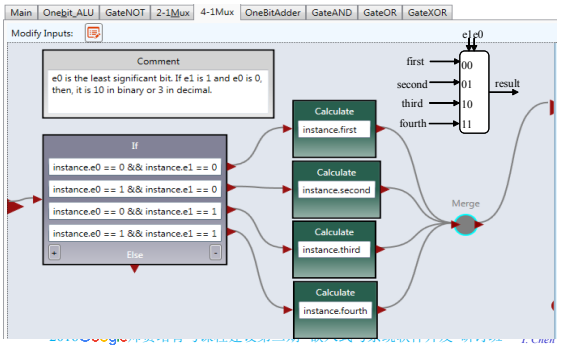
18 Case Study: Building the One-Bit Adder



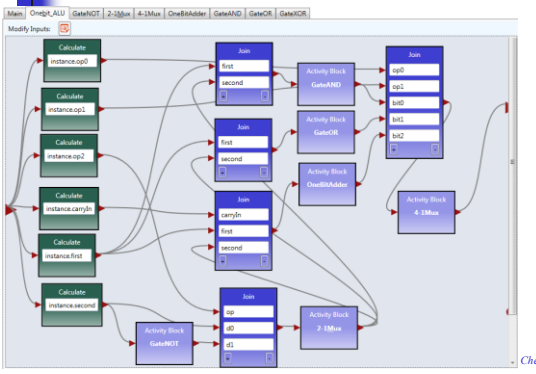
19 Testing the One-Bit Adder



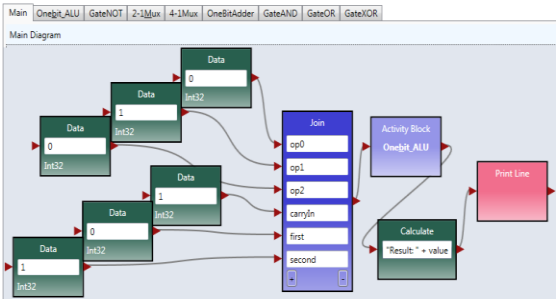
20 Creating 2-1 Multiplexor



21 Creating One-Bit ALU



22 Testing One-Bit ALU



23 Automated Test Case Generation

- Manual testing is time consuming and tedious
- Analyze the test case generation for 1-bit adder

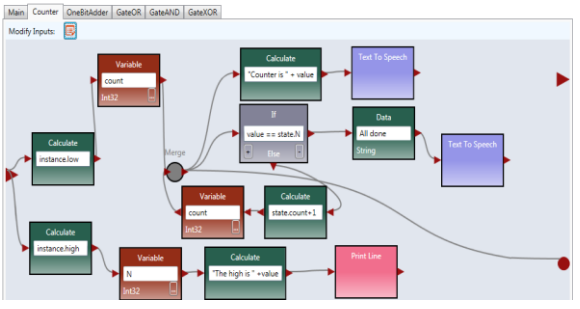
CountTo7	a	b	carryIn
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

if CountTo7 = 0, 1, 2, 3, then a = 0, else a = 1;

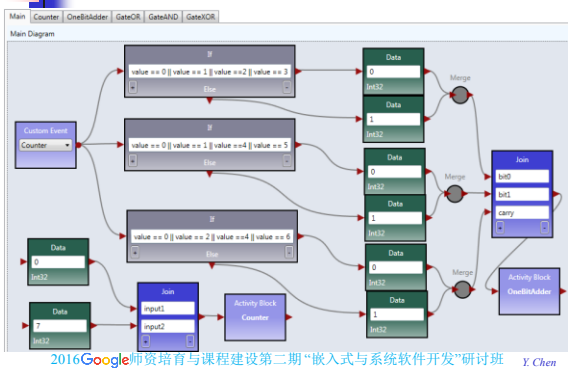
if CountTo7 = 0, 1, 4, 5, then b = 0, else b = 1;

if CountTo7 = 0, 2, 4, 6, then carryIn = 0, else carryIn = 1;

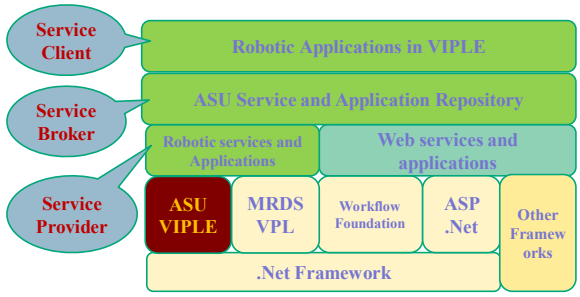
24 Counter that Generate Test Cases as Events



25 Automated Testing of One-Bit Adder

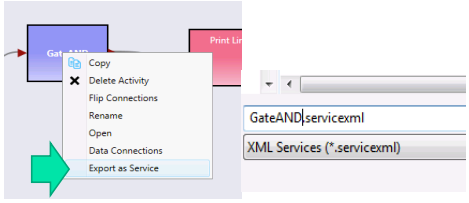


ASU VIPLE is Service Oriented



27 Converting an Activity into a Service

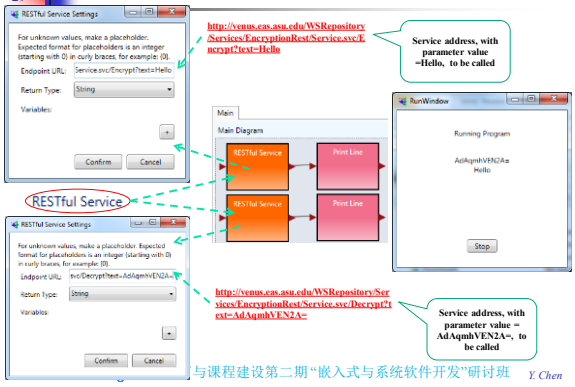
- An activity is a part of an application
- It cannot be reused in another application
- To convert an activity into a service: Right click:



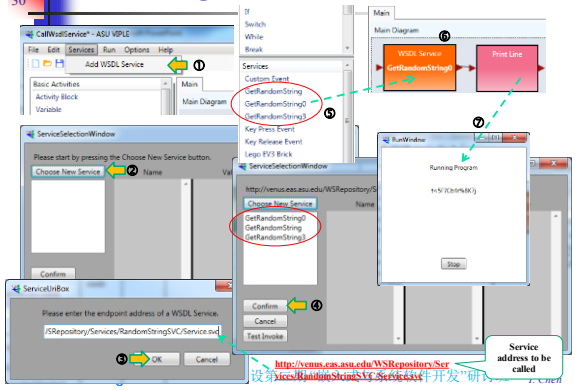
28 After a Custom Service is Created

- When you “Export as a Service”, you can save the service anywhere you want.
- By default, it will be saved into the “CustomServices” folder in your VIPLE program folder.
- When VIPLE is started, all services will be imported into the VIPLE service list, where you find the other services like Print Line and Text to Speech.
- To delete (remove) a custom service, open the folder CustomServices and delete the file of the service. After you restart the service, the custom service will disappear from the service list.
- To share a service in another application, copy the service file into the CustomServices folder of another application.

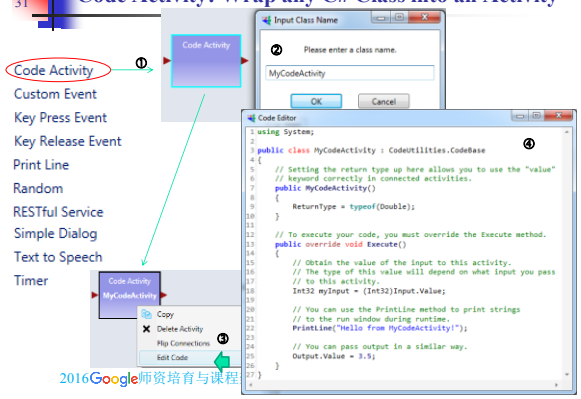
29 Calling RESTful Services



30 Calling a WSDL Service



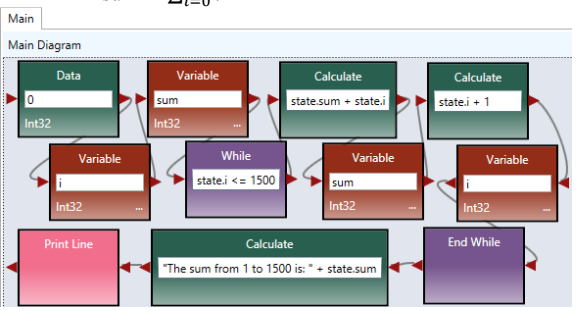
31 Code Activity: Wrap any C# Class into an Activity



32 Sequential vs. Parallel Computing

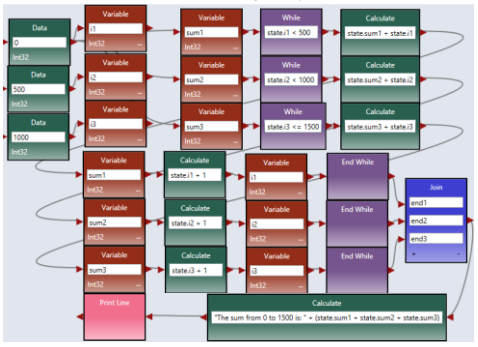
- Sequential version of adding many numbers

$$\text{sum} = \sum_{i=0}^{n} i$$



33 Parallel / Distributed Computing

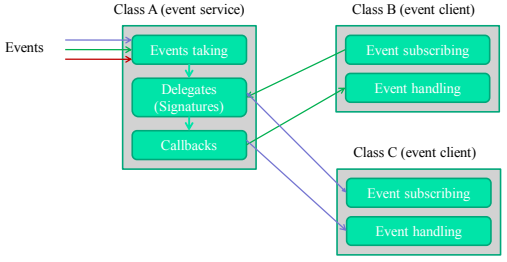
- Parallel version of adding many numbers



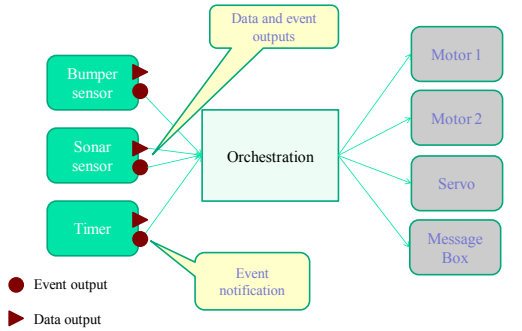
34 Events and Event Handling

- A common technique in distributed computing
  - XML validation and handling
  - Exceptions and handling
  - Mouse click and code processing the click
  - Sensory input arrived (touch sensor) and the action
  - A timer elapsed and the action
- Event-driven computing assumes there are multiple processors to handle events in parallel
- Event handling process
  - Class A publishes event delegates (signatures) for subscription;
  - Class B implements an event handler and subscribes to an event delegate by adding the handler name into the delegate;
  - When an event occurs in class A, class A will callback the handler in class B, which handles the event.

35 Events and Event Handling (Contd.)



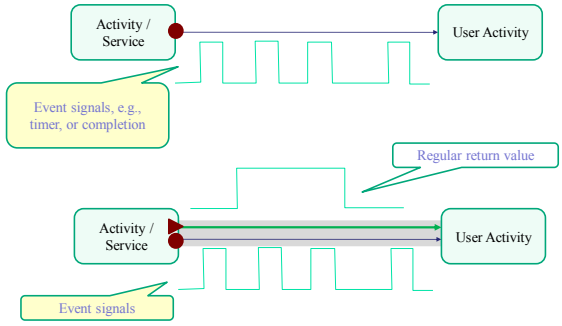
36 Concurrency and Events in Robotics Programming



37 Concurrency and Events in Robotics Programming

- Handling sensory inputs and controlling actuators must be dealt with concurrently, as otherwise sensor inputs can easily be ignored and actuators can get starved.
- Orchestration and composition should not be in control flow model. Event-driven model is a better way to handle such applications.
- Event notification can be sensible alone, or in combination with the return data

38 Event-Driven Notification

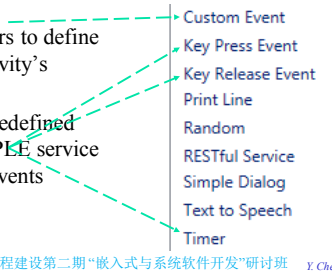


39 Event-Driven Programming

VIPLE supports two types of events

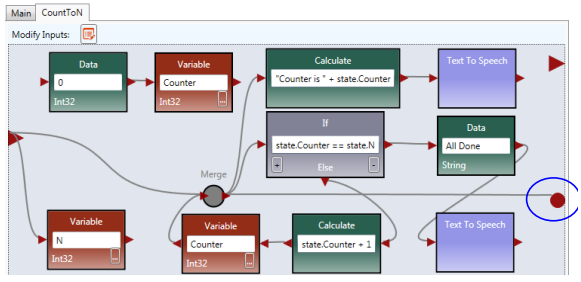
- Custom events: Allow programmers to define an event as an activity's output
- Built-in events: Predefined services in the VIPLE service list that generate events

General-purpose and event services



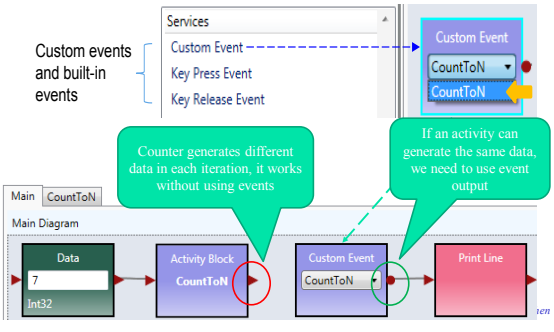
40 Event-Driven Programming: Custom Event

- Implementing the CountToN activity with event output



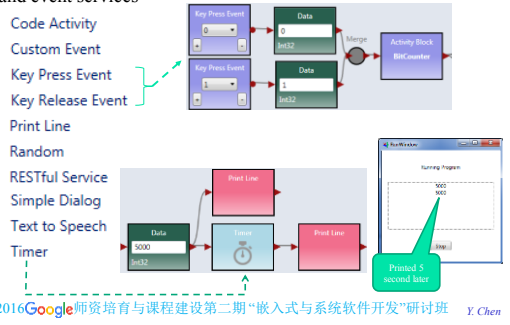
41 Event-Driven Programming: Custom Event

- Accessing the custom event



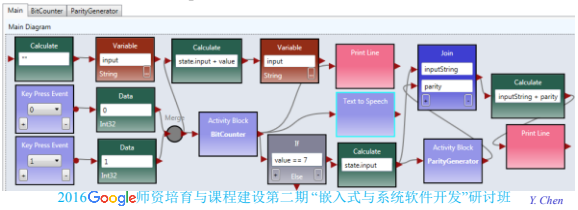
42 Event-Driven Programming: Key Press Event

General-purpose and event services



Parity Bit Generation Using Key Press Event

- An ASCII code consists of 7 bits of 0s or 1s.
- The 8th bit is often generated for parity checking:
  - If the first 7 bits has odd number of 1s, the 8th bit is 0, otherwise, it is 1, to keep the total number of 1s is an odd number.
- Write a VIPLE application to generate the odd-parity bit of an ASCII code. Example:



Next Lecture …

- Finite State Machine (FSM)
- Maze navigation
- Robotics Programming in Simulation