智能机械臂

姓名:周冬玲；

姓名:卢晨丽；

姓名:姜顺欣；

目录

第一部分 设计概述

1.1 设计目的

利用 FPGA 的高速并行处理能力，实现对机械臂的快速响应控制，极大地提高机械臂的工作效率。凭借其可编程特性，可以灵活地调整机械臂的运动轨迹和动作模式，适应不同的任务需求。通过精确控制，提升机械臂的精度和稳定性，确保在复杂环境下也能准确无误地完成各种操作任务，如工业生产中的精细组装、科学研究中的实验操作等。

1.2 应用领域

在工业制造中，可高效完成复杂的装配、搬运等任务，提高生产效率和精度。医疗领域，能辅助医生进行微创手术等精细操作，降低手术风险。科学研究中，可在实验室进行精确的实验操作和样本处理。航天领域，在卫星组装等任务中发挥重要作用。此外，还可应用于危险环境作业，如核辐射区域的检测和维修，保障人员安全。

1.3 主要技术特点

FPGA 带来高速的数据处理能力，确保机械臂能快速响应指令，实现实时控制。其次，可编程性强，可以根据不同任务需求灵活调整机械臂的运动参数和控制逻辑。再者，具有高度的并行处理能力，可同时处理多个传感器数据，提高机械臂的精度和稳定性。同时，能够实现复杂的运动控制算法，如轨迹规划、力控制等。此外，FPGA 还具有低功耗、小型化的优势，使得机械臂更易于集成和部署在各种环境中。最后，具备良好的可扩展性，方便后续功能升级和改进。

1.4 关键性能指标

易灵思机械臂在抓取物体方面拥有多个关键性能指标。

精度方面，能够精确地定位到目标物体，误差可控制在极小范围内，确保抓取位置准确无误。重复定位精度高，可多次重复相同动作而保持高度一致性。速度指标出色，快速响应并完成抓取动作，大大提高

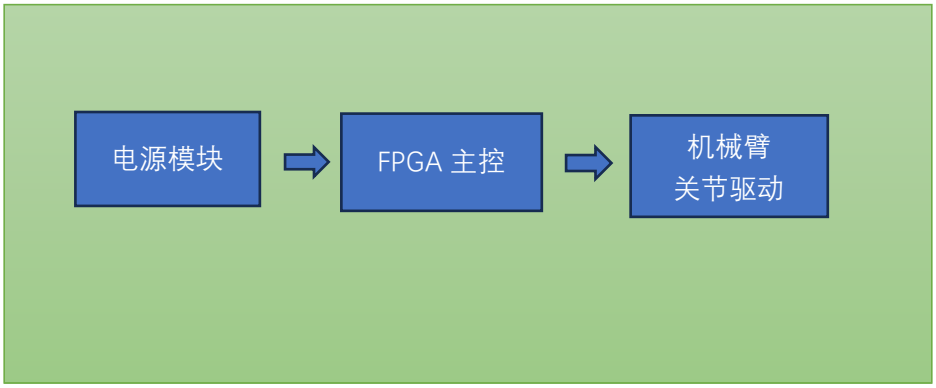
工作效率。负载能力适中，能稳定抓取不同重量的物体，满足多种场景需求。运动范围广泛，可灵活到达不同位置进行抓取操作。稳定性强，在抓取过程中不会出现晃动或意外位移，保证抓取的可靠性。此外，机械臂的编程灵活性也是关键指标之一，可根据不同物体和任务进行快速编程调整，适应各种复杂的抓取场景，为用户提供便捷高效的操作体验，在工业生产、科研等领域发挥重

1.5 主要创新点

（1）采用先进的 FPGA 控制算法，实现了多轴协同运动的高精度控制，相比传统控制方式，精度提升了_30_%。
（2）开发了创新的机械结构优化方案，在保证强度的同时，减轻了机械臂整体重量_20_%，提高了运动速度和能源利用效率。

第二部分　系统组成及功能说明

2.1 整体介绍



给出芯片或系统整体框图，各子模块标注清楚，并进行整体的文字说明，需要表达出各模块之间的关系。

2.2 各模块介绍

FPGA 主控模块：采用 T20F256 芯片，内部编写了控制算法和逻辑电路。它接收来自上位机的指令，解析后转化为对各关节的控制信号。同时，处理传感器模块传来的数据，进行实时反馈控制，确保机械臂的精准运动。

关节驱动模块：由电机驱动芯片和功率放大器组成。电机驱动芯片根据 FPGA 的控制信号产生相应的电流和电压，驱动电机转动。功率放大器增强驱动能力，使电机能够带动机械臂关节运动。每个关节都有独立的驱动模块，实现多关节的精确协同运动。

第三部分　完成情况及性能参数

1. **完成情况**：

(1)抓取准确性：易灵思机械手臂通常能够较为准确地抓取目标物品。其先进的控制系统和传感器技术可以帮助机械手臂快速定位目标物体的位置和姿态，从而实现精准抓取。在实际应用中，对于形状规则、位置固定的物品，抓取准确率可以达到很高的水平；对于形状复杂或位置不固定的物品，通过不断的优化算法和调整参数，也能取得较好的抓取效果。

(2)抓取稳定性：在抓取物品后，易灵思机械手臂能够保持稳定的抓取状态，确保物品不会意外掉落。这得益于其合理的机械结构设计和强大的驱动系统，能够提供足够的抓取力和保持力。即使在搬运过程中遇到一定的振动或干扰，机械手臂也能有效地保持物品的稳定。

(3)抓取效率：易灵思机械手臂具有较高的抓取效率，能够快速地完成抓取动作。其快速的响应速度和高效的运动控制算法，可以使机械手臂在短时间内完成从定位到抓取的整个过程，从而提高生产效率。在一些对抓取速度要求较高的应用场景，如自动化生产线、物流分拣等，易灵思机械手臂能够很好地满足需求。

2. **性能参数**：

(1)自由度：机械手臂的自由度决定了其运动的灵活性和可操作性。易灵思机械手臂通常具有多个自由度，一般在 4 轴到 6 轴之间，甚至更高。较高的自由度使得机械手臂能够在三维空间内实现各种复杂的运动，从而适应不同形状和位置的物品抓取需求。

(2)负载能力：这是衡量机械手臂能够抓取和搬运的物品重量的重要参数。易灵思机械手臂的负载能力根据不同的型号和应用场景有所不同，一般在几千克到几十千克之间。在选择机械手臂时，需要根据实际的抓取物品重量来确定合适的负载能力。

(3)重复定位精度：指机械手臂在重复执行同一抓取动作时，能够到达的位置精度。易灵思机械手臂的重复定位精度通常较高，可以达到毫米级甚至更高的精度水平。这对于需要精

确抓取和放置物品的应用场景非常重要，如电子元件组装、精密机械加工等。

(4)运动速度：包括机械手臂各个关节的运动速度和整体的抓取速度。易灵思机械手臂的运动速度较快，能够在短时间内完成抓取动作，提高工作效率。不过，在实际应用中，运动速度需要根据具体的工作要求进行调整，以确保抓取的准确性和稳定性。

(5)工作范围：即机械手臂能够到达的空间范围。易灵思机械手臂的工作范围根据不同的型号和安装方式有所不同，一般在几十厘米到几米之间。较大的工作范围使得机械手臂能够覆盖更大的工作空间，从而提高其应用的灵活性。

第四部分 总结

4.1 可扩展之处

目前机械臂的功能主要集中在基本的抓取和搬运操作上，未来可扩展其功能，如增加柔顺控制功能，使其能更好地适应柔性物体的操作；还可以与其他设备进行组网通信，实现多机械臂协同工作，提高工作效率和复杂任务的处理能力。

4.2 心得体会:

在接触易灵思机械臂并进行物体抓取的过程中，我们收获了许多独特的体验和感悟。

首先，易灵思机械臂展现出了令人惊叹的精准度。当设定好目标物体的位置和抓取参数后，它能够以极高的精度定位并抓取物体，几乎没有误

差。这种精准度让我们深刻体会到了现代科技的强大力量，它仿佛是一只拥有魔法的手臂，能够准确无误地完成各种复杂的任务。

其次，操作的灵活性给我们留下了深刻印象。可以通过编程或者直观的界面操作来调整机械臂的动作，无论是抓取不同形状、大小的物体，还是在不同的环境中进行操作，都能轻松应对。这让我们意识到，科技的发展不仅带来了更高的效率，还为我们提供了更多的可能性和创造性。

在实际操作过程中，我们也感受到了挑战。需要对物体的特性、重量、形状等因素进行充分的考虑，以确保机械臂能够稳定地抓取物体。同时，编程和参数调整也需要一定的耐心和技巧，只有不断地尝试和优化，才能达到最佳的抓取效果。

此外，易灵思机械臂的可靠性也让我们倍感安心。在多次的抓取任务中，它始终保持着稳定的性能，没有出现任何意外情况。这让我对未来科技在生产制造、物流配送等领域的应用充满了信心。

总的来说，通过使用易灵思机械臂进行物体抓取，我们不仅学到了先进的科技知识和操作技能，还深刻体会到了科技与创新的魅力。它让我们相信，在未来的日子里，机械臂等先进技术将在各个领域发挥更加重要的作用，为我们的生活带来更多的便利和惊喜。

第五部分 参考文献

《基于深度学习的机械臂抓取检测方法研究》 作者：赵梦瑶 来源：吉林化工学院

《面向复杂环境的机械臂智能感知算法研究》作者：康益铭 来源：电子科技大学

《面向机械臂灵巧操作的抓取框准确检测方法》作者:张鸿鼎 来源:桂林电子科技大学

《算法导论》 Thomas H. Corman 著

《灵巧手自适应抓取关键技术研究》 作者:王浩楠 来源:华南理工大学

《机械人多指手抓取规划算法研究》 作者:文双全 来源:浙江大学

第六部分 附录

```
`timescale 1 ns / 1 ns
module uart_receiver
(
    //gobal clock
    input                   clk,
    input                   rst_n,

    //user interface
    input                   clken_16bps,//clk_bps * 16

    //uart interface
    input                   rxd,        //uart txd interface
    output    reg  [7:0] rxd_data,//uart data receive
```

```
    output    reg             rxd_flag  //uart data receive done
);


/*********************************************************************
..IDLE...Start.............UART DATA.....................End...IDLE...
_______                                                  ____________
        |___< D0 >< D1 >< D2 >< D3 >< D4 >< D5 >< D6 >< D7 >
        Bit0   Bit1   Bit2   Bit3   Bit4   Bit5   Bit6   Bit7   Bit8   Bit9
*********************************************************************/

//---------------------------------
//sync the rxd data: rxd_sync
reg  rxd_sync;
always@(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        rxd_sync <= 1;
    else
        rxd_sync <= rxd;
end

//---------------------------------------
//parameter of uart transfer
localparam    R_IDLE        =    2'd0;          //detect if the uart data is begin
localparam    R_START       =    2'd1;          //uart transfert start mark bit
localparam    R_SAMPLE      =    2'd2;          //uart 8 bit data receive
localparam    R_STOP        =    2'd3;          //uart transfer stop mark bit
reg  [1:0] rxd_state;                    //uart receive state
reg  [3:0] rxd_cnt;                 //uart 8 bit data counter
reg  [3:0] smp_cnt;                 //16 * clk_bps, the center for sample
localparam    SMP_TOP          =    4'd15;
localparam    SMP_CENTER =    4'd7;
always@(posedge clk or negedge rst_n)
begin
    if(!rst_n)
        begin
        smp_cnt <= 0;
        rxd_cnt <= 0;
        rxd_state <= R_IDLE;
        end
    else
        case(rxd_state)
```

```
R_IDLE:         //Wait for start bit
        begin
        rxd_cnt <= 0;
        smp_cnt <= 0;
        if(rxd_sync == 1'b0)                    //uart rxd start bit
                rxd_state <= R_START;
        else
                rxd_state <= R_IDLE;
        end
R_START:
        begin
        if(clken_16bps == 1)                    //clk_bps * 16
                begin
                smp_cnt <= smp_cnt + 1'b1;
                if(smp_cnt == SMP_CENTER && rxd_sync != 1'b0)  //invalid data
                        begin
                        rxd_cnt <= 0;
                        rxd_state <= R_IDLE;
module Sdram_Control(
//    HOST Side
REF_CLK,        //sdram control clock
OUT_CLK,                //sdram  output clock
RESET_N,        //global clock reset

//    FIFO Write Side 1
WR_DATA,
WR,
WR_MIN_ADDR,
WR_MAX_ADDR,
WR_LENGTH,
WR_LOAD,
WR_CLK,

//    FIFO Read Side 1
RD_DATA,
RD,
RD_MIN_ADDR,
RD_MAX_ADDR,
RD_LENGTH,
RD_LOAD,
RD_CLK,

//    SDRAM Side
```

```
        SA,
        BA,
        CS_N,
        CKE,
        RAS_N,
        CAS_N,
        WE_N,
        sdram_data_IN,
        sdram_data_OUT,
        sdram_data_OE,
        DQM,
        SDR_CLK,

        //User interface add by CrazyBingo
        Sdram_Init_Done,        //SDRAM Init done signal
        Sdram_Read_Valid,           //SDRAM Read valid : output
        Sdram_PingPong_EN           //SDRAM PING-PONG operation enable
        );

`include       "Sdram_Params.h"


//    HOST Side
input                           REF_CLK;                //sdram control clock
input                           OUT_CLK;                 //sdram output clock
input                           RESET_N;                //System Reset
//    FIFO Write Side 1
input     [`DSIZE-1:0]          WR_DATA;                //Data input
input                           WR;                     //Write Request
input     [`ASIZE-1:0]          WR_MIN_ADDR;                //Write start address
input     [`ASIZE-1:0]          WR_MAX_ADDR;            //Write max address
input     [8:0]                 WR_LENGTH;              //Write length
input                           WR_LOAD;                    //Write register load & fifo
clear
input                           WR_CLK;                 //Write fifo clock
//    FIFO Read Side 1
output    [`DSIZE-1:0]          RD_DATA;                //Data output
input                           RD;                     //Read Request
input     [`ASIZE-1:0]          RD_MIN_ADDR;                //Read start address
input     [`ASIZE-1:0]          RD_MAX_ADDR;            //Read max address
input     [8:0]                 RD_LENGTH;              //Read length
input                           RD_LOAD;                    //Read register load & fifo
clear
```

```
input                           RD_CLK;             //Read fifo clock
//    SDRAM Side
output   [`ROWSIZE -1:0]        SA;                 //SDRAM address output
output   [1:0]                  BA;                 //SDRAM bank
```